MATLAB 智能算法 30 个案例分析

第 1 章

1、案例背景

遗传算法（Genetic Algorithm，GA）是一种进化算法，其基本原理是仿效生物界中的“物竞天择、适者生存”的演化法则。遗传算法的做法是把问题参数编码为染色体，再利用迭代的方式进行选择、交叉以及变异等运算来交换种群中染色体的信息，最终生成符合优化目标的染色体。

在遗传算法中，染色体对应的是数据或数组，通常是由一维的串结构数据来表示，串上各个位置对应基因的取值。基因组成的串就是染色体，或者叫基因型个体(Individuals) 。一定数量的个体组成了群体（Population)。群体中个体的数目称为群体大小（Population Size），也叫群体规模。而各个个体对环境的适应程度叫做适应度(Fitness) 。

2、案例目录：

3、主程序：

1. 简单一元函数优化：

```
clc
clear all
close all
%% 画出函数图
figure(1);
hold on;
lb=1;ub=2; %函数自变量范围【1,2】
ezplot('sin(10*pi*X)/X',[lb,ub]); %画出函数曲线
xlabel('自变量/X')
ylabel('函数值/Y')
%% 定义遗传算法参数
NIND=40; %个体数目
MAXGEN=20; %最大遗传代数
PRECI=20; %变量的二进制位数
GGAP=0.95; %代沟
px=0.7; %交叉概率
pm=0.01; %变异概率
trace=zeros(2,MAXGEN); %寻优结果的初始值
FieldD=[PRECI;lb;ub;1;0;1;1]; %区域描述器
Chrom=crtbp(NIND,PRECI); %初始种群
%% 优化
gen=0; %代计数器
X=bs2rv(Chrom,FieldD); %计算初始种群的十进制转换
ObjV=sin(10*pi*X)./X; %计算目标函数值
while gen<MAXGEN
FitnV=ranking(ObjV); %分配适应度值
SelCh=select('sus',Chrom,FitnV,GGAP); %选择
SelCh=recombin('xovsp',SelCh,px); %重组
SelCh=mut(SelCh,pm); %变异
X=bs2rv(SelCh,FieldD); %子代个体的十进制转换
ObjVSel=sin(10*pi*X)./X; %计算子代的目标函数值
[Chrom,ObjV]=reins(Chrom,SelCh,1,1,ObjV,ObjVSel); %重插入子代到父代，得到新种群
X=bs2rv(Chrom,FieldD);
gen=gen+1; %代计数器增加
```

```
%获取每代的最优解及其序号，Y 为最优解,I 为个体的序号
[Y,I]=min(ObjV);
trace(1,gen)=X(I); %记下每代的最优值
trace(2,gen)=Y; %记下每代的最优值
end
plot(trace(1,:),trace(2,:),'bo'); %画出每代的最优点
grid on;
plot(X,ObjV,'b*'); %画出最后一代的种群
hold off
%% 画进化图
figure(2);
plot(1:MAXGEN,trace(2,:));
grid on
xlabel('遗传代数')
ylabel('解的变化')
title('进化过程')
bestY=trace(2,end);
bestX=trace(1,end);
fprintf(['最优解:\nX=',num2str(bestX),'\nY=',num2str(bestY),'\n'])
```

2. 多元函数优化

```
clc
clear all
close all
%% 画出函数图
figure(1);
lbx=-2;ubx=2; %函数自变量 x 范围【-2,2】
lby=-2;uby=2; %函数自变量 y 范围【-2,2】
ezmesh('y*sin(2*pi*x)+x*cos(2*pi*y)',[lbx,ubx,lby,uby],50); %画出函数曲线
hold on;
%% 定义遗传算法参数
NIND=40; %个体数目
MAXGEN=50; %最大遗传代数
PRECI=20; %变量的二进制位数
GGAP=0.95; %代沟
px=0.7; %交叉概率
pm=0.01; %变异概率
trace=zeros(3,MAXGEN); %寻优结果的初始值
FieldD=[PRECI PRECI;lbx lby;ubx uby;1 1;0 0;1 1;1 1]; %区域描述器
Chrom=crtbp(NIND,PRECI*2); %初始种群
%% 优化
gen=0; %代计数器
XY=bs2rv(Chrom,FieldD); %计算初始种群的十进制转换
X=XY(:,1);Y=XY(:,2);
```

```
ObjV=Y.*sin(2*pi*X)+X.*cos(2*pi*Y); %计算目标函数值
while gen<MAXGEN
FitnV=ranking(-ObjV); %分配适应度值
SelCh=select('sus',Chrom,FitnV,GGAP); %选择
SelCh=recombin('xovsp',SelCh,px); %重组
SelCh=mut(SelCh,pm); %变异
XY=bs2rv(SelCh,FieldD); %子代个体的十进制转换
X=XY(:,1);Y=XY(:,2);
ObjVSel=Y.*sin(2*pi*X)+X.*cos(2*pi*Y); %计算子代的目标函数值
[Chrom,ObjV]=reins(Chrom,SelCh,1,1,ObjV,ObjVSel); %重插入子代到父代，得到新种群
XY=bs2rv(Chrom,FieldD);
gen=gen+1; %代计数器增加
%获取每代的最优解及其序号，Y 为最优解,I 为个体的序号
[Y,I]=max(ObjV);
trace(1:2,gen)=XY(I,:); %记下每代的最优值
trace(3,gen)=Y; %记下每代的最优值
end
plot3(trace(1,:),trace(2,:),trace(3,:),'bo'); %画出每代的最优点
grid on;
plot3(XY(:,1),XY(:,2),ObjV,'bo'); %画出最后一代的种群
hold off
%% 画进化图
figure(2);
plot(1:MAXGEN,trace(3,:));
grid on
xlabel('遗传代数')
ylabel('解的变化')
title('进化过程')
bestZ=trace(3,end);
bestX=trace(1,end);
bestY=trace(2,end);
fprintf(['最优解:\nX=',num2str(bestX),'\nY=',num2str(bestY),'\nZ=',num2str(bestZ), '\n'])
```

第 2 章　**基于遗传算法和非线性规划的函数寻优算法**

1.1 案例背景

1.1.1 非线性规划方法

非线性规划是 20 世纪 50 年代才开始形成的一门新兴学科。1951 年 H.W.库恩和 A.W.塔克发表的关于最优性条件(后来称为库恩.塔克条件)的论文是非线性规划正式诞生的一个重要标志。

非线性规划研究一个 n 元实函数在一组等式或不等式的约束条件下的极值问题，且目标函数和约束条件至少有一个是未知量的非线性函数。非线性规划的一个重要理论是 1951 年 Kuhn-Tucker 最优条件（简称 KT 条件）的建立。此后的 50 年代主要是对梯度法和牛顿法的研究。以 Davidon（1959），Fletcher 和 Powell（1963）提出的 DFP 方法为起点，60 年代是研究拟牛顿方法活跃时期，同时对共轭梯度法也有较好的研究。在 1970 年由 Broyden，Fletcher、Goldfarb 和 Shanno 从不同的角度共同提出的 BFGS 方法是目前为止最有效的拟牛顿方法。由于 Broyden，Dennis 和 More 的工作使得拟牛顿方法的理论变得很完善。 70 年代是非线性规划飞速发展时期， 约束变尺度（SQP）方法（Han 和 Powell 为代表）和 Lagrange 乘子法（代表人物是 Powell 和 Hestenes）是这一时期主要研究成果.计算机的飞速发展使非线性规划的研究如虎添翼。80 年**始研究信赖域法、稀疏拟牛顿法、大规模问题的方法和并行计算，90 年代研究解非线性规划问题的内点法和有限储存法。可以毫不夸张的说，这半个世纪是最优化发展的黄金时期。

1.1.2 非线性规划函数

fmincon 函数是 Matlab 最优化工具箱中用来求解非线性规划问题的重要函数，它从一个预估值出发，搜索约束条件下非线性多元函数的最小值。

1.1.3 案例

案例 1：

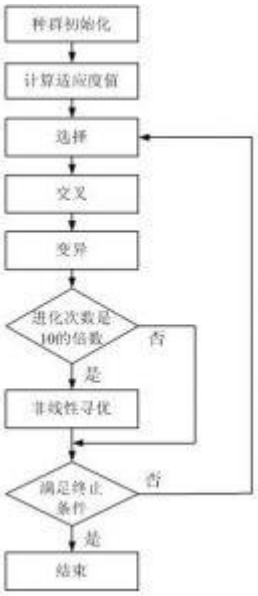
将遗传算法和 MATLAB 非线性寻优函数相结合，求解函数

$$f(x) = -5\sin x_1 \sin x_2 \sin x_3 \sin x_4 \sin x_5 - \sin 5x_1 5x_2 5x_3 5x_4 5x_5 + 8$$

的最小值，其中，x_1、x_2、x_3、x_4、x_5 是 0 到 0.9π 之间的实数。

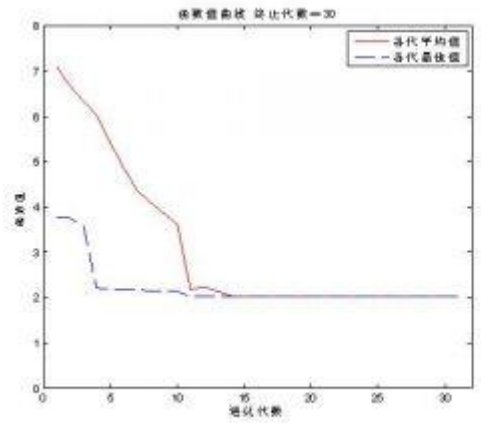
1.2 模型建立

算法流程图如下：

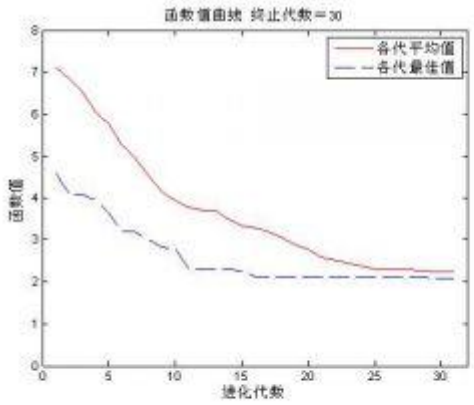


1.3 仿真结果

非线性遗传算法寻优结果如下：



普通遗传算法寻优结果如下：



主函数代码如下：

```
%% 清空环境
clc
clear

%% 遗传算法参数
maxgen=30; %进化代数
sizepop=100; %种群规模
pcross=[0.6]; %交叉概率
pmutation=[0.01]; %变异概率
lenchrom=[1 1 1 1 1]; %变量字串长度
```

```
bound=[0 0.9*pi;0 0.9*pi;0 0.9*pi;0 0.9*pi;0 0.9*pi]; %变量范围

%% 个体初始化
individuals=struct('fitness',zeros(1,sizepop), 'chrom',[]); %种群结构体
avgfitness=[]; %种群平均适应度
bestfitness=[]; %种群最佳适应度
bestchrom=[]; %适应度最好染色体
% 初始化种群
for i=1:sizepop
individuals.chrom(i,:)=Code(lenchrom,bound); %随机产生个体
x=individuals.chrom(i,:);
individuals.fitness(i)=fun(x); %个体适应度
end

%找最好的染色体
[bestfitness bestindex]=min(individuals.fitness);
bestchrom=individuals.chrom(bestindex,:); %最好的染色体
avgfitness=sum(individuals.fitness)/sizepop; %染色体的平均适应度
% 记录每一代进化中最好的适应度和平均适应度
trace=[avgfitness bestfitness];

%% 进化开始
for i=1:maxgen

% 选择操作
individuals=Select(individuals,sizepop);
avgfitness=sum(individuals.fitness)/sizepop;
% 交叉操作
individuals.chrom=Cross(pcross,lenchrom,individuals.chrom,sizepop,bound);
% 变异操作
individuals.chrom=Mutation(pmutation,lenchrom,individuals.chrom,sizepop,[i maxgen],bound);

% 计算适应度
for j=1:sizepop
x=individuals.chrom(j,:);
individuals.fitness(j)=fun(x);
end

%找到最小和最大适应度的染色体及它们在种群中的位置
[newbestfitness,newbestindex]=min(individuals.fitness);
[worestfitness,worestindex]=max(individuals.fitness);
% 代替上一次进化中最好的染色体
if bestfitness>newbestfitness
bestfitness=newbestfitness;
```

```
bestchrom=individuals.chrom(newbestindex,:);
end
individuals.chrom(worestindex,:)=bestchrom;
individuals.fitness(worestindex)=bestfitness;

avgfitness=sum(individuals.fitness)/sizepop;

trace=[trace;avgfitness bestfitness]; %记录每一代进化中最好的适应度和平均适应度
end
%进化结束

%% 结果显示
[r c]=size(trace);
plot([1:r]',trace(:,1),'r-',[1:r]',trace(:,2),'b--');
title(['函数值曲线 ''终止代数＝' num2str(maxgen)]);
xlabel('进化代数');ylabel('函数值');
legend('各代平均值','各代最佳值');
disp('函数值 变量');
% 窗口显示
disp([bestfitness x]);
```

第 3 章 **基于遗传算法的 BP 神经网络优化算法**

1、案例背景

BP 网络是一类多层的前馈神经网络。它的名字源于在网络训练的过程中，调整网络的权值的算法是误差的反向传播的学习算法，即为 BP 学习算法。BP 算法是 Rumelhart 等人在 1986 年提出来的。由于它的结构简单，可调整的参数多，训练算法也多，而且可操作性好，BP 神经网络获得了非常广泛的应用。据统计，有 80%～90%的神经网络模型都是采用了 BP 网络或者是它的变形。BP 网络是前向网络的核心部分，是神经网络中最精华、最完美的部分。BP 神经网络虽然是人工神经网络中应用最广泛的算法，但是也存在着一些缺陷，例如：
①、学习收敛速度太慢；
②、不能保证收敛到全局最小点；
③、网络结构不易确定。
另外，网络结构、初始连接权值和阈值的选择对网络训练的影响很大，但是又无法准确获得，针对这些特点可以采用遗传算法对神经网络进行优化。

本节以某型号拖拉机的齿轮箱为工程背景，介绍使用基于遗传算法的 BP 神经网络进行齿轮箱故障的诊断。

2、案例目录：

3、主程序：

```
clc
clear all
close all
%% 加载神经网络的训练样本 测试样本每列一个样本 输入 P 输出 T
%样本数据就是前面问题描述中列出的数据
load data
% 初始隐层神经元个数
hiddennum=31;
% 输入向量的最大值和最小值
threshold=[0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1;0 1];
inputnum=size(P,1); % 输入层神经元个数
outputnum=size(T,1); % 输出层神经元个数
w1num=inputnum*hiddennum; % 输入层到隐层的权值个数
w2num=outputnum*hiddennum;% 隐层到输出层的权值个数
N=w1num+hiddennum+w2num+outputnum; %待优化的变量的个数

%% 定义遗传算法参数
NIND=40; %个体数目
MAXGEN=50; %最大遗传代数
PRECI=10; %变量的二进制位数
GGAP=0.95; %代沟
```

```
px=0.7; %交叉概率
pm=0.01; %变异概率
trace=zeros(N+1,MAXGEN); %寻优结果的初始值

FieldD=[repmat(PRECI,1,N);repmat([-0.5;0.5],1,N);repmat([1;0;1;1],1,N)]; %区域描述器
Chrom=crtbp(NIND,PRECI*N); %初始种群
%% 优化
gen=0; %代计数器
X=bs2rv(Chrom,FieldD); %计算初始种群的十进制转换
ObjV=Objfun(X,P,T,hiddennum,P_test,T_test); %计算目标函数值
while gen<MAXGEN
fprintf('%d\n',gen)
FitnV=ranking(ObjV); %分配适应度值
SelCh=select('sus',Chrom,FitnV,GGAP); %选择
SelCh=recombin('xovsp',SelCh,px); %重组
SelCh=mut(SelCh,pm); %变异
X=bs2rv(SelCh,FieldD); %子代个体的十进制转换
ObjVSel=Objfun(X,P,T,hiddennum,P_test,T_test); %计算子代的目标函数值
[Chrom,ObjV]=reins(Chrom,SelCh,1,1,ObjV,ObjVSel); %重插入子代到父代，得到新种群
X=bs2rv(Chrom,FieldD);
gen=gen+1; %代计数器增加
%获取每代的最优解及其序号，Y 为最优解,I 为个体的序号
[Y,I]=min(ObjV);
trace(1:N,gen)=X(I,:); %记下每代的最优值
trace(end,gen)=Y; %记下每代的最优值
end
%% 画进化图
figure(1);
plot(1:MAXGEN,trace(end,:));
grid on
xlabel('遗传代数')
ylabel('误差的变化')
title('进化过程')
bestX=trace(1:end-1,end);
bestErr=trace(end,end);
fprintf(['最优初始权值和阈值:\nX=',num2str(bestX'),'\n 最小误差 err=',num2str(bestErr),'\n'])
```

第 4 章　**基于遗传算法的 TSP 算法**

1、案例背景

TSP (旅行商问题—Traveling Salesman Problem)，是典型的 NP 完全问题，即其最坏情况下的时间复杂性随着问题规模的增大按指数方式增长，到目前为止不能找到一个多项式时间的有效算法。遗传算法是一种进化算法，其基本原理是仿效生物界中的"物竞天择、适者生存"的演化法则。遗传算法的做法是把问题参数编码为染色体，再利用迭代的方式进行选择、交叉以及变异等运算来交换种群中染色体的信息，最终生成符合优化目标的染色体。实践证明，遗传算法对于解决 TSP 问题等组合优化问题具有较好的寻优性能。

2、案例目录：

3、案例实例及结果：

本案例以 14 个城市为例，假定 14 个城市的位置坐标为：

表 4.1 14 个城市的位置坐标

城市编号	X 坐标	Y 坐标
1	16.47	96.1
2	16.47	94.44
3	20.09	92.54
4	22.39	93.37
5	25.23	97.24
6	22	96.05
7	20.47	97.02
8	17.2	96.29

9	16.3	97.38
10	14.05	98.12
11	16.53	97.38
12	21.52	95.59
13	19.41	97.13
14	20.09	92.55

从某个城市出发访问每个城市一次且仅一次， 最后回到出发城市，如何安排才使其所走路线最短。

结果：

优化前的一个随机路线轨迹图

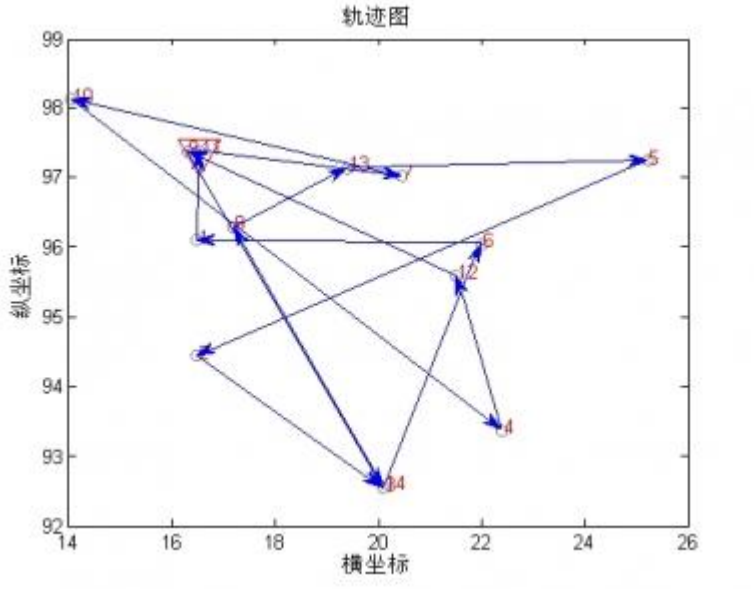


图 4.1 随机路线图

随机路线为：

11—>7—>10—>4—>12—>9—>14—>8—>13—>5—>2—>3—>6—>1—>11

总距离：71.1144

优化后的路线图：

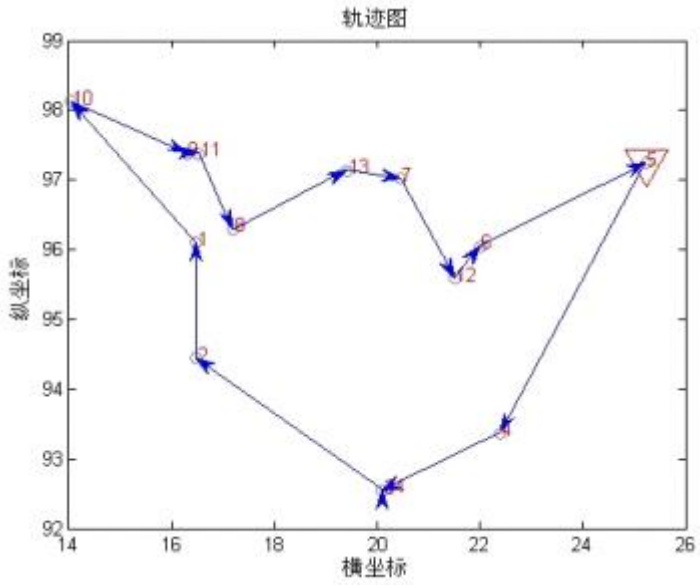


图 4.2 最优解路线图

最优解路线:

5—>4—>3—>14—>2—>1—>10—>9—>11—>8—>13—>7—>12—>6—>5

总距离：29.3405

优化迭代过程：

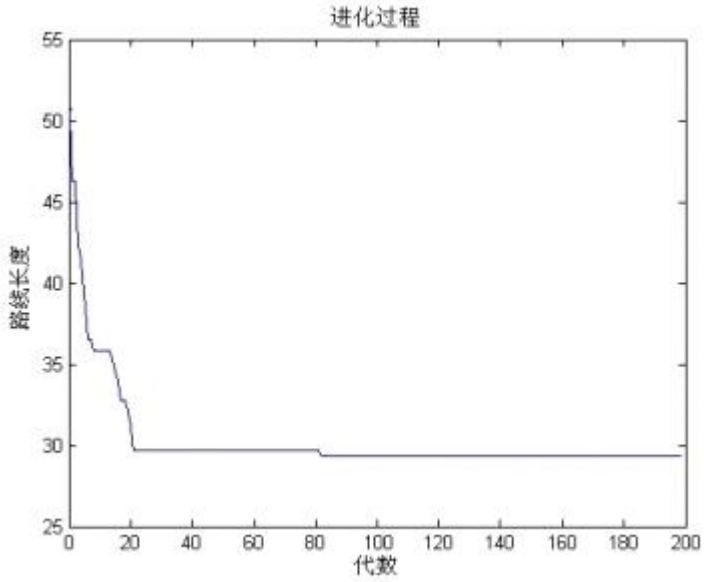


图 4.3 遗传算法进化过程图

4、主程序：

```
clear
```

```
clc
close all
load CityPosition1;%个城市坐标位置
NIND=100;            %种群大小
MAXGEN=200;
Pc=0.9;              %交叉概率
Pm=0.05;             %变异概率
GGAP=0.9;            %代沟(Generation gap)
D=Distanse(X);   %生成距离矩阵
N=size(D,1);       %(34*34)
%% 初始化种群
Chrom=InitPop(NIND,N);
%% 在二维图上画出所有坐标点
% figure
% plot(X(:,1),X(:,2),'o');
%% 画出随机解的路线图
DrawPath(Chrom(1,:),X)
pause(0.0001)
%% 输出随机解的路线和总距离
disp('初始种群中的一个随机值:')
OutputPath(Chrom(1,:));
Rlength=PathLength(D,Chrom(1,:));
disp(['总距离：',num2str(Rlength)]);
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
%% 优化
gen=0;
figure;
hold on;box on
xlim([0,MAXGEN])
title('优化过程')
xlabel('代数')
ylabel('最优值')
ObjV=PathLength(D,Chrom);   %计算路线长度
preObjV=min(ObjV);
while gen<MAXGEN
    %% 计算适应度
    ObjV=PathLength(D,Chrom);   %计算路线长度
    % fprintf('%d     %1.10f\n',gen,min(ObjV))
    line([gen-1,gen],[preObjV,min(ObjV)]);pause(0.0001)
    preObjV=min(ObjV);
    FitnV=Fitness(ObjV);
    %% 选择
    SelCh=Select(Chrom,FitnV,GGAP);
    %% 交叉操作
```

```
    SelCh=Recombin(SelCh,Pc);
    %% 变异
    SelCh=Mutate(SelCh,Pm);
    %% 逆转操作
    SelCh=Reverse(SelCh,D);
    %% 重插入子代的新种群
    Chrom=Reins(Chrom,SelCh,ObjV);
    %% 更新迭代次数
    gen=gen+1 ;
end
%% 画出最优解的路线图
ObjV=PathLength(D,Chrom);  %计算路线长度
[minObjV,minInd]=min(ObjV);
DrawPath(Chrom(minInd(1),:),X)
%% 输出最优解的路线和总距离
disp('最优解:')
p=OutputPath(Chrom(minInd(1),:));
disp(['总距离：',num2str(ObjV(minInd(1)))]);
disp('-----------------------------------------------------------')
```

第 5 章　基于遗传算法的 LQR 控制器优化设计

1、案例背景

LQR 控制在工程中得到了广泛的应用，对于 LQR 最优控制，其最优性完全取决于加权矩阵的选择，然而该加权矩阵如何选择并没有解析方法，只能定性地去选择矩阵参数，所以这样的"最优"控制事实上完全是认为的。如果选择不当，虽然可以求出最优解，但这样的"最优解"没有任何意义。另一方面，加权矩阵的选择依赖于设计者的经验，需要设计者根据系统输出逐步调整加权矩阵，直到获得满意的输出响应量为止，这样不仅费时，而且无法保证获得最优的权重矩阵，因此获得的最优控制反馈系数不能保证使系统达到最优。遗传算法（Genetic Algorithm，GA）是模仿自然界生物进化机制发展起来的全局搜索优化方法，它在迭代过程中使用适者生存的原则，采用交叉、变异等操作使得种群朝着最优的方向进化，最终获得最优解。鉴于 LQR 控制方法权重矩阵确定困难的问题，本案例以汽车主动悬架作为被控对象，将遗传算法应用于 LQR 控制器的设计中，利用遗传算法的全局搜索能力，以主动悬架的性能指标作为目标函数对加权矩阵进行优化设计，以提高 LQR 的设计效率和性能。

2、案例目录：

第 5 章　基于遗传算法的 LQR 控制器优化设计
5.1　案例背景
　5.1.1 LQR 控制
　5.1.2　基于遗传算法设计 LQR 控制器
5.2　模型建立
　5.2.1　主动悬架及其 LQR 控制器
　5.2.2　基于遗传算法的主动悬架 LQR 控制器优化设计
5.3　模型及算法实现
　5.3.1　模型实现
　5.3.2　遗传算法实现
　5.3.3　结果分析
5.4　参考文献

3、案例实例及结果：

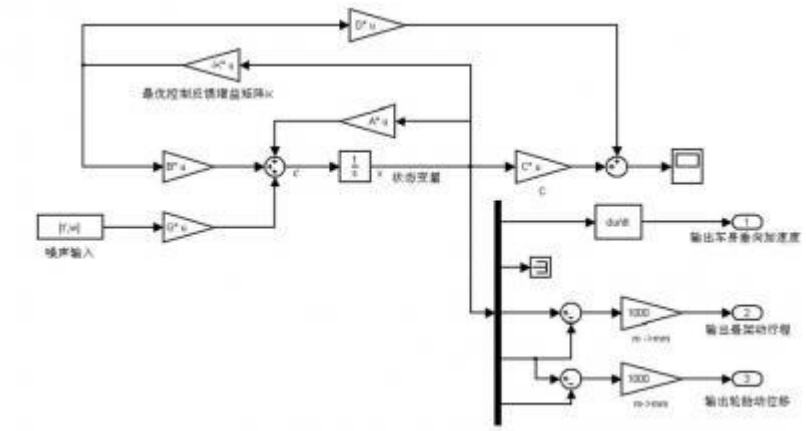


图 5-3　主动悬架 LQR 控制模型的 Simulink 实现

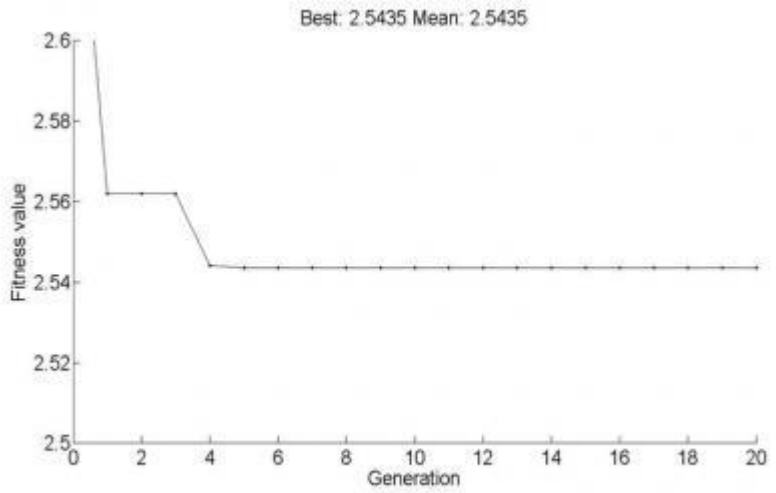


图 5-5　遗传算法最优个体适应度函数值变化情况的局部放大图

表 5-2　遗传算法优化 LQR 控制器的主动悬架与被动悬架的对比

性能指标	被动悬架	本文	改善
车身加速度 (m/s^2)	1.7816	1.7247	3.19%
悬架动行程 (mm)	17.1284	11.9624	30.16%
轮胎动位移 (mm)	6.2526	5.4839	12.29%

4、主程序：

```
clear
clc
fitnessfcn = @GA_LQR;        % 适应度函数句柄
nvars=3;                     % 个体变量数目
LB = [0.1 0.1 0.1];          % 下限
UB = [1e6 1e6 1e6];          % 上限
options=gaoptimset('PopulationSize',100,'PopInitRange',[LB;UB],'EliteCount',10,'CrossoverFraction',0.4,'Generations',20,'StallGenLimit',20,'TolFun',1e-100,'PlotFcns',{@gaplotbestf,@gaplotbestindiv});    % 算法参数设置
[x_best,fval]=ga(fitnessfcn,nvars, [],[],[],[],LB,UB,[],options);                % 运行遗传算法
```

第 6 章　遗传算法工具箱详解及应用

1、案例背景

MATLAB 自带的遗传算法与直接搜索工具箱(Genetic Algorithm and Direct Search Toolbox, GADST)，可以较好地解决与遗传算法相关的各种问题。GADST 可以通过 GUI 界面调用，也可以通过命令行方式调用，使用简单方便。本案例将对 GADST 函数库的遗传算法部分进行详细的代码分析和讲解，并通过求解非线性方程组介绍 GADST 的使用方法。

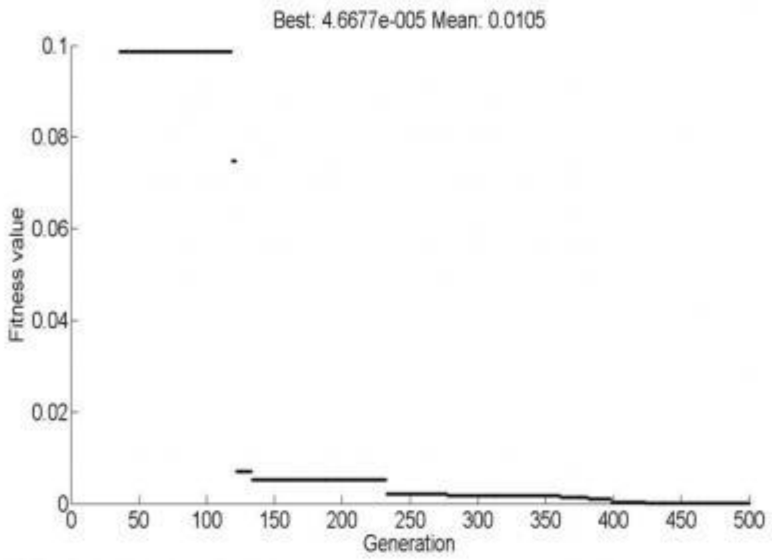
2、案例目录：

3、案例实例及结果：

作为案例，这里将使用遗传算法与直接搜索工具箱(GADST)求解一个非线性方程组。求解以下非线性方程组：

$$\begin{cases} 4x_1^3 + 4x_1x_2 + 2x_2^2 - 42x_1 - 14 = 0 \\ 4x_2^3 + 4x_1x_2 + 2x_1^2 - 26x_1 - 22 = 0 \end{cases}$$

优化过程如下：



得到的最优解为

[x1,x2] = [-0.247800834353742, 1.62131572868496]

4、主程序：

```
clear
clc
fitnessfcn = @GA_demo;           % 适应度函数句柄
nvars = 2;                                              % 个体所含的变量数目
options                                                                        =
gaoptimset('PopulationSize',100,'EliteCount',10,'CrossoverFraction',0.75,'Generations',500,'StallG
enLimit',500,'TolFun',1e-100,'PlotFcns',{@gaplotbestf,@gaplotbestindiv});                    % 参数
设置
[x_best,fval] =ga(fitnessfcn,nvars,[],[],[],[],[],[],[],options);% 调用 ga 函数
```

第 7 章　**多种群遗传算法的函数优化算法**

1、案例背景

针对遗传算法所存在的问题，一种多种群遗传算法结构模型（Multiple Population GA，简称 MPGA）可以用来取代常规的标准计算模型（SGA）。

MPGA 在 SGA 的基础上主要引入了以下几个概念：

（1）突破 SGA 仅靠单个群体进行遗传进化的框架，引入多个种群同时进行优化搜索；不同的种群赋以不同的控制参数，实现不同的搜索目的。

（2）各个种群之间通过移民算子进行联系，实现多种群的协同进化；最优解的获取是多个种群协同进化的综合结果。

（3）通过人工选择算子保存各种群每个进化代中的最优个体，并作为判断算法收敛的依据。

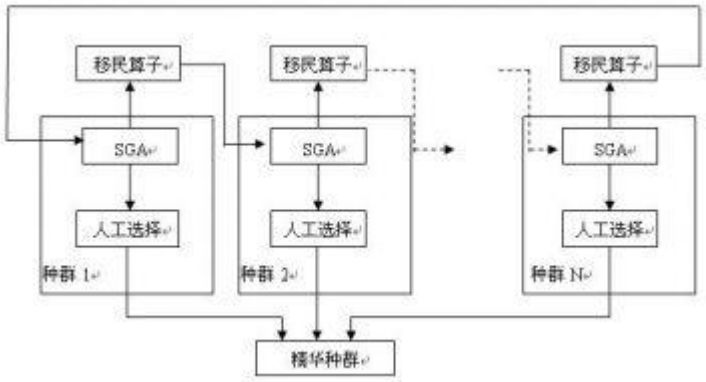


图 7-1 MPGA 的算法结构示意图

复杂二元函数求最值：

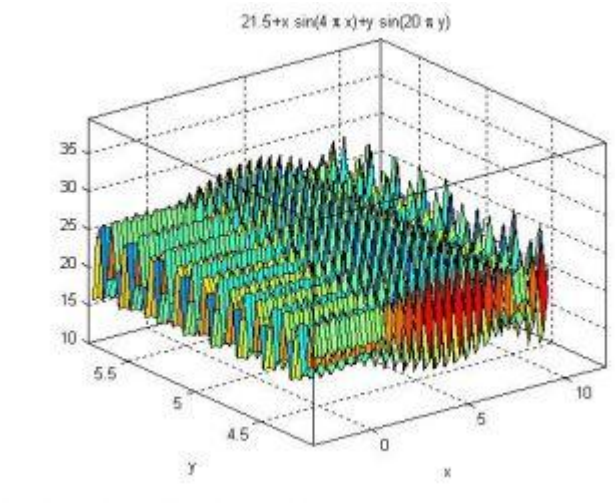


图 7-2 二元函数图像

2、案例目录：

3、主程序：

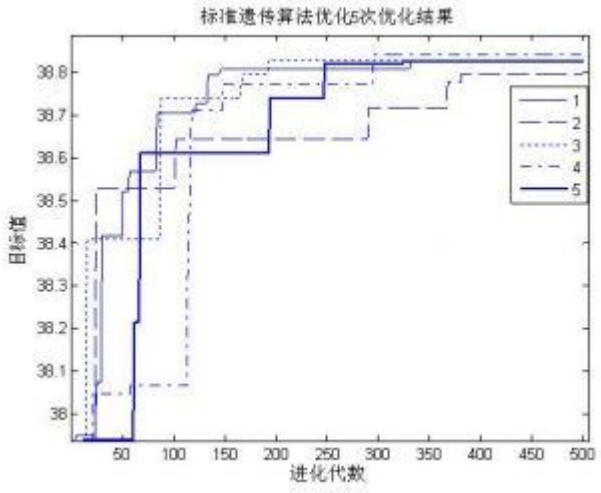
```
%% 多种群遗传算法
clear;
clc
close all
NIND=40; %个体数目
```

```
NVAR=2; %变量的维数
PRECI=20; %变量的二进制位数
GGAP=0.9; %代沟
MP=10; %种群数目
FieldD=[rep(PRECI,[1,NVAR]);[-3,4.1;12.1,5.8];rep([1;0;1;1],[1,NVAR])]; %译码矩阵
for i=1:MP
Chrom{i}=crtbp(NIND, NVAR*PRECI); %创建初始种群
end
pc=0.7+(0.9-0.7)*rand(MP,1); %在【0.7,0.9】范围 i 内随机产生交叉概率
pm=0.001+(0.05-0.001)*rand(MP,1); %在【0.001,0.05】范围内随机产生变异概率
gen=0; %初始遗传代数
gen0=0; %初始保持代数
MAXGEN=10; %最优个体最少保持代数
maxY=0; %最优值
for i=1:MP
ObjV{i}=ObjectFunction(bs2rv(Chrom{i}, FieldD));%计算各初始种群个体的目标函数值
end
MaxObjV=zeros(MP,1); %记录精华种群
MaxChrom=zeros(MP,PRECI*NVAR); %记录精华种群的编码
while gen0<=MAXGEN
gen=gen+1; %遗传代数加 1
for i=1:MP
FitnV{i}=ranking(-ObjV{i}); % 各种群的适应度
SelCh{i}=select('sus', Chrom{i}, FitnV{i},GGAP); % 选择操作
SelCh{i}=recombin('xovsp',SelCh{i}, pc(i)); % 交叉操作
SelCh{i}=mut(SelCh{i},pm(i)); % 变异操作
ObjVSel=ObjectFunction(bs2rv(SelCh{i}, FieldD)); % 计算子代目标函数值
[Chrom{i},ObjV{i}]=reins(Chrom{i},SelCh{i},1,1,ObjV{i},ObjVSel); %重插入操作
end
[Chrom,ObjV]=immigrant(Chrom,ObjV); % 移民操作
[MaxObjV,MaxChrom]=EliteInduvidual(Chrom,ObjV,MaxObjV,MaxChrom); % 人工选择精华种群
YY(gen)=max(MaxObjV); %找出精华种群中最优的个体
if YY(gen)>maxY %判断当前优化值是否与前一次优化值相同
maxY=YY(gen); %更新最优值
gen0=0;
else
gen0=gen0+1; %最优值保持次数加 1
end
end
%% 进化过程图
plot(1:gen,YY)
xlabel('进化代数')
ylabel('最优解变化')
title('进化过程')
```

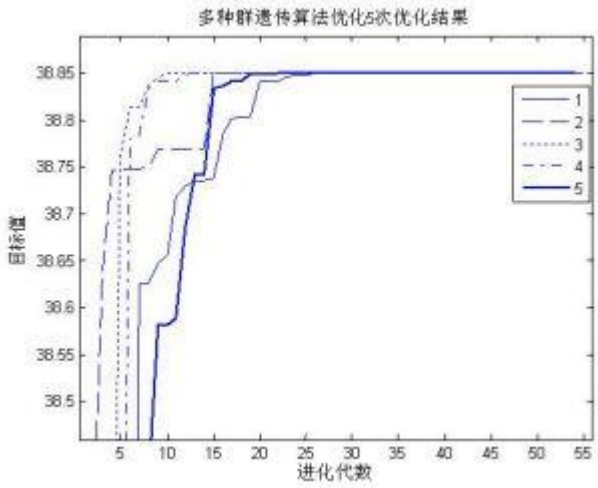
```
xlim([1,gen])
%% 输出最优解
[Y,I]=max(MaxObjV); %找出精华种群中最优的个体
X=(bs2rv(MaxChrom(I,:), FieldD)); %最优个体的解码解
disp(['最优值为：',num2str(Y)])
disp(['对应的自变量取值：',num2str(X)])
```

4、运行结果：

标准遗传算法运行 5 次得到的结果图：



多种群遗传算法运行 5 次得到的结果图：



第 8 章　**基于量子遗传算法的函数寻优算法**

1、案例背景

量子遗传算法就是基于量子计算原理的一种遗传算法。将量子的态矢量表达引入了遗传编码，利用量子逻辑门实现染色体的演化，实现了比常规遗传算法更好的效果。
量子遗传算法建立在量子的态矢量表示的基础之上，将量子比特的几率幅表示应用于染色体的编码，使得一条染色体可以表达多个态的叠加，并利用量子逻辑门实现染色体的更新操作，从而实现了目标的优化求解。
复杂二元函数求最值：

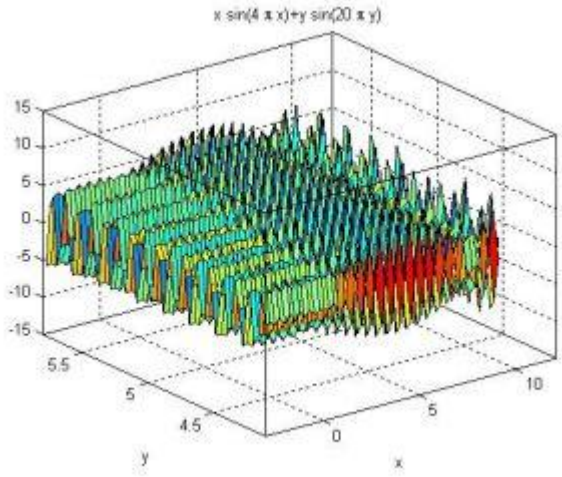


图 8-1 二元函数图像

2、案例目录：

3、主程序：

```
clc;
clear all;
close all;
%----------------参数设置-----------------------
MAXGEN=200; %  最大遗传代数
sizepop=40; %  种群大小
lenchrom=[20 20]; %  每个变量的二进制长度
trace=zeros(1,MAXGEN);
%-------------------------------------------------------------------------
%  最佳个体  记录其适应度值、十进制值、二进制编码、量子比特编码
best=struct('fitness',0,'X',[],'binary',[],'chrom',[]);
%%  初始化种群
chrom=InitPop(sizepop*2,sum(lenchrom));
%%  对种群实施一次测量  得到二进制编码
binary=collapse(chrom);
%%  求种群个体的适应度值，和对应的十进制值
[fitness,X]=FitnessFunction(binary,lenchrom); %  使用目标函数计算适应度
%%  记录最佳个体到 best
[best.fitness bestindex]=max(fitness); %  找出最大值
best.binary=binary(bestindex,:);
best.chrom=chrom([2*bestindex-1:2*bestindex],:);
best.X=X(bestindex,:);
trace(1)=best.fitness;
fprintf('%d\n',1)
%%  进化
for gen=2:MAXGEN
fprintf('%d\n',gen) %提示进化代数
```

```
%% 对种群实施一次测量
binary=collapse(chrom);
%% 计算适应度
[fitness,X]=FitnessFunction(binary,lenchrom);
%% 量子旋转门
chrom=Qgate(chrom,fitness,best,binary);
[newbestfitness,newbestindex]=max(fitness); % 找到最佳值
% 记录最佳个体到 best
if newbestfitness>best.fitness
best.fitness=newbestfitness;
best.binary=binary(newbestindex,:);
best.chrom=chrom([2*newbestindex-1:2*newbestindex],:);
best.X=X(newbestindex,:);
end
trace(gen)=best.fitness;
end

%% 画进化曲线
plot(1:MAXGEN,trace);
title('进化过程');
xlabel('进化代数');
ylabel('每代的最佳适应度');

%% 显示优化结果
disp(['最优解 X： ',num2str(best.X)])
disp(['最大值 Y:',num2str(best.fitness)]);
```

4、运行结果：

最优解 X：11.6255　　5.72504
最大值 Y:17.3503
量子遗传算法优化 200 代得到的进化过程图如图 8-3 所示。

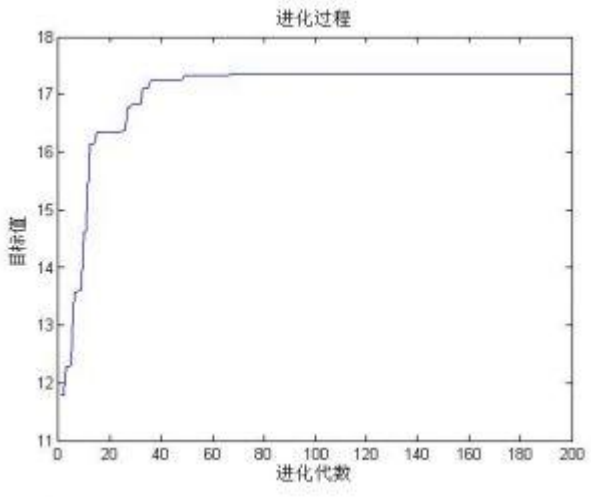


图 8-3 QGA 进化过程图

第 9 章　基于遗传算法的多目标优化算法

1、案例背景

目前的多目标优化算法有很多，Kalyanmoy Deb 的 NSGA-II（Nondominated Sorting Genetic Algorithm II，带精英策略的快速非支配排序遗传算法）无疑是其中应用最为广泛也是最为成功的一种。MATLAB 自带的 gamultiobj 函数所采用的算法，就是基于 NSGA-II 改进的一种多目标优化算法（a variant of NSGA-II）。gamultiobj 函数的出现，为在 MATLAB 平台下解决多目标优化问题提供了良好的途径。gamultiobj 函数包含在遗传算法与直接搜索工具箱(Genetic Algorithm and Direct Search Toolbox, GADST)中，这里我们称 gamultiobj 函数为基于遗传算法的多目标优化函数，相应的算法为基于遗传算法的多目标优化算法。本案例将以 gamultiobj 函数为基础，对基于遗传算法的多目标优化算法进行讲解，并详细分析其代码，同时通过一个简单的案例介绍 gamultiobj 函数的使用。

2、案例目录：

3、案例实例及结果：

作为案例，这里将使用 MATLAB 自带的基于遗传算法的多目标优化函数 gamultiobj 求解一个简单的多目标优化问题。待优化的多目标问题表述如下：

$$\min f_1(x_1,x_2) = x_1^4 - 10x_1^2 + x_1x_2 + x_2^4 - x_1^2x_2^2$$

$$\min f_2(x_1,x_2) = x_2^4 - x_1^2x_2^2 + x_1^4 + x_1x_2$$

$$subject\ to \begin{cases} -5 \le x_1 \le 5 \\ -5 \le x_2 \le 5 \end{cases}$$

可以看到，在基于遗传算法的多目标优化算法的运行过程中，自动绘制了第一前端中个体的分布情况，且分布随着算法进化一代而更新一次。当迭代停止后，得到如图 9.5 所示的第一前端个体分布图。同时，Worksapce 中返回了 gamultiobj 函数得到的 Pareto 解集 x 及与 x 对应的目标函数值，如表 9.2 所示。

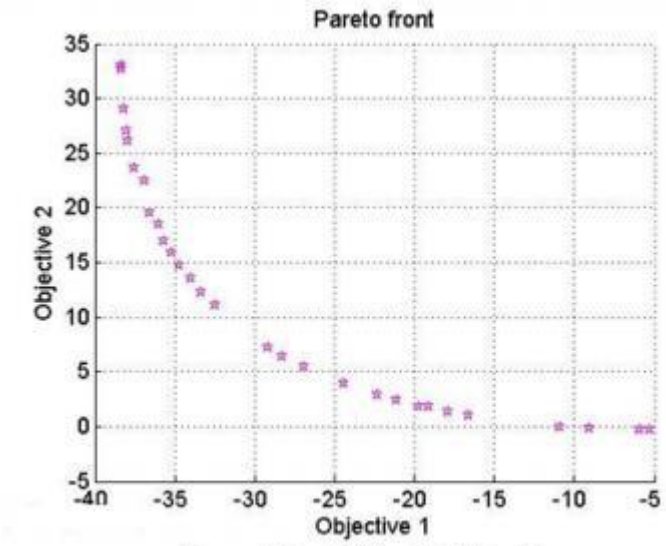


图9.5 第一前端个体分布图

4、主程序：

```
clear
```

```
clc
fitnessfcn = @my_first_multi;      % Function handle to the fitness function
nvars = 2;                          % Number of decision variables
lb = [-5,-5];                       % Lower bound
ub = [5,5];                         % Upper bound
A = []; b = [];                     % No linear inequality constraints
Aeq = []; beq = [];                 % No linear equality constraints
options                                                                                 =
gaoptimset('ParetoFraction',0.3,'PopulationSize',100,'Generations',200,'StallGenLimit',200,'TolFu
n',1e-100,'PlotFcns',@gaplotpareto);
  [x,fval] = gamultiobj(fitnessfcn,nvars, A,b,Aeq,beq,lb,ub,options);
```

第 10 章　**基于粒子群算法的多目标搜索算法**

1、案例背景

在实际工程优化问题中，多数问题是多目标优化问题。相对于单目标优化问题，多目标优化问题的显著特点是优化各个目标使其同时达到综合的最优值。然而，由于多目标优化问题的各个目标之间往往是相互冲突的，在满足其中一个目标最优的同时，其他的目标往往可能会受其影响而变得很差。因此，一般适用于单目标问题的方法难以用于多目标问题的求解。
多目标优化问题很早就引起了人们的重视，现已经发展出多种求解多目标优化问题的方法。多目标优化问题求解中的最重要概念是非劣解和非劣解集，两者的定义如下。
非劣解（noninferior solution）：在多目标优化问题的可行域中存在一个问题解，若不存在另一个可行解，使得一个解中的目标全部劣于该解，则该解称为多目标优化问题的非劣解。所有非劣解的集合叫做非劣解集（noninferior Set）。

2、案例目录

本案例的目录为：

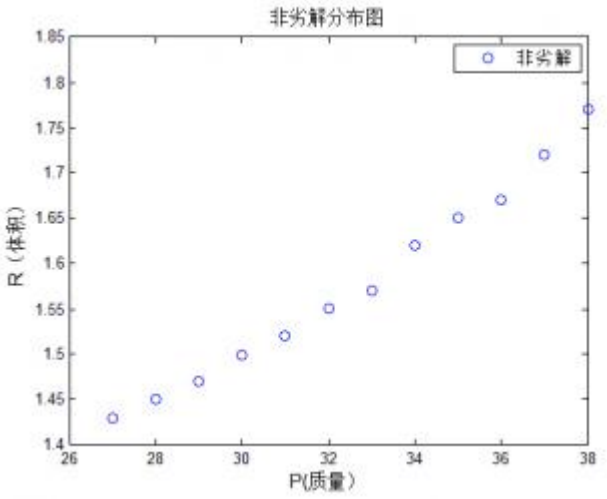
3、主程序

```
%% 循环迭代
for iter=1:MaxIt
% 权值更新
w=wmax-(wmax-wmin)*iter/MaxIt;

%从非劣解中选择粒子作为全局最优解
s=size(fljx,1)
index=randi(s,1,1);
gbest=fljx(index,:);

%% 群体更新
for i=1:xSize
%速度更新
v(i,:)=w*v(i,:)+c1*rand(1,1)*(xbest(i,:)-x(i,:))+c2*rand(1,1)*(gbest-x(i,:));
```

```
%位置更新
x(i,:)=x(i,:)+v(i,:);
x(i,:) = rem(x(i,:),objnum)/double(objnum);
index1=find(x(i,:)<=0);
if length(index1)~=0
x(i,index1)=rand(size(index1));
end
x(i,:)=ceil(4*x(i,:));
end

%% 更新粒子历史最佳
for i=1:xSize
%现在的支配原有的，替代原有的
if ((px(i)<ppx(i)) && (rx(i)<rrx(i))) ||((abs(px(i)-ppx(i))<tol)...
&& (rx(i)<rrx(i)))||((px(i)<ppx(i)) && (abs(rx(i)-rrx(i))<tol)) || (cx(i)>weight)
xbest(i,:)=x(i,:);%没有记录目标值
pxbest(i)=ppx(i);rxbest(i)=rrx(i);cxbest(i)=ccx(i);
end

%% 更新非劣解集合
px=ppx;
rx=rrx;
cx=ccx;
%更新升级非劣解集合
s=size(flj,1);%目前非劣解集合中元素个数

%先将非劣解集合和 xbest 合并
pppx=zeros(1,s+xSize);
rrrx=zeros(1,s+xSize);
cccx=zeros(1,s+xSize);
pppx(1:xSize)=pxbest;pppx(xSize+1:end)=flj(:,1)';
rrrx(1:xSize)=rxbest;rrrx(xSize+1:end)=flj(:,2)';
cccx(1:xSize)=cxbest;cccx(xSize+1:end)=flj(:,3)';
xxbest=zeros(s+xSize,Dim);
xxbest(1:xSize,:)=xbest;
xxbest(xSize+1:end,:)=fljx;


%筛选非劣解
flj=[];
fljx=[];
k=0;
tol=1e-7;
for i=1:xSize+s
```

```
flag=0;%没有被支配
%判断该点是否非劣
for j=1:xSize+s
if j~=i
if ((pppx(i)<pppx(j)) && (rrrx(i)<rrrx(j))) ||((abs(pppx(i)-pppx(j))<tol) ...
&& (rrrx(i)<rrrx(j)))||((pppx(i)<pppx(j)) && (abs(rrrx(i)-rrrx(j))<tol)) ...
|| (cccx(i)>weight) %有一次被支配
flag=1;
break;
end
end
end

%去掉重复粒子
repflag=0; %重复标志
k=1; %不同非劣解粒子数
flj2=[]; %存储不同非劣解
fljx2=[]; %存储不同非劣解粒子位置
flj2(k,:)=flj(1,:);
fljx2(k,:)=fljx(1,:);
for j=2:size(flj,1)
repflag=0; %重复标志
for i=1:size(flj2,1)
result=(fljx(j,:)==fljx2(i,:));
if length(find(result==1))==Dim
repflag=1;%有重复
end
end
%粒子不同，存储
if repflag==0
k=k+1;
flj2(k,:)=flj(j,:);
fljx2(k,:)=fljx(j,:);
end

end

%非劣解更新
flj=flj2;
fljx=fljx2;

end
```

4、运行结果



第 11 章　基于多层编码遗传算法的车间调度算法

1 案例背景

遗传算法具有较强的问题求解能力，能够解决非线性优化问题。对于遗传算法中的染色体表示问题中的一个潜在最优解，对于简单的问题来说，染色体可以方便的表达问题的潜在解，然而，对于较为复杂的优化问题，一个染色体难以准确表达问题的解。多层编码遗传算法，把个体编码分为多层，每层编码均表示不同的含义，多层编码共同完整表达了问题的解，从而用一个染色体准确表达出了复杂问题的解。多层编码遗传算法扩展了遗传算法的使用领域，使得遗传算法可以方便用于复杂问题的求解。

2、案例目录

3、主程序：

```
%% 基本参数
NIND=40; %个体数目
MAXGEN=50; %最大遗传代数
GGAP=0.9; %代沟
XOVR=0.8; %交叉率
MUTR=0.6; %变异率
gen=0; %代计数器
%PNumber 工件个数 MNumber 工序个数
[PNumber MNumber]=size(Jm);
trace=zeros(2, MAXGEN); %寻优结果的初始值
WNumber=PNumber*MNumber; %工序总个数

%% 初始化
Number=zeros(1,PNumber); % PNumber 工件个数
for i=1:PNumber
Number(i)=MNumber; %MNumber 工序个数
end

% 代码 2 层，第一层工序，第二层机器
Chrom=zeros(NIND,2*WNumber);
for j=1:NIND
WPNumberTemp=Number;
for i=1:WNumber

%随机产成工序
val=unidrnd(PNumber);
while WPNumberTemp(val)==0
val=unidrnd(PNumber);
end

%第一层代码表示工序
Chrom(j,i)= val;
WPNumberTemp(val)=WPNumberTemp(val)-1;

%第 2 层代码表示机器
```

```
Temp=Jm{val,MNumber-WPNumberTemp(val)};
SizeTemp=length(Temp);
%随机产成工序机器
Chrom(j,i+WNumber)= unidrnd(SizeTemp);

end
end

%计算目标函数值
[PVal ObjV P S]=cal(Chrom,JmNumber,T,Jm);

%% 循环寻找
while gen<MAXGEN

%分配适应度值
FitnV=ranking(ObjV);
%选择操作
SelCh=select('rws', Chrom, FitnV, GGAP);
%交叉操作
SelCh=across(SelCh,XOVR,Jm,T);
%变异操作
SelCh=aberranceJm(SelCh,MUTR,Jm,T);

%计算目标适应度值
[PVal ObjVSel P S]=cal(SelCh,JmNumber,T,Jm);
%重新插入新种群
[Chrom ObjV] =reins(Chrom, SelCh,1, 1, ObjV, ObjVSel);
%代计数器增加
gen=gen+1;

%保存最优值
trace(1, gen)=min(ObjV);
trace(2, gen)=mean(ObjV);

% 记录最佳值
if gen==1
Val1=PVal;
Val2=P;
MinVal=min(ObjV);%最小时间
STemp=S;
end
%记录 最小的工序
if MinVal> trace(1,gen)
Val1=PVal;
```

```
Val2=P;
MinVal=trace(1,gen);
STemp=S;
end

end
```

第 12 章　免疫优化算法在物流配送中心选址中的应用.

1、案例背景

12. 1. 1

物流中心选址问题

随着世界经济的快速发展以及现代科学技术的进步，物流业作为国民经济的一个新兴服务部门，正在全球范围内迅速发展。物流业的发展给社会的生产和管理、人们的生活和就业乃至政府的职能以及社会的法律制度等都带来巨大的影响，因此物流业被认为是国民经济发展的动脉和基础产业，被形象地喻为促进经济发展的“加速器”。

在物流系统的运作中，配送中心的任务就是根据各个用户的需求及时、准确和经济地配送商品货物。配送中心是连接供应商与客户的中间桥梁，其选址方式往往决定着物流的配送距离和配送模式，进而影响着物流系统的运作效率；另外物流中心的位置一旦被确定，其位置难以再改变。因此，研究物流配送中心的选址具有重要的理论和现实应用意义，一般说来，物流中心选址模型是非凸和非光滑的带有复杂约束的非线性规划模型，属 NP-hard 问题。

12.1.2 免疫算法的基本思想

生物免疫系统是一个高度进化的生物系统，它旨在区分外部有害抗原和自身组织，从而保持有机体的稳定。从计算的角度来看，生物免疫系统是一个高度并行、分布、自适应和自组织的系统，具有很强的学习、识别和记忆能力。

2、案例目录

3、主程序

```
% 免疫优化算法在物流配送中心选址中的应用
%% 免疫优化算法求解
clc
clear
% 算法基本参数
sizepop=50; % 种群规模
overbest=10; % 记忆库容量
MAXGEN=100; % 迭代次数
pcross=0.5; % 交叉概率
pmutation=0.4; % 变异概率
ps=0.95; % 多样性评价参数
length=6; % 配送中心数
M=sizepop+overbest;

% step1 识别抗原,将种群信息定义为一个结构体
individuals = struct('fitness',zeros(1,M),
'concentration',zeros(1,M),'excellence',zeros(1,M),'chrom',[]);
% step2 产生初始抗体群
individuals.chrom = popinit(M,length);
trace=[]; %记录每代最个体优适应度和平均适应度

% 迭代寻优
for iii=1:MAXGEN
iii
% step3 抗体群多样性评价
for i=1:M
individuals.fitness(i) = fitness(individuals.chrom(i,:)); % 抗体与抗原亲和度(适应度值）计算
individuals.concentration(i) = concentration(i,M,individuals); % 抗体浓度计算
end
% 综合亲和度和浓度评价抗体优秀程度，得出繁殖概率
individuals.excellence = excellence(individuals,M,ps);

% 记录当代最佳个体和种群平均适应度
[best,index] = min(individuals.fitness); % 找出最优适应度
bestchrom = individuals.chrom(index,:); % 找出最优个体
average = mean(individuals.fitness); % 计算平均适应度
trace = [trace;best,average]; % 记录
```

```
% step4 根据 excellence，形成父代群，更新记忆库（加入精英保留策略，可由 s 控制）
bestindividuals = bestselect(individuals,M,overbest); % 更新记忆库
individuals = bestselect(individuals,M,sizepop); % 形成父代群

% step6 选择，交叉，变异操作，再加入记忆库中抗体，产生新种群
individuals = Select(individuals,sizepop); % 选择
individuals.chrom = Cross(pcross,individuals.chrom,sizepop,length); % 交叉
individuals.chrom = Mutation(pmutation,individuals.chrom,sizepop,length); % 变异
individuals = incorporate(individuals,sizepop,bestindividuals,overbest); % 加入记忆库中抗体

end
```

第 13 章　粒子群优化算法的寻优算法

1、案例背景

粒子群优化算法（PSO,particle swarm optimization）是计算智能领域，除了蚁群算法，鱼群算法之外的一种群体智能的优化算法，该算法最早由 Kennedy 和 Eberhart 在 1995 年提出的。PSO 算法源于对鸟类捕食行为的研究，鸟类捕食时，找到食物最简单有效的策略就是搜寻当前距离食物最近的鸟的周围区域。PSO 算法是从这种生物种群行为特征中得到启发并用于求解优化问题的，算法中每个粒子都代表问题的一个潜在解，每个粒子对应一个由适应度函数决定的适应度值。粒子的速度决定了粒子移动的方向和距离，速度随自身及其他粒子的移动经验进行动态调整，从而实现个体在可解空间中的寻优。

PSO 算法首先在可行解空间中初始化一群粒子，每个粒子都代表极值优化问题的一个潜在最优解，用位置、速度和适应度值三项指标表示该粒子特征，适应度值由适应度函数计算得到，其值的好坏表示粒子的优劣。粒子在解空间中运动，通过跟踪个体极值 *Pbest* 和群体极值 *Gbest* 更新个体位置，个体极值 *Pbest* 是指个体所经历位置中计算得到的适应度值最优位置，群体极值 *Gbest* 是指种群中的所有粒子搜索到的适应度最优位置。粒子每更新一次位置，就计算一次适应度值，并且通过比较新粒子的适应度值和个体极值、群体极值的适应度值更新个体极值 *Pbest* 和群体极值 *Gbest* 位置。

2、案例目录

本案例的目录为：

3、主程序

```
%% 迭代寻优
for i=1:maxgen

for j=1:sizepop

%速度更新
V(j,:) = V(j,:) + c1*rand*(gbest(j,:) - pop(j,:)) + c2*rand*(zbest - pop(j,:));
V(j,find(V(j,:)>Vmax))=Vmax;
```

```
V(j,find(V(j,:)<Vmin))=Vmin;

%种群更新
pop(j,:)=pop(j,:)+V(j,:);
pop(j,find(pop(j,:)>popmax))=popmax;
pop(j,find(pop(j,:)<popmin))=popmin;

%适应度值
fitness(j)=fun(pop(j,:));

end

for j=1:sizepop

%个体最优更新
if fitness(j) > fitnessgbest(j)
gbest(j,:) = pop(j,:);
fitnessgbest(j) = fitness(j);
end

%群体最优更新
if fitness(j) > fitnesszbest
zbest = pop(j,:);
fitnesszbest = fitness(j);
end
end
yy(i)=fitnesszbest;

end
```

第 14 章　基于粒子群算法的 PID 控制器优化设计

1、案例背景

PID 控制器的性能取决于 Kp、Ki、Kd 这 3 个参数是否合理，因此，优化 PID 控制器参数具

有重要意义。目前，PID 控制器参数主要是人工调整，这种方法不仅费时，而且不能保证获得最佳的性能。PSO 已经广泛应用于函数优化、神经网络训练、模式分类、模糊系统控制以及其它应用领域，本案例将使用 PSO 进行 PID 控制器参数的优化设计。

2、案例目录：

3、案例实例及结果：

PID 控制器的系统结构图如图 14-1 所示。

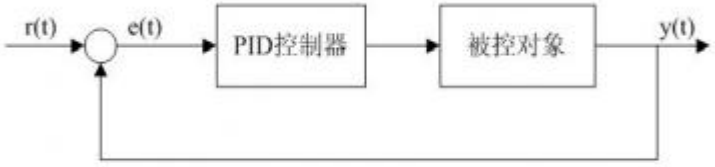
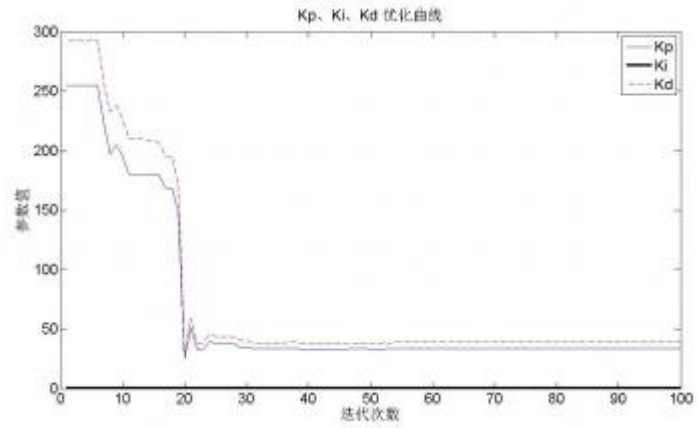


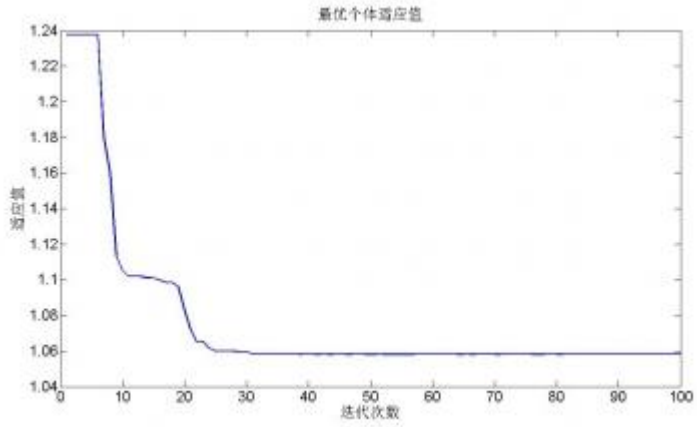
图 14-1 PID 控制器系统结构图

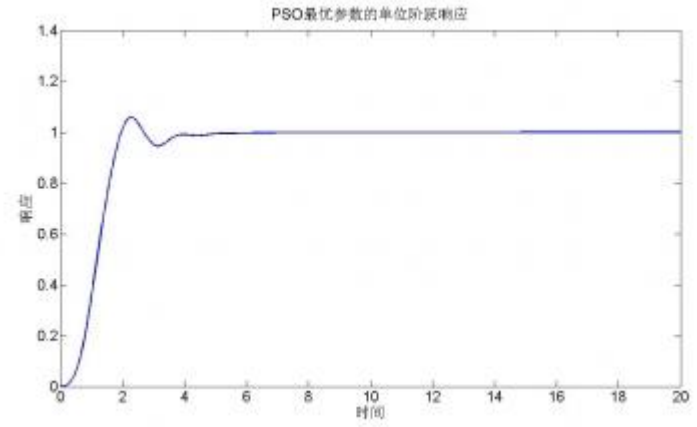
选取的被控对象为以下不稳定系统：

$$G(s)=\frac{s+2}{s^4+8s^3+4s^2-s+0.4}$$

运行代码，得到优化过程如图 14-4 和图 14-5 所示，前者为 PID 控制器 3 个参数 Kp、Ki、Kd 的变化曲线，后者为性能指标 ITAE 的变化曲线。得到的最优控制器参数及性能指标为 Kp= 33.6469，Ki= 0.1662，Kd= 38.8063，
ITAE= 1.0580，
将以上参数代回如图 14-2 所示的模型，得到的单位阶跃响应曲线如图 14-6 所示。

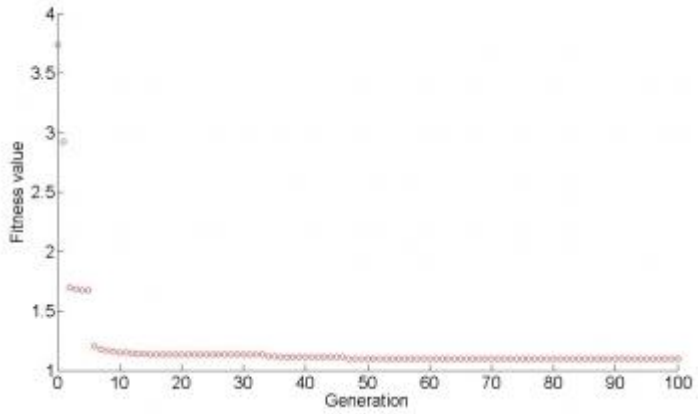


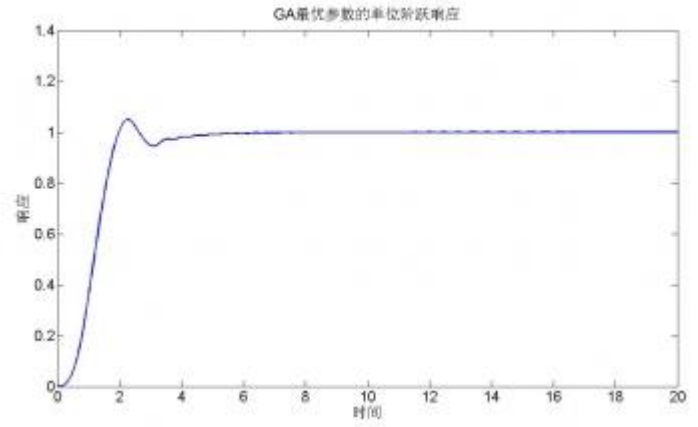




本案例使用粒子群算法优化 PID 控制器参数，事实上，其它的优化算法，比如遗传算法、模拟退火算法等，也可以用于 PID 控制器的参数优化，这里将使用遗传算法（Genetic Algorithm, GA）对 PID 控制器进行参数优化。

得到的进化过程曲线、最优参数对应的单位阶跃响应曲线分别如图 14-7、图 14-8 所示。





4、主程序：

```
%% 清空环境
clear
clc

%% 参数设置
w = 0.6;          % 惯性因子
c1 = 2;           % 加速常数
c2 = 2;           % 加速常数

Dim = 3;                % 维数
SwarmSize = 100;        % 粒子群规模
ObjFun = @PSO_PID;    % 待优化函数句柄

MaxIter = 100;          % 最大迭代次数
MinFit = 0.1;           % 最小适应值

Vmax = 1;
Vmin = -1;
Ub = [300 300 300];
Lb = [0 0 0];
```

```
%% 粒子群初始化
    Range = ones(SwarmSize,1)*(Ub-Lb);
    Swarm = rand(SwarmSize,Dim).*Range + ones(SwarmSize,1)*Lb        % 初始化粒子群
    VStep = rand(SwarmSize,Dim)*(Vmax-Vmin) + Vmin                   % 初始化速度
    fSwarm = zeros(SwarmSize,1);
for i=1:SwarmSize
    fSwarm(i,:) = feval(ObjFun,Swarm(i,:));                          % 粒子群的适应值
end

%% 个体极值和群体极值
[bestf bestindex]=min(fSwarm);
zbest=Swarm(bestindex,:);    % 全局最佳
gbest=Swarm;                 % 个体最佳
fgbest=fSwarm;               % 个体最佳适应值
fzbest=bestf;                % 全局最佳适应值

%% 迭代寻优
iter = 0;
y_fitness = zeros(1,MaxIter);    % 预先产生 4 个空矩阵
K_p = zeros(1,MaxIter);
K_i = zeros(1,MaxIter);
K_d = zeros(1,MaxIter);
while( (iter < MaxIter) && (fzbest > MinFit) )
    for j=1:SwarmSize
        % 速度更新
        VStep(j,:) = w*VStep(j,:) + c1*rand*(gbest(j,:) - Swarm(j,:)) + c2*rand*(zbest -
Swarm(j,:));
        if VStep(j,:)>Vmax, VStep(j,:)=Vmax; end
        if VStep(j,:)<Vmin, VStep(j,:)=Vmin; end
        % 位置更新
        Swarm(j,:)=Swarm(j,:)+VStep(j,:);
        for k=1:Dim
            if Swarm(j,k)>Ub(k), Swarm(j,k)=Ub(k); end
            if Swarm(j,k)<Lb(k), Swarm(j,k)=Lb(k); end
        end
        % 适应值
        fSwarm(j,:) = feval(ObjFun,Swarm(j,:));
        % 个体最优更新
        if fSwarm(j) < fgbest(j)
            gbest(j,:) = Swarm(j,:);
            fgbest(j) = fSwarm(j);
        end
        % 群体最优更新
```

```
            if fSwarm(j) < fzbest
                zbest = Swarm(j,:);
                fzbest = fSwarm(j);
            end
        end
        iter = iter+1;                              % 迭代次数更新
        y_fitness(1,iter) = fzbest;                 % 为绘图做准备
        K_p(1,iter) = zbest(1);
        K_i(1,iter) = zbest(2);
        K_d(1,iter) = zbest(3);
end
%% 绘图输出
figure(1)           % 绘制性能指标 ITAE 的变化曲线
plot(y_fitness,'LineWidth',2)
title('最优个体适应值','fontsize',18);
xlabel('迭代次数','fontsize',18);ylabel('适应值','fontsize',18);
set(gca,'Fontsize',18);

figure(2)           % 绘制 PID 控制器参数变化曲线
plot(K_p)
hold on
plot(K_i,'k','LineWidth',3)
plot(K_d,'--r')
title('Kp、Ki、Kd 优化曲线','fontsize',18);
xlabel('迭代次数','fontsize',18);ylabel('参数值','fontsize',18);
set(gca,'Fontsize',18);
legend('Kp','Ki','Kd',1);
```

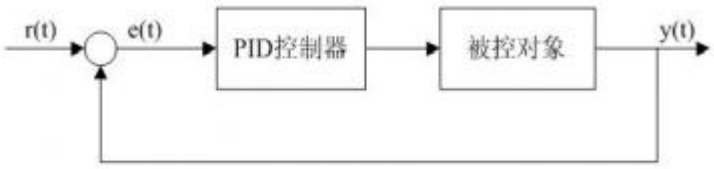


图 14-1 PID 控制器系统结构图

1、案例背景

粒子群算法（Particle Swarm Optimization,PSO）是 Kennedy 博士和 Eberhart 博士在 1995 年提出的一种基于群体智能的优化算法，它首先初始化一群随机粒子，然后通过迭代寻找问题的最优解，在每一次迭代过程中，粒子通过个体极值和群体极值更新自身的速度和位置。

标准粒子群算法通过追随个体极值和群体极值来完成极值寻优的，虽然操作简单，且能够快速收敛，但是随着迭代次数的不断增加，在种群收敛集中的同时，各粒子也越来越相似，可能在局部解周边无法跳出。混合粒子群算法摒弃了传统粒子群算法中的通过跟踪极值来更新粒子位置的方法，而是引入了遗传算法中的交叉和变异操作，通过粒子同个体极值和群体极值的交叉以及粒子自身变异的方式来搜索最优解。

2、案例目录

本案例的目录为：

3、主程序

```
%% 循环寻找最优路径
for N=1:nMax
N
%计算适应度值
indiFit=fitness(individual,cityCoor,cityDist);

%更新当前最优和历史最优
for i=1:indiNumber
if indiFit(i)<recordPbest(i)
recordPbest(i)=indiFit(i);
tourPbest(i,:)=individual(i,:);
end
```

```
if indiFit(i)<recordGbest
recordGbest=indiFit(i);
tourGbest=individual(i,:);
end
end

[value,index]=min(recordPbest);
recordGbest(N)=recordPbest(index);

%% 变异操作
c1=round(rand*(n-1))+1; %产生变异位
c2=round(rand*(n-1))+1; %产生变异位
while c1==c2
c1=round(rand*(n-2))+1;
c2=round(rand*(n-2))+1;
end
temp=xnew1(i,c1);
xnew1(i,c1)=xnew1(i,c2);
xnew1(i,c2)=temp;

%新路径长度变短则接受
dist=0;
for j=1:n-1
dist=dist+cityDist(xnew1(i,j),xnew1(i,j+1));
end
dist=dist+cityDist(xnew1(i,1),xnew1(i,n));
if indiFit(i)>dist
individual(i,:)=xnew1(i,:);
end
end

[value,index]=min(indiFit);
L_best(N)=indiFit(index);
tourGbest=individual(index,:);

end
```

第 16 章　基于动态粒子群算法的动态环境寻优算

1、案例背景

为了跟踪动态极值，需要对基本的 PSO 算法进行两方面改进，第一是引入探测机制，使种群或粒子获得感知外部环境变化的能力；第二是引入响应机制，在探测到环境的变化后，采取某种响应方式对种群进行更新，以适应动态环境。基于敏感粒子的动态粒子群算法是一种典型的动态粒子群算法，它在算法初始化时随机选择一个或多个位置，称为敏感粒子，每次迭代中计算敏感粒子适应度值，当发现适应度值变化时，认为环境已发生变化。响应的方式是按照一定比例重新初始化粒子位置和粒子速度。

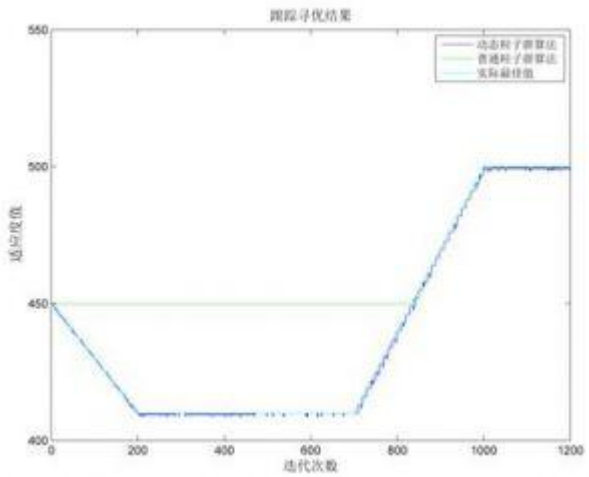
2、案例目录

3、主程序

```
%% 循环寻找最优点
for k = 1:1200
k
% 新数字地图
h = DF1function(X1,Y1,H1,X2(k),Y2(k),H2(k));

% 敏感粒子变化
for i=1:5*n
fitnessTest(i)=h(popTest(i,1),popTest(i,2));
end
oFitness=sum(fitnessTest);

% 变化超过一定范围,重新初始化
if abs(oFitness - nFitness)>1
index=randperm(20);
pop(index(1:10),:)=unidrnd(501,[10,2]);
V(index(1:10),:)=unidrnd(100,[10,2])-50;
```

```
end

% 粒子搜索
for i=1:Tmax

for j=1:n
% 速度更新
V(j,:)=V(j,:)+floor(rand*(popgbest(j,:)-pop(j,:)))+floor(rand*(popzbest - pop(j,:)));
index1=find(V(j,:)>Vmax);
V(j,index1)=Vmax;
index2=find(V(j,:)<Vmin);
V(j,index2)=Vmin;

% 个体更新
pop(j,:)=pop(j,:)+V(j,:);
index1=find(pop(j,:)>popMax);
pop(j,index1)=popMax;
index2=find(pop(j,:)<popMin);
pop(j,index2)=popMin;

% 新适应度值
fitness(j)=h(pop(j,1),pop(j,2));

% 个体极值更新
if fitness(j) > fitnessgbest(j)
popgbest(j,:) = pop(j,:);
fitnessgbest(j) = fitness(j);
end

% 群体极值更新
if fitness(j) > fitnesszbest
popzbest= pop(j,:);
fitnesszbest = fitness(j);
end

end
end

fitnessRecord(k)=fitnesszbest;
fitnesszbest=0;
fitnessgbest=zeros(1,20);
end
```

4、运行结果



第 17 章　**基于 PSO 工具箱的函数优化算法**

1、案例背景

粒子群算法具有操作简单，算法搜索效率较高等优点，该算法对待优化函数没有连续，可微的要求，算法通用性较强，对多变量、非线性、不连续及不可微的问题求解有较大的优势。PSO 工具箱由美国北卡罗来纳州立大学航天航空与机械系教授 Brian Birge 开发，该工具箱将 PSO 算法的核心部分封装起来，提供给用户的为算法的可调参数，用户只需要定义需要优化的函数，并设置好函数自变量的取值范围、每步迭代允许的最大变化量等，即可进行优化。

2、案例目录

3、主程序

```
Max_V = 0.2*(range(:,2)-range(:,1)); %最大速度取变化范围的 10%~20%
n=2; %待优化函数的维数，此例子中仅 x、y 两个自变量，故为 2

PSOparams= [25 2000 24 2 2 0.9 0.4 1500 1e-25 250 NaN 0 0];

pso_Trelea_vectorized('Rosenbrock',n,Max_V,range,0,PSOparams) %调用 PSO 核心模块
```

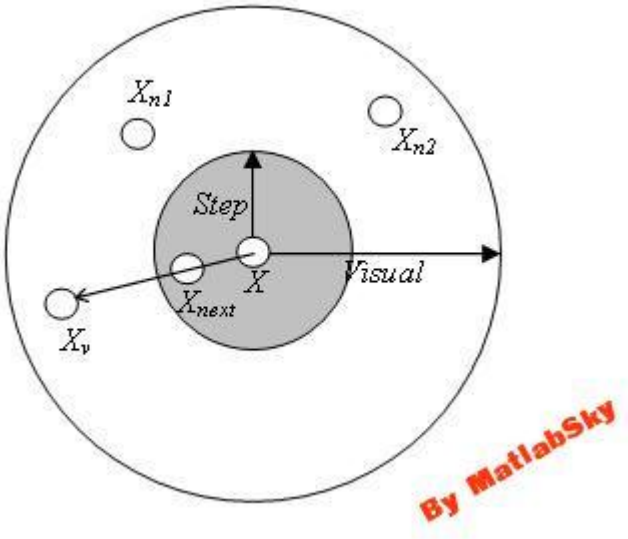
第 18 章　**基于鱼群算法的函数寻优算法**

1、人工鱼群算法原理

人工鱼群算法是李晓磊等人于 2002 年提出的一类基于动物行为的群体智能优化算法. 该算法是通过模拟鱼类的觅食、聚群、追尾、随机等行为在搜索域中进行寻优，是集群体智能思想的一个具体应用.

生物的视觉是极其复杂的，它能快速感知大量的空间事物，这对于任何仪器和程序都是难以与之相比的，为了实施的简便和有效，在鱼群模式中应用了如下的方法来实现虚拟人工鱼的视觉：

如图 5.1 所示，一虚拟人工鱼实体的当前位置为 ，它的视野范围为 ，位置 为其在某时刻的视点所在的位置，如果该位置的食物浓度高于当前位置，则考虑向该位置方向前进一步，即到达位置 ；如果位置 不比当前位置食物浓度更高，则继续巡视视野内的其他位置。巡视的次数越多，则对视野内的状态了解更全面，从而对周围的环境有一个全方面立体的认知，这有助于做出相应的判断和决策，当然，对于状态多或无限状态的环境也不必全部遍历，允许一定的不确定性对于摆脱局部最优，从而寻找全局最优是有帮助的。



人工鱼的视野和移动步长
其中，位置 ，位置 ，则该过程可以表示如下：

式中， 函数为产生范围(-1, 1)之间的随机数，为移动步长。由于环境中同伴的数目是有限的，因此在视野中感知同伴的位置，并相应的调整自身位置的方法与上式类似。

2、案例目录：

3、案例实例及结果：

案例 1：

一元函数的优化实例：，该函数的图形如图 5.2 所示：

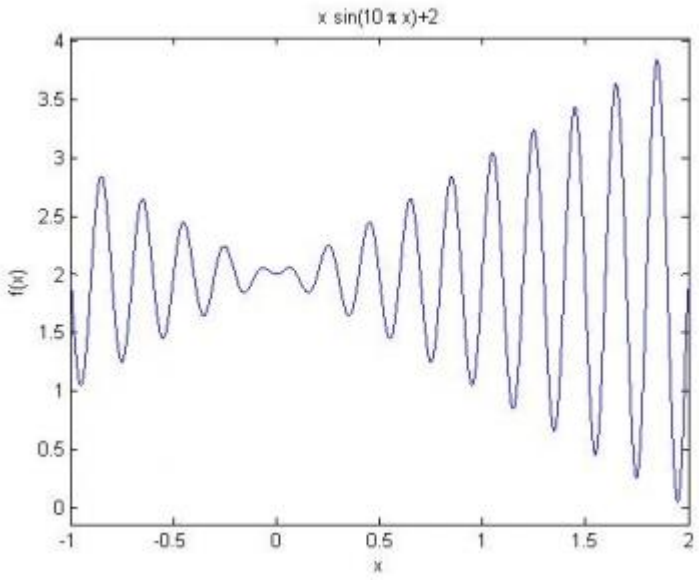


图 5.2 一元函数图像

结果：
鱼群算法的运行结果如下:

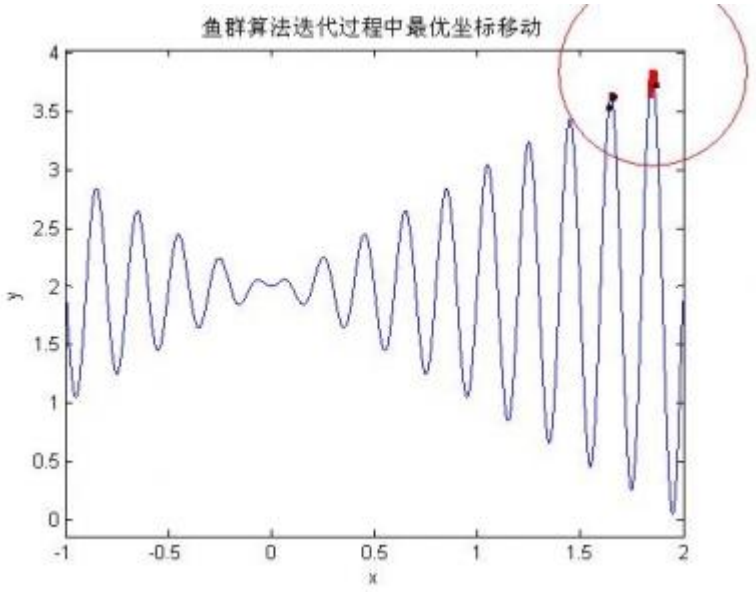


图 5.8 50 次鱼群算法迭代结果

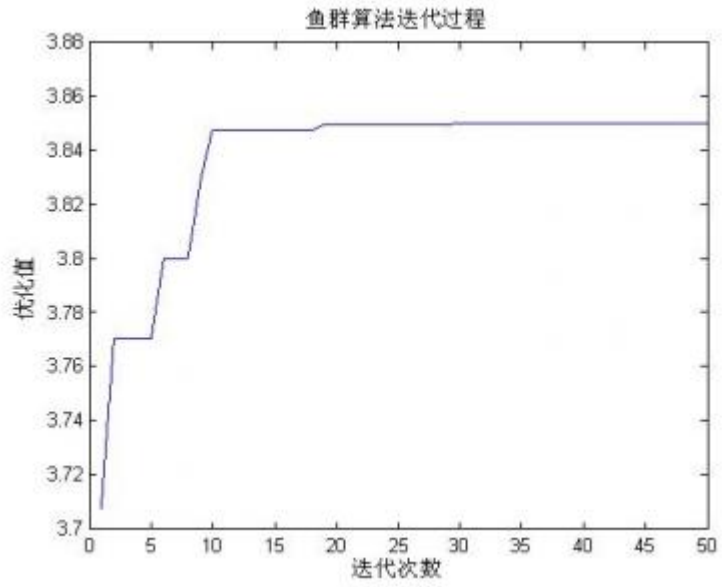


图 5.9 最优解的变化 C

ommand Window 上的运行结果：
最优解 X：1.85060
最优解 Y：3.85027
Elapsed time is 1.640857 seconds.

案例 2
二元函数的优化实例：
该函数的图形如图 5.3 所示：

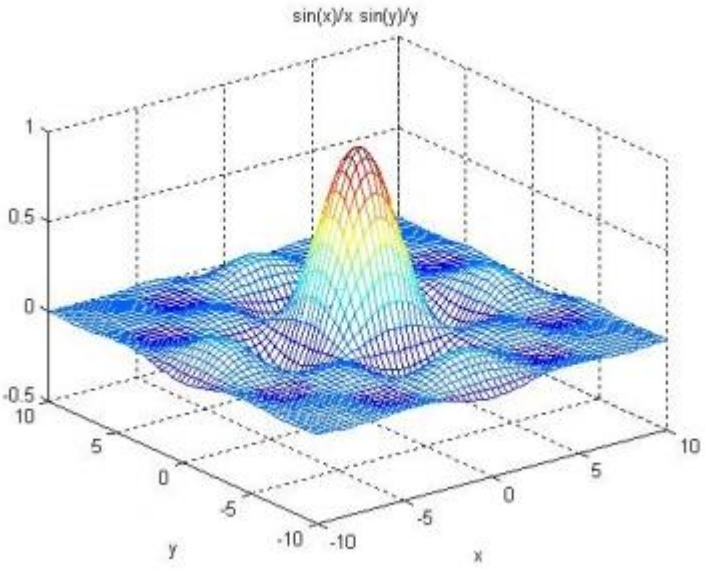


图 5.3 二元函数图像

结果：
鱼群算法的运行结果如下:

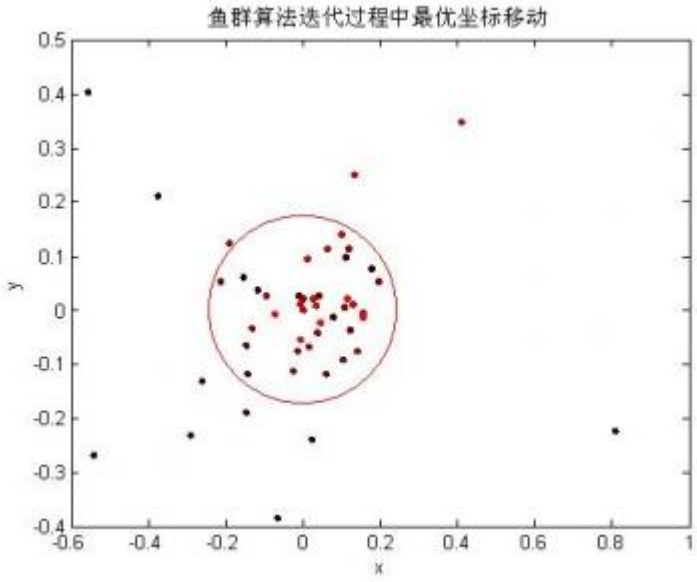


图 5.10 50 次鱼群算法迭代结果

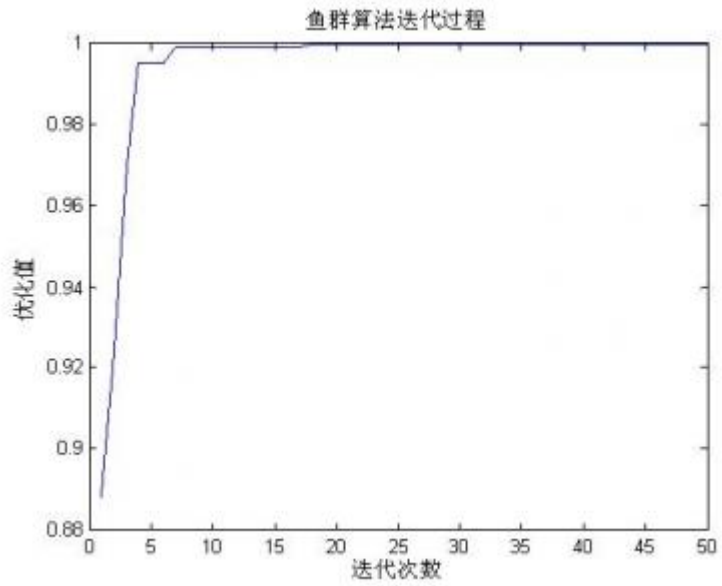


图 5.11 最优解的变化

Command Window 上的运行结果：
最优解 X：-0.002690.00018
最优解 Y：1.00000
Elapsed time is 3.094503 seconds.

4、主程序

案例 1：

```
clc
clear all
close all
tic
figure(1);hold on
ezplot('x*sin(10*pi*x)+2',[-1,2]);
%% 参数设置
fishnum=50; %生成 50 只人工鱼
MAXGEN=50; %最多迭代次数
try_number=100;%最多试探次数
visual=1; %感知距离
delta=0.618; %拥挤度因子
step=0.1; %步长
%% 初始化鱼群
lb_ub=[-1,2,1];
X=AF_init(fishnum,lb_ub);
LBUB=[];
for i=1:size(lb_ub,1)
```

```
LBUB=[LBUB;repmat(lb_ub(i,1:2),lb_ub(i,3),1)];
end
gen=1;
BestY=-1*ones(1,MAXGEN); %每步中最优的函数值
BestX=-1*ones(1,MAXGEN); %每步中最优的自变量
besty=-100; %最优函数值
Y=AF_foodconsistence(X);
while gen<=MAXGEN
fprintf(1,'%d\n',gen)
for i=1:fishnum
[Xi1,Yi1]=AF_swarm(X,i,visual,step,delta,try_number,LBUB,Y); %聚群行为
[Xi2,Yi2]=AF_follow(X,i,visual,step,delta,try_number,LBUB,Y); %追**为
if Yi1>Yi2
X(:,i)=Xi1;
Y(1,i)=Yi1;
else
X(:,i)=Xi2;
Y(1,i)=Yi2;
end
end
[Ymax,index]=max(Y);
figure(1);
plot(X(1,index),Ymax,'.','color',[gen/MAXGEN,0,0])
if Ymax>besty
besty=Ymax;
bestx=X(:,index);
BestY(gen)=Ymax;
[BestX(:,gen)]=X(:,index);
else
BestY(gen)=BestY(gen-1);
[BestX(:,gen)]=BestX(:,gen-1);
end
gen=gen+1;
end
plot(bestx(1),besty,'ro','MarkerSize',100)
xlabel('x')
ylabel('y')
title('鱼群算法迭代过程中最优坐标移动')
figure
plot(1:MAXGEN,BestY)
xlabel('迭代次数')
ylabel('优化值')
title('鱼群算法迭代过程')
disp(['最优解 X： ',num2str(bestx,'%1.5f')])
```

```
disp(['最优解 Y： ',num2str(besty,'%1.5f')])
toc
```

案例 2：

```
clc
clear all
close all
tic
figure(1);hold on
%% 参数设置
fishnum=100; %生成 50 只人工鱼
MAXGEN=50; %最多迭代次数
try_number=100;%最多试探次数
visual=1; %感知距离
delta=0.618; %拥挤度因子
step=0.1; %步长
%% 初始化鱼群
lb_ub=[-10,10,2;];
X=AF_init(fishnum,lb_ub);
LBUB=[];
for i=1:size(lb_ub,1)
LBUB=[LBUB;repmat(lb_ub(i,1:2),lb_ub(i,3),1)];
end
gen=1;
BestY=-1*ones(1,MAXGEN); %每步中最优的函数值
BestX=-1*ones(2,MAXGEN); %每步中最优的自变量
besty=-100; %最优函数值
Y=AF_foodconsistence(X);
while gen<=MAXGEN
fprintf(1,'%d\n',gen)
for i=1:fishnum
[Xi1,Yi1]=AF_swarm(X,i,visual,step,delta,try_number,LBUB,Y); %聚群行为
[Xi2,Yi2]=AF_follow(X,i,visual,step,delta,try_number,LBUB,Y); %追**为
if Yi1>Yi2
X(:,i)=Xi1;
Y(1,i)=Yi1;
else
X(:,i)=Xi2;
Y(1,i)=Yi2;
end
end
[Ymax,index]=max(Y);
figure(1);
plot(X(1,index),X(2,index),'.','color',[gen/MAXGEN,0,0])
```

```
if Ymax>besty
besty=Ymax;
bestx=X(:,index);
BestY(gen)=Ymax;
[BestX(:,gen)]=X(:,index);
else
BestY(gen)=BestY(gen-1);
[BestX(:,gen)]=BestX(:,gen-1);
end
gen=gen+1;
end
plot(bestx(1),bestx(2),'ro','MarkerSize',100)
xlabel('x')
ylabel('y')
title('鱼群算法迭代过程中最优坐标移动')
figure
plot(1:MAXGEN,BestY)
xlabel('迭代次数')
ylabel('优化值')
title('鱼群算法迭代过程')
disp(['最优解 X：',num2str(bestx','%1.5f')])
disp(['最优解 Y：',num2str(besty,'%1.5f')])
toc
```

第 19 章　**基于模拟退火算法的 TSP 算法**

1、案例背景

模拟退火算法（Simulated Annealing，简称 SA）的思想最早是由 Metropolis 等提出的。其出发点是基于物理中固体物质的退火过程与一般的组合优化问题之间的相似性。模拟退火法是一种通用的优化算法，其物理退火过程由以下三部分组成：

（1）加温过程。其目的是增强粒子的热运动，使其偏离平衡位置。当温度足够高时，固体将熔为液体，从而消除系统原先存在的非均匀状态。

（2）等温过程。对于与周围环境交换热量而温度不变的封闭系统，系统状态的自发变化总是朝自由能减少的方向进行的，当自由能达到最小时，系统达到平衡状态。

（3）冷却过程。使粒子热运动减弱，系统能量下降，得到晶体结构。

其中，加温过程对应算法的设定初温，等温过程对应算法的 Metropolis 抽样过程，冷却过程对应控制参数的下降。这里能量的变化就是目标函数，我们要得到的最优解就是能量最低

态。其中 Metropolis 准则是 SA 算法收敛于全局最优解的关键所在，Metropolis 准则以一定的概率接受恶化解，这样就使算法跳离局部最优的陷阱。

2、案例目录：

3、主程序：

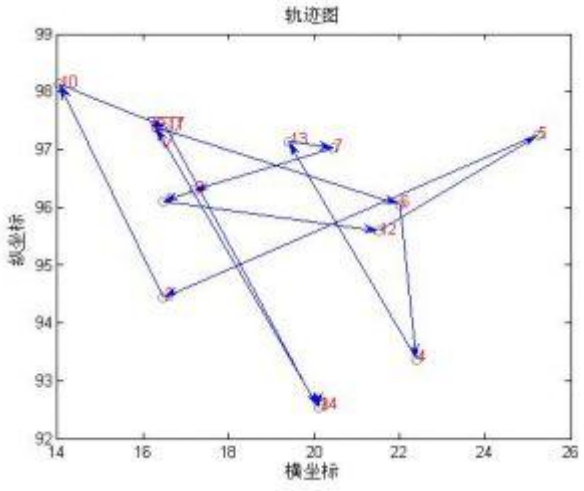
```
clc;
clear;
close all;
%%
tic
T0=1000; %  初始温度
Tend=1e-3; %  终止温度
L=200; %  各温度下的迭代次数（链长）
q=0.9; %降温速率
```

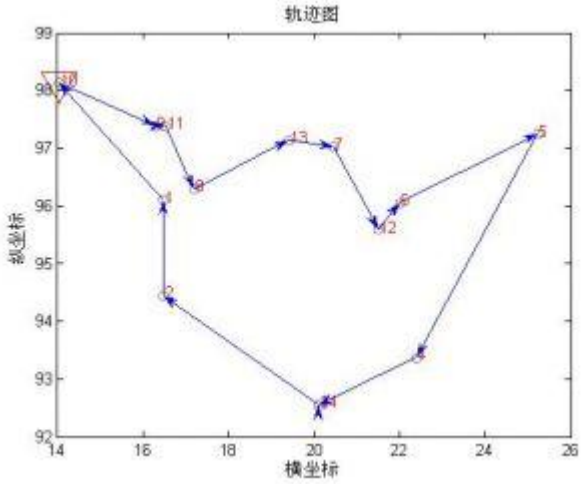
```
load CityPosition1; %加载数据

%%
D=Distanse(X); %计算距离矩阵
N=size(D,1); %城市的个数
%% 初始解
S1=randperm(N); %随机产生一个初始路线

%% 画出随机解的路径图
DrawPath(S1,X)
pause(0.0001)
%% 输出随机解的路径和总距离
disp('初始种群中的一个随机值:')
OutputPath(S1);
Rlength=PathLength(D,S1);
disp(['总距离：',num2str(Rlength)]);

%% 计算迭代的次数 Time
Time=ceil(double(solve(['1000*(0.9)^x=',num2str(Tend)])));
count=0; %迭代计数
Obj=zeros(Time,1); %目标值矩阵初始化
track=zeros(Time,N); %每代的最优路线矩阵初始化
%% 迭代
while T0>Tend
count=count+1; %更新迭代次数
temp=zeros(L,N+1);
for k=1:L
%% 产生新解
S2=NewAnswer(S1);
%% Metropolis 法则判断是否接受新解
[S1,R]=Metropolis(S1,S2,D,T0); %Metropolis 抽样算法
temp(k,:)=[S1 R]; %记录下一路线的及其路程
end
%% 记录每次迭代过程的最优路线
[d0,index]=min(temp(:,end)); %找出当前温度下最优路线
if count==1 || d0<Obj(count-1)
Obj(count)=d0; %如果当前温度下最优路程小于上一路程则记录当前路程
else
Obj(count)=Obj(count-1);%如果当前温度下最优路程大于上一路程则记录上一路程
end
track(count,:)=temp(index,1:end-1); %记录当前温度的最优路线
T0=q*T0; %降温
fprintf(1,'%d\n',count) %输出当前迭代次数
end
```

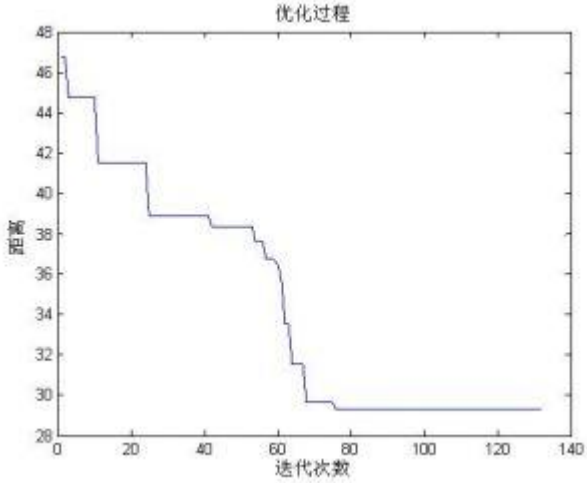
```
%% 优化过程迭代图
figure
plot(1:count,Obj)
xlabel('迭代次数')
ylabel('距离')
title('优化过程')

%% 最优解的路径图
DrawPath(track(end,:),X)

%% 输出最优解的路线和总距离
disp('最优解:')
S=track(end,:);
p=OutputPath(S);
disp(['总距离：',num2str(PathLength(D,S))]);
disp('------------------------------------------------------------')
toc
```

4、运行结果：

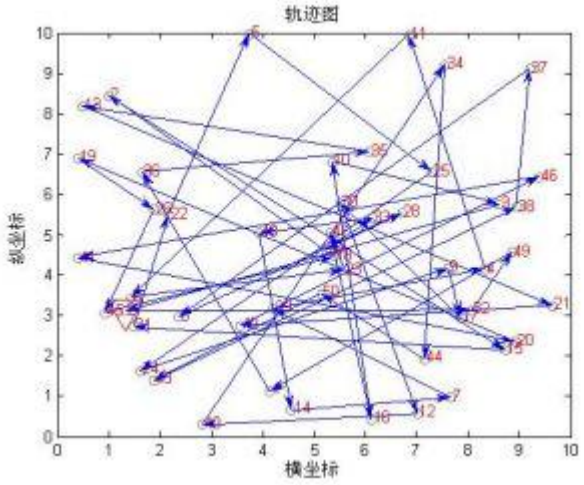
N=14：

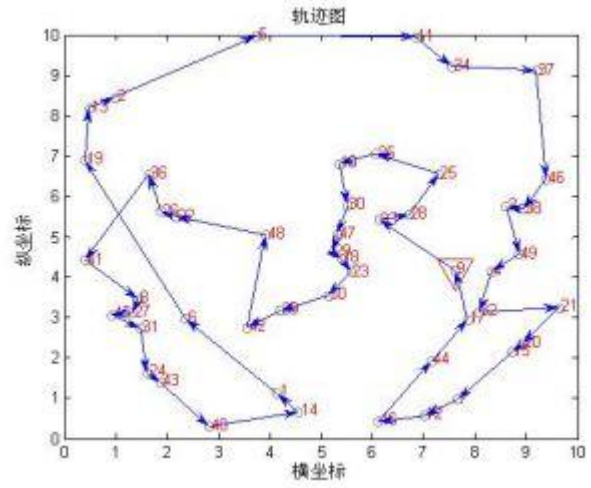


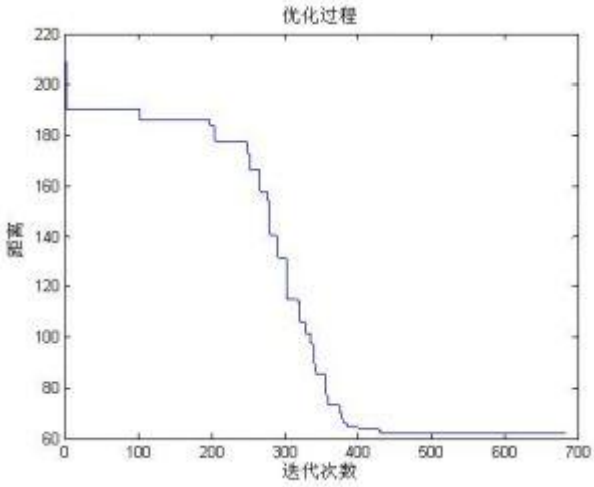




N=50：







第 20 章　**基于遗传模拟退火算法的聚类算法**

1、案例背景

模糊 C-均值聚类（FCM） 是目前比较流行的一种聚类方法。该方法使用了在欧几里德

空间确定数据点的几何贴近度的概念，它将这些数据分配到不同的聚类，然后确定这些聚类之间的距离。模糊 C-均值聚类算法在理论和应用上都为其他的模糊聚类分析方法奠定了基础，应用也最广泛。但是，从本质上 FCM 算法是一种局部搜索优化算法，如果初始值选择不当，它就会收敛到局部极小点上。因此，FCM 算法的这一缺点限制了人们对它的使用。

将模拟退火算法与遗传算法相结合用于聚类分析，由于模拟退火算法和遗传算法可以互相取长补短，因此有效地克服了传统遗传算法的早熟现象，同时根据聚类问题的具体情况设计遗传编码方式、适应度函数，使该算法更有效、更快速地收敛到全局最优解。

本章将 SAGA 作用于随机产生的数据进行实验。数据由 400 个二维平面上的点组成，这些点构成 4 个集合，但彼此之间并没有明显的界限，数据如图 20-1 所示。通过使用单纯的 FCM 聚类和 SAGA 优化初始聚类中心点后的 FCM 聚类，来说明 SAGA 优势。

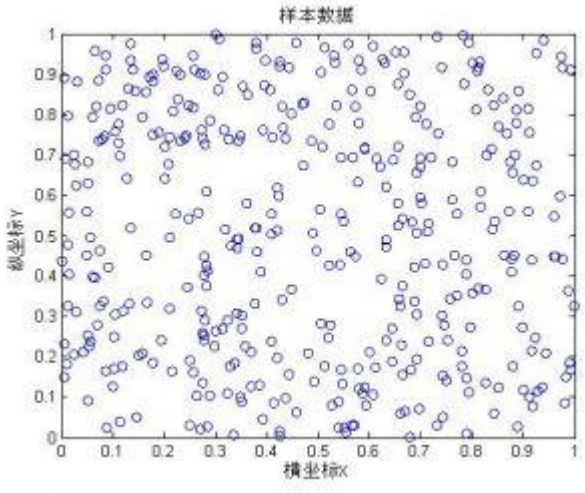


图 20-1 400 个随机样本数据

2、案例目录：

3、主程序：

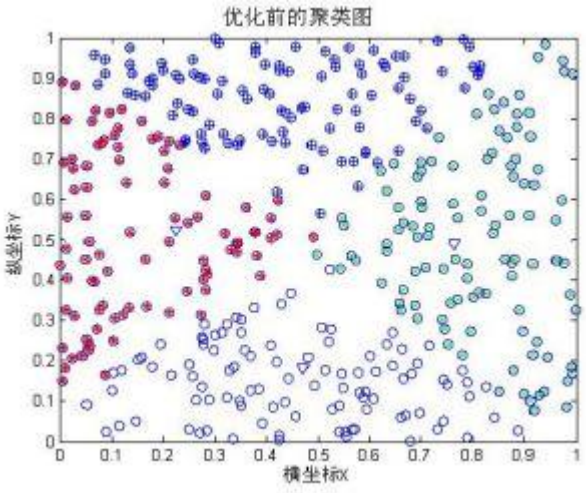
```
clear all;close all
load X
m=size(X,2);% 样本特征维数
% 中心点范围[lb;ub]
lb=min(X);
ub=max(X);
%% 模糊 C 均值聚类参数
% 设置幂指数为 3，最大迭代次数为 20，目标函数的终止容限为 1e-6
options=[3,20,1e-6];
% 类别数 cn
cn=4;
%% 模拟退火算法参数
q =0.8; % 冷却系数
T0=100; % 初始温度
Tend=1; % 终止温度
%% 定义遗传算法参数
sizepop=10; %个体数目(Numbe of individuals)
MAXGEN=10; %最大遗传代数(Maximum number of generations)
NVAR=m*cn; %变量的维数
PRECI=10; %变量的二进制位数(Precision of variables)
GGAP=0.9; %代沟(Generation gap)
pc=0.7;
pm=0.01;
trace=zeros(NVAR+1,MAXGEN);
%建立区域描述器(Build field descriptor)
FieldD=[rep([PRECI],[1,NVAR]);rep([lb;ub],[1,cn]);rep([1;0;1;1],[1,NVAR])];
Chrom=crtbp(sizepop, NVAR*PRECI); % 创建初始种群
V=bs2rv(Chrom, FieldD);
ObjV=ObjFun(X,cn,V,options); %计算初始种群个体的目标函数值
T=T0;
while T>Tend
gen=0; %代计数器
while gen<MAXGEN %迭代
FitnV=ranking(ObjV); %分配适应度值(Assign fitness values)
SelCh=select('sus', Chrom, FitnV, GGAP); %选择
SelCh=recombin('xovsp', SelCh,pc); %重组
```

```
SelCh=mut(SelCh,pm); %变异
V=bs2rv(SelCh, FieldD);
ObjVSel=ObjFun(X,cn,V,options); %计算子代目标函数值
[newChrom newObjV]=reins(Chrom, SelCh, 1, 1, ObjV, ObjVSel); %重插入
V=bs2rv(newChrom,FieldD);
%是否替换旧个体
for i=1:sizepop
if ObjV(i)>newObjV(i)
ObjV(i)=newObjV(i);
Chrom(i,:)=newChrom(i,:);
else
p=rand;
if p<=exp((newObjV(i)-ObjV(i))/T)
ObjV(i)=newObjV(i);
Chrom(i,:)=newChrom(i,:);
end
end
end
gen=gen+1; %代计数器增加
[trace(end,gen),index]=min(ObjV); %遗传算法性能跟踪
trace(1:NVAR,gen)=V(index,:);
fprintf(1,'%d ',gen);
end
T=T*q;
fprintf(1,'\n 温度:%1.3f\n',T);
end
[newObjV,center,U]=ObjFun(X,cn,[trace(1:NVAR,end)]',options);  %计算最佳初始聚类中心的目标函数值
%  查看聚类结果
Jb=newObjV
U=U{1}
center=center{1}
figure
plot(X(:,1),X(:,2),'o')
hold on
maxU = max(U);
index1 = find(U(1,:) == maxU);
index2 = find(U(2, :) == maxU);
index3 = find(U(3, :) == maxU);
%  在前三类样本数据中分别画上不同记号  不加记号的就是第四类了
line(X(index1,1), X(index1, 2), 'linestyle', 'none','marker', '*', 'color', 'g');
line(X(index2,1), X(index2, 2), 'linestyle', 'none', 'marker', '*', 'color', 'r');
line(X(index3,1), X(index3, 2), 'linestyle', 'none', 'marker', '*', 'color', 'b');
%  画出聚类中心
```

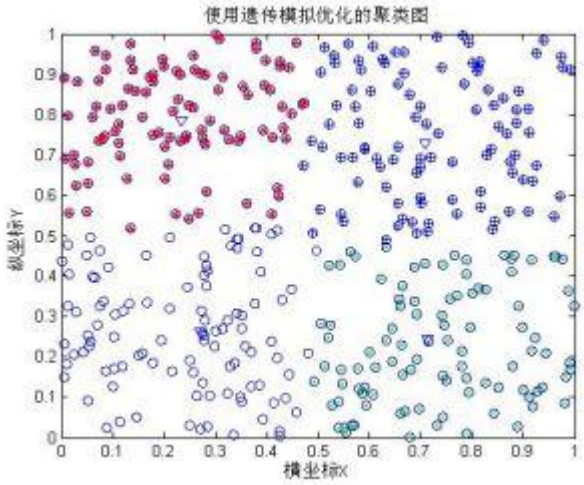
```
plot(center(:,1),center(:,2),'v')
hold off
```

4、运行结果：

随机初始的聚类中心的 FCM 聚类：



遗传算法与模拟退火算法优化初始聚类中心后得到的聚类图：



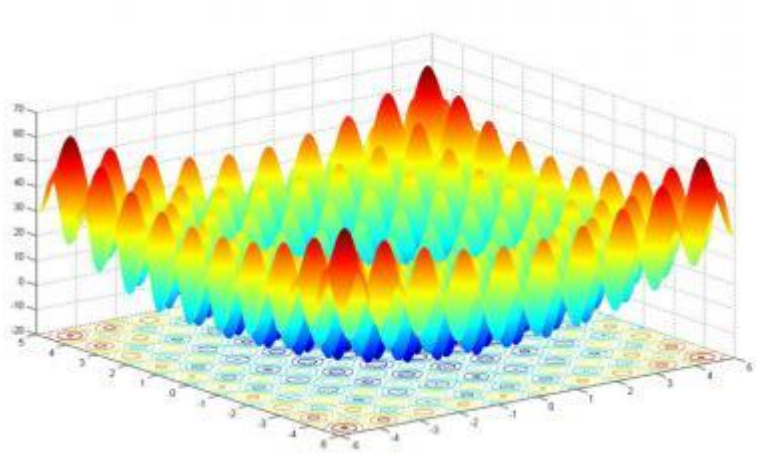
1、案例背景

模拟退火算法（simulated annealing, SA）是一种模拟固体退火过程的迭代搜索优化算法。在每一次迭代过程中，SA 随机产生一个新的搜索点，该新点与当前点之间的距离，或者说算法搜索的范围，与固体退火的温度有关，具体地说，随着固体温度的下降，算法搜索的范围会越来越小，以使 SA 收敛于最小值点。假设是最小值优化问题，那么当新点的目标函数值比当前点的目标函数值小时，SA 毫不犹豫地接受该新点，使其成为下一次迭代的当前点，而当新点的目标函数值比当前点的目标函数值大时，SA 不是一刀切地拒绝该新点，而是以一定的概率接受该新点，该概率的大小也与固体温度的高低有关。通过这样一种接受目标函数值比当前点的目标函数值差的新点的方式，SA 可以跳出局部最优值，有可能搜索到全局最优点。

2、案例目录：

3、案例实例及结果：

作为案例，这里将使用模拟退火算法工具箱(SAT)求 Rastrigin 函数的最小值。其图形如下所示：



优化过程如下：

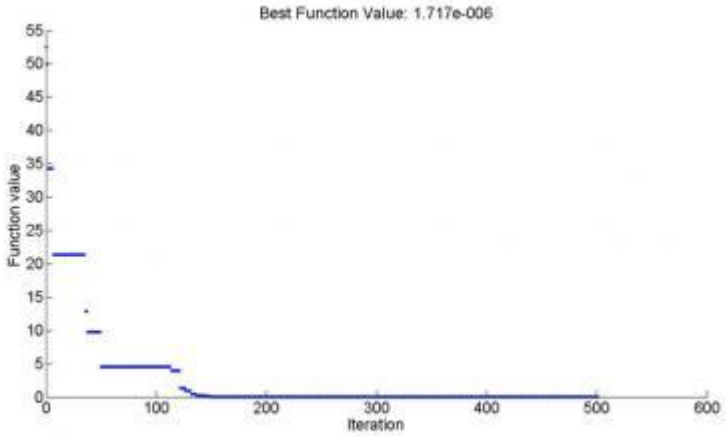


图 21.3 某次得到的最优解目标函数值历程曲线

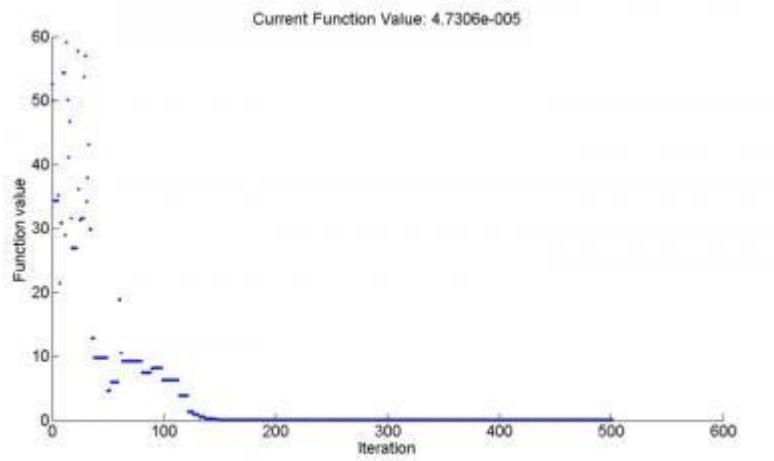


图 21.4 某次得到的当前解目标函数值历程曲线

所得最优解及其对应的目标函数值为：

$$(x_1, x_2) = [-8.2734\times10^{-5}, 4.2540\times10^{-5}]$$

$$y = 1.7170\times10^{-6}$$

4、主程序：

```
clear
clc
ObjectiveFunction = @my_first_SA;      % Function handle to the objective function
X0 = [2.5 2.5];              % Starting point
lb = [-5 -5];                    % Lower bound
ub = [5 5];                      % Upper bound
options                  =                  saoptimset('MaxIter',500,'StallIterLim',500,'TolFun',1e-100,
'AnnealingFcn',@annealingfast,'InitialTemperature',100,'TemperatureFcn',@temperatureexp,'Rea
nnealInterval',500, 'PlotFcns',{@saplotbestx, @saplotbestf, @saplotx, @saplotf});
[x,fval] = simulannealbnd(ObjectiveFunction,X0,lb,ub,options);
```

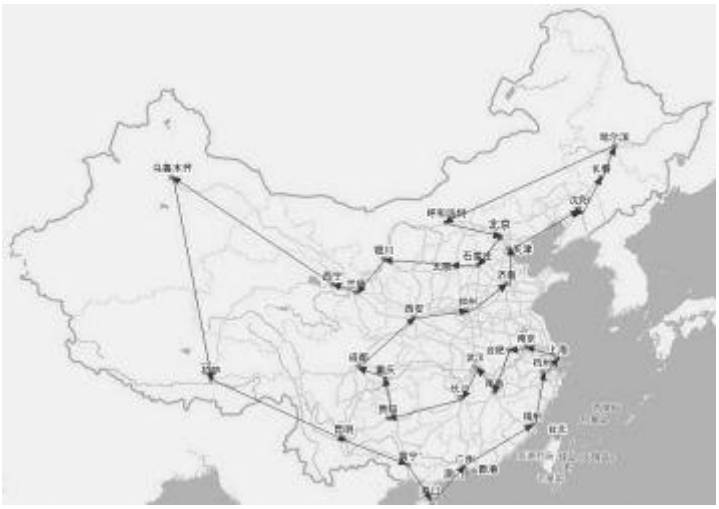
第 22 章　蚁群算法的优化计算——旅行商问题(TSP)优化

1、案例背景

蚁群算法（Ant Colony Algorithm，ACA）是由意大利学者 M.Dorigo 等人于 20 世纪 90 年代初提出的一种新的模拟进化算法，其真实地模拟了自然界蚂蚁群体的觅食行为。M.Dorigo 等人将其应用于解决旅行商问题（Traveling Salesman Problem，TSP），取得了较好的实验结果。

近年来，许多专家与学者致力于蚁群算法的研究，并将其应用于交通、通信、化工、电力等领域，成功解决了许多组合优化问题，如调度问题（Job-shop Scheduling Problem）、指派问题（Quadratic Assignment Problem）、旅行商问题（Traveling Salesman Problem）等。本章将详细阐述蚁群算法的基本思想及原理，并以实例的形式介绍其应用于解决中国旅行商问题（Chinese TSP，CTSP）的情况。

按照枚举法，我国 31 个直辖市、省会和自治区首府（未包括港、澳、台）的巡回路径应有约 1.326*1032 种，其中一条路径如图 22-2 所示。试利用蚁群算法寻找到一条最佳或者较佳的路径。



2、案例目录：

3、主程序：

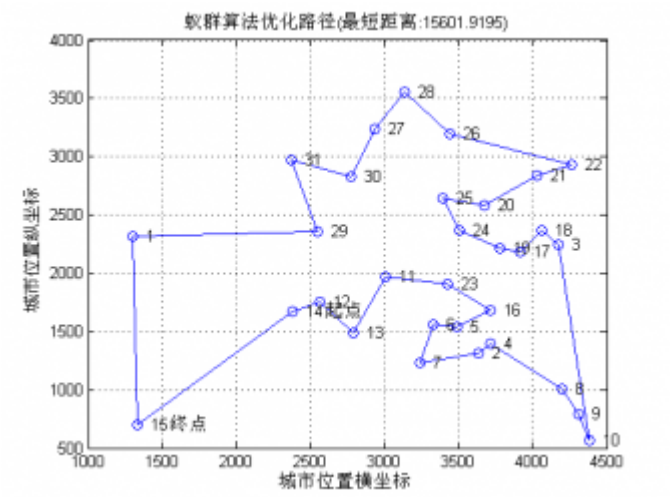
```
%% 清空环境变量
clear all
clc

%% 导入数据
load citys_data.mat

%% 计算城市间相互距离
n = size(citys,1);
D = zeros(n,n);
for i = 1:n
    for j = 1:n
        if i ~= j
            D(i,j) = sqrt(sum((citys(i,:) - citys(j,:)).^2));
        else
            D(i,j) = 1e-4;
        end
    end
end

%% 初始化参数
m = 50;                                 % 蚂蚁数量
alpha = 1;                              % 信息素重要程度因子
beta = 5;                               % 启发函数重要程度因子
rho = 0.1;                              % 信息素挥发因子
Q = 1;                                  % 常系数
Eta = 1./D;                             % 启发函数
Tau = ones(n,n);                        % 信息素矩阵
Table = zeros(m,n);                     % 路径记录表
iter = 1;                               % 迭代次数初值
iter_max = 200;                         % 最大迭代次数
Route_best = zeros(iter_max,n);         % 各代最佳路径
Length_best = zeros(iter_max,1);        % 各代最佳路径的长度
Length_ave = zeros(iter_max,1);         % 各代路径的平均长度
```

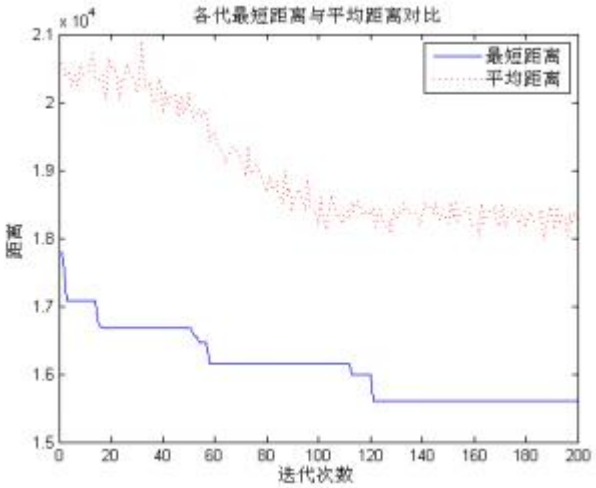
```
%% 迭代寻找最佳路径
while iter <= iter_max
    % 随机产生各个蚂蚁的起点城市
      start = zeros(m,1);
      for i = 1:m
          temp = randperm(n);
          start(i) = temp(1);
      end
      Table(:,1) = start;
      % 构建解空间
      citys_index = 1:n;
      % 逐个蚂蚁路径选择
      for i = 1:m
          % 逐个城市路径选择
        for j = 2:n
            tabu = Table(i,1:(j - 1));           % 已访问的城市集合(禁忌表)
            allow_index = ~ismember(citys_index,tabu);
            allow = citys_index(allow_index);  % 待访问的城市集合
            P = allow;
            % 计算城市间转移概率
            for k = 1:length(allow)
                P(k) = Tau(tabu(end),allow(k))^alpha * Eta(tabu(end),allow(k))^beta;
            end
            P = P/sum(P);
            % 轮盘赌法选择下一个访问城市
            Pc = cumsum(P);
          target_index = find(Pc >= rand);
          target = allow(target_index(1));
          Table(i,j) = target;
        end
      end
      % 计算各个蚂蚁的路径距离
      Length = zeros(m,1);
      for i = 1:m
          Route = Table(i,:);
          for j = 1:(n - 1)
              Length(i) = Length(i) + D(Route(j),Route(j + 1));
          end
          Length(i) = Length(i) + D(Route(n),Route(1));
      end
      % 计算最短路径距离及平均距离
      if iter == 1
          [min_Length,min_index] = min(Length);
          Length_best(iter) = min_Length;
```

```
            Length_ave(iter) = mean(Length);
            Route_best(iter,:) = Table(min_index,:);
        else
            [min_Length,min_index] = min(Length);
            Length_best(iter) = min(Length_best(iter - 1),min_Length);
            Length_ave(iter) = mean(Length);
            if Length_best(iter) == min_Length
                Route_best(iter,:) = Table(min_index,:);
            else
                Route_best(iter,:) = Route_best((iter-1),:);
            end
        end
        % 更新信息素
        Delta_Tau = zeros(n,n);
        % 逐个蚂蚁计算
        for i = 1:m
            % 逐个城市计算
            for j = 1:(n - 1)
                Delta_Tau(Table(i,j),Table(i,j+1)) = Delta_Tau(Table(i,j),Table(i,j+1)) + Q/Length(i);
            end
            Delta_Tau(Table(i,n),Table(i,1)) = Delta_Tau(Table(i,n),Table(i,1)) + Q/Length(i);
        end
        Tau = (1-rho) * Tau + Delta_Tau;
    % 迭代次数加 1，清空路径记录表
    iter = iter + 1;
    Table = zeros(m,n);
end

%% 结果显示
[Shortest_Length,index] = min(Length_best);
Shortest_Route = Route_best(index,:);
disp(['最短距离:' num2str(Shortest_Length)]);
disp(['最短路径:' num2str([Shortest_Route Shortest_Route(1)])]);

%% 绘图
figure(1)
plot([citys(Shortest_Route,1);citys(Shortest_Route(1),1)],...
     [citys(Shortest_Route,2);citys(Shortest_Route(1),2)],'o-');
grid on
for i = 1:size(citys,1)
    text(citys(i,1),citys(i,2),['      ' num2str(i)]);
end
text(citys(Shortest_Route(1),1),citys(Shortest_Route(1),2),'            起点');
text(citys(Shortest_Route(end),1),citys(Shortest_Route(end),2),'            终点');
```

```
xlabel('城市位置横坐标')
ylabel('城市位置纵坐标')
title(['蚁群算法优化路径(最短距离:' num2str(Shortest_Length) ')'])
figure(2)
plot(1:iter_max,Length_best,'b',1:iter_max,Length_ave,'r:')
legend('最短距离','平均距离')
xlabel('迭代次数')
ylabel('距离')
title('各代最短距离与平均距离对比')
```

4、运行结果：





第 23 章　基于蚁群算法的二维路径规划算

1、案例背景

路径规划算法是指在有障碍物的工作环境中寻找一条从起点到终点的，无碰撞的绕过所有障碍物运动路径。路径规划算法较多，大体上可分为全局路径规划算法和局部路径规划方法两类，其中，全局路径规划方法包括位形空间法、广义锥方法、顶点图像法、栅格划归法；局部路径规划算法主要有人工势场法等。

MAKLINK 图论可以建立二维路径规划的空间模型，MAKLINK 图论通过生成大量的 MAKLINK 线构造二维路径规划可行空间，MAKLINK 线定义为两个障碍物之间不与障碍物相交的顶点之间的连线，以及障碍物顶点与边界相交的连线。

蚁群算法是由 Dorigo M 等人在 20 世纪 90 年代初提出的一种新型进化算法，它来源于对蚂蚁搜索问题的研究。人们在观察蚂蚁搜索食物的时候发现，蚂蚁在寻找食物时，总在走过的路径上释放一种称为信息素的分泌物，信息素能够保留一段时间，使得在一定范围内的其他蚂蚁能够觉察到该信息素的存在。后继蚂蚁在选择路径时，会选择信息素浓度较高的路径，并且在经过时留下自己的信息素，这样该路径的信息素会不断增强，蚂蚁选择的概率也在不断增大。蚁群算法最优路径寻找如图 23-3 所示。

2、案例目录

3、主程序

```
%% 循环搜索
```

```
for num = 1:NC

%% 蚂蚁迭代寻优一次
for i=1:pathCount
for k=1:m
q = rand();
qfz(i,:) = (qfzPara2-abs((1:10)'/10-qfzPara1))/qfzPara2; %启发信息
if q<=pheThres %选择信息素最大值
arg = phePara(i,:).*(qfz(i,:).^pheCacuPara);
j = find(arg == max(arg));
pathk(i,k) = j(1);
else % 轮盘赌选择
arg = phePara(i,:).*(qfz(i,:).^pheCacuPara);
sumarg = sum(arg);
qq = (q-qo)/(1-qo);
qtemp = 0;
j = 1;
while qtemp < qq
qtemp = qtemp + (phePara(i,j)*(qfz(i,j)^pheCacuPara))/sumarg;
j=j+1;
end
j=j-1;
pathk(i,k) = j(1);
end
% 信息素更新
phePara(i,j) = (1-pheUpPara(1))*phePara(i,j)+pheUpPara(1)*pheUpPara(2);
end
end

%% 计算路径长度
len = zeros(1,k);
for k=1:m
Pstart = S;
Pend = lines(1,1:2) + (lines(1,3:4)-lines(1,1:2))*pathk(1,k)/10;
for l=1:pathCount
len(1,k) = len(1,k)+sqrt(sum((Pend-Pstart).^2));
Pstart = Pend;
if l<pathCount
Pend = lines(l+1,1:2) + (lines(l+1,3:4)-lines(l+1,1:2))*pathk(l+1,k)/10;
end
end
Pend = T;
len(1,k) = len(1,k)+sqrt(sum((Pend-Pstart).^2));
end
```

```
%% 更新信息素
% 寻找最短路径
minlen = min(len);
minlen = minlen(1);
minant = find(len == minlen);
minant = minant(1);

% 更新全局最短路径
if minlen < LL
LL = minlen;
end

% 更新信息素
for i=1:pathCount
phePara(i,pathk(i,minant)) = (1-pheUpPara(1))*
phePara(i,pathk(i,minant))+pheUpPara(1)*(1/minlen);
end
shortestpath(num) = minlen;
end
```

第 24 章 **基于蚁群算法的三维路径规划算**

1、案例背景

三维路径规划指在已知三维地图中，规划出一条从出发点到目标点满足某项指标最优，并且避开了所有三维障碍物的三维最优路径。现有的路径规划算法中，大部分算法是在二维规划平面或准二维规划平面中进行路径规划。一般的三维路径规划算法具有计算过程复杂，信息存储量大，难以直接进行全局规划等问题。已有的三维路径规划算法主要包括 A*算法，遗传算法，粒子群算法等，但是 A*算法的计算量会随着维数的增加而急剧增加，遗传算法和粒子群算法只是准三维规划算法。

蚁群算法具有分布计算、群体智能等优势，在路径规划上具有很大潜力，在成功用于二维路径规划的同时也可用于规划三维路径，本章采用蚁群算法进行水下机器人三维路径规划。

2、案例目录

3、主程序

```
%% 循环寻找最优路径
for kk=1:200
```

```
%% 路径搜索
[path,pheromone]=searchpath(PopNumber,LevelGrid,PortGrid,...
pheromone,HeightData,starty,starth,endy,endh);

%% 适应度值计算更新
fitness=CacuFit(path);
[newbestfitness,newbestindex]=min(fitness);
if newbestfitness<bestfitness
bestfitness=newbestfitness;
bestpath=path(newbestindex,:);
end
BestFitness=[BestFitness;bestfitness];

%% 更新信息素
cfit=100/bestfitness;
for i=2:PortGrid-1
pheromone(i,bestpath(i*2-1),bestpath(i*2))=(1-rou)* ...
pheromone(i,bestpath(i*2-1),bestpath(i*2))+rou*cfit;
end

end
```

第 25 章　**有导师学习神经网络的回归拟合——基于近红外光谱的汽油辛烷值预测**

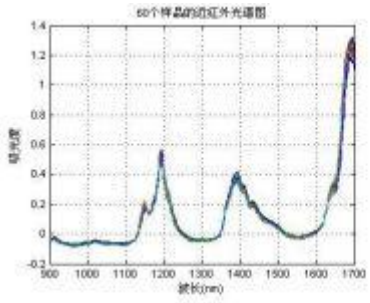
1、案例背景

神经网络的学习规则又称神经网络的训练算法，用来计算更新神经网络的权值和阈值。学习规则有两大类别：有导师学习和无导师学习。在有导师学习中，需要为学习规则提供一系列正确的网络输入/输出对（即训练样本），当网络输入时，将网络输出与相对应的期望值进行比较，然后应用学习规则调整权值和阈值，使网络的输出接近于期望值。而在无导师学习中，权值和阈值的调整只与网络输入有关系，没有期望值，这类算法大多用聚类法，将输入模式归类于有限的类别。本章将详细分析两种应用最广的有导师学习神经网络（BP 及 RBF）的原理及其在回归拟合中的应用。

辛烷值是汽油最重要的品质指标，传统的实验室检测方法存在样品用量大，测试周期长和费用高等问题，不适用于生产控制，特别是在线测试。近年发展起来的近红外光谱分析方法（NIR），作为一种快速分析方法，已广泛应用于农业、制药、生物化工、石油产品等领域。其优越性是无损检测、低成本、无污染，能在线分析，更适合于生产和控制的需要。

针对采集得到的 60 组汽油样品，利用傅里叶近红外变换光谱仪对其进行扫描，扫描范围 900～1700nm，扫描间隔 2nm，每个样品的光谱曲线共含 401 个波长点。样品的近红外光

谱曲线如图 25-3 所示。同时，利用传统实验室检测方法测定其辛烷值含量。现要求利用 BP 及 RBF 神经网络分别建立汽油样品近红外光谱及其辛烷值间的数学模型，并对模型的性能进行评价。



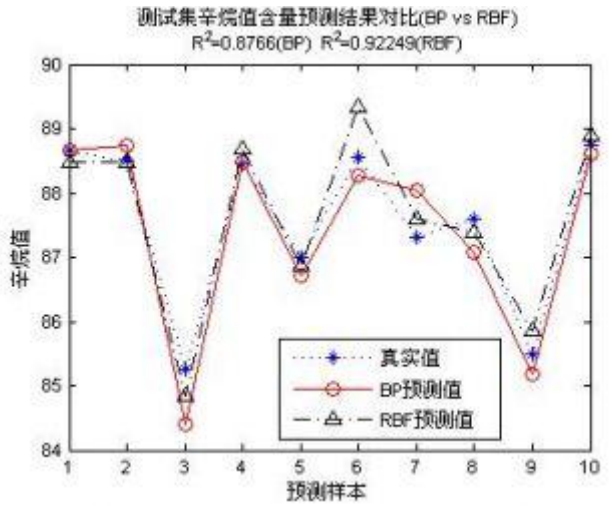
2、案例目录：

3、主程序：

```
%% 清空环境变量
clear all
clc

%% 训练集/测试集产生
load spectra_data.mat
% 随机产生训练集和测试集
temp = randperm(size(NIR,1));
% 训练集——50 个样本
P_train = NIR(temp(1:50),:)';
T_train = octane(temp(1:50),:)';
% 测试集——10 个样本
P_test = NIR(temp(51:end),:)';
T_test = octane(temp(51:end),:)';
N = size(P_test,2);

%% BP 神经网络创建、训练及仿真测试

% 创建网络
net = newff(P_train,T_train,9);
% 设置训练参数
net.trainParam.epochs = 1000;
net.trainParam.goal = 1e-3;
net.trainParam.lr = 0.01;
% 训练网络
net = train(net,P_train,T_train);
% 仿真测试
```

```
T_sim_bp = sim(net,P_test);

%% RBF 神经网络创建及仿真测试

% 创建网络
net = newrbe(P_train,T_train,0.3);
% 仿真测试
T_sim_rbf = sim(net,P_test);

%% 性能评价

% 相对误差 error
error_bp = abs(T_sim_bp - T_test)./T_test;
error_rbf = abs(T_sim_rbf - T_test)./T_test;
% 决定系数 R^2
R2_bp = (N * sum(T_sim_bp .* T_test) - sum(T_sim_bp) * sum(T_test))^2 / ((N *
sum((T_sim_bp).^2) - (sum(T_sim_bp))^2) * (N * sum((T_test).^2) - (sum(T_test))^2));
R2_rbf = (N * sum(T_sim_rbf .* T_test) - sum(T_sim_rbf) * sum(T_test))^2 / ((N *
sum((T_sim_rbf).^2) - (sum(T_sim_rbf))^2) * (N * sum((T_test).^2) - (sum(T_test))^2));
% 结果对比
result_bp = [T_test' T_sim_bp' T_sim_rbf' error_bp' error_rbf']

%% 绘图
figure
plot(1:N,T_test,'b:*',1:N,T_sim_bp,'r-o',1:N,T_sim_rbf,'k-.^')
legend('真实值','BP 预测值','RBF 预测值')
xlabel('预测样本')
ylabel('辛烷值')
string = {'测试集辛烷值含量预测结果对比(BP vs RBF)';['R^2=' num2str(R2_bp) '(BP)' '  R^2='
num2str(R2_rbf) '(RBF)']};
title(string)
```

4、运行结果：



第 26 章　**有导师学习神经网络的分类——鸢尾花种类识别**

1、案例背景

有导师学习神经网络以其良好的学习能力广泛应用于各个领域中，其不仅可以解决拟合回归问题，亦可以用于模式识别、分类识别。本章将继续介绍两种典型的有导师学习神经网络（GRNN 和 PNN），并以实例说明其在分类识别中的应用。

植物的分类与识别时植物学研究和农林业生产经营中的重要基础工作，对于区分植物种类、探索植物间的亲缘关系、阐明植物系统的进化规律具有重要意义。目前常用的植物种类鉴别方法是利用分类检索表进行鉴定，但该方法花费时间较多，且分类检索表的建立是一件费时费力的工作，需投入大量的财力物力。

叶片的识别植物的重要组成部分，叶子的外轮廓是其主要形态特征。在提取叶子形态特征的基础上，利用计算机进行辅助分类与识别成为当前的主要研究方向，同时也是研究的热点与重点。

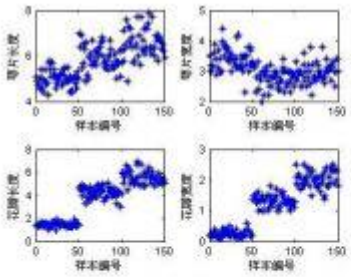
现采集到 150 组不同类型鸢尾花（Setosa、Versicolour 和 Virginica）的四种属性：萼片长度、萼片宽度、花瓣长度和花瓣宽度，样本编号与四种属性的关系如图 26-3 所示（其中，样本编号 1-50 为 Setosa，51-100 为 Versicolour，101-150 为 Virginica）。从图中大致可以看出，花瓣长度、花瓣宽度与鸢尾花类型间有较好的线性关系，而萼片长度、萼片宽度与鸢尾花类型间呈现出非线性的关系。

现要求：

（1）利用 GRNN 和 PNN 分别建立鸢尾花种类识别模型，并对模型的性能进行评价。

（2）利用 GRNN 和 PNN 分别建立各个属性及属性组合与鸢尾花种类间的识别模型，并与（1）中所建模型的性能及运算时间进行对比，从而探求各个属性及属性组合与鸢尾花种类

的相关程度。



2、案例目录：

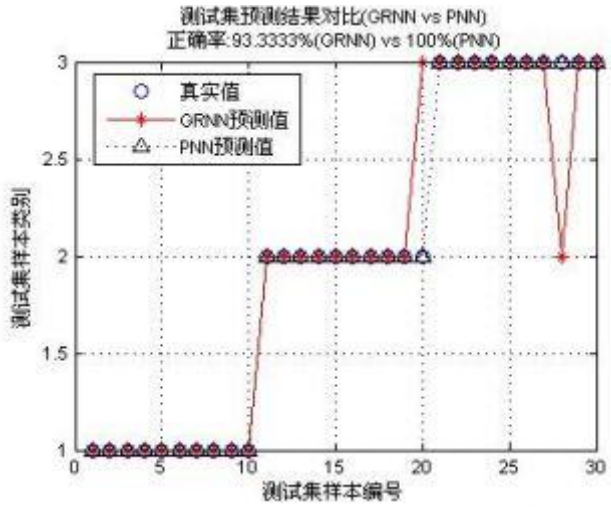
3、主程序：

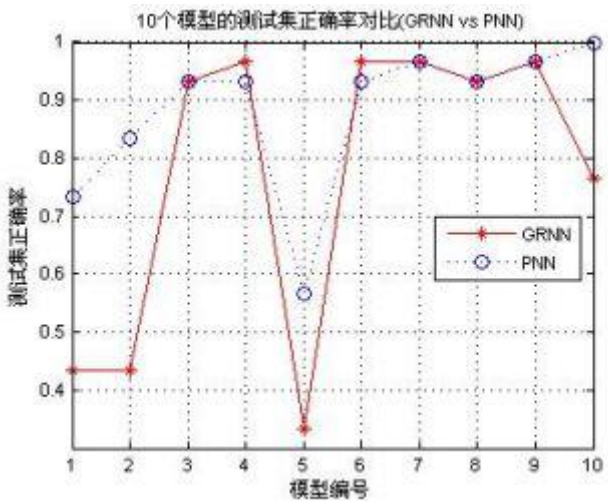
```
%% 清空环境变量
clear all
clc

%% 训练集/测试集产生

% 导入数据
load iris_data.mat
% 随机产生训练集和测试集
P_train = [];
T_train = [];
P_test = [];
T_test = [];
for i = 1:3
    temp_input = features((i-1)*50+1:i*50,:);
    temp_output = classes((i-1)*50+1:i*50,:);
    n = randperm(50);
    % 训练集——120 个样本
    P_train = [P_train temp_input(n(1:40),:)'];
    T_train = [T_train temp_output(n(1:40),:)'];
    % 测试集——30 个样本
    P_test = [P_test temp_input(n(41:50),:)'];
    T_test = [T_test temp_output(n(41:50),:)'];
end

%% 模型建立
result_grnn = [];
result_pnn = [];
time_grnn = [];
time_pnn = [];
```

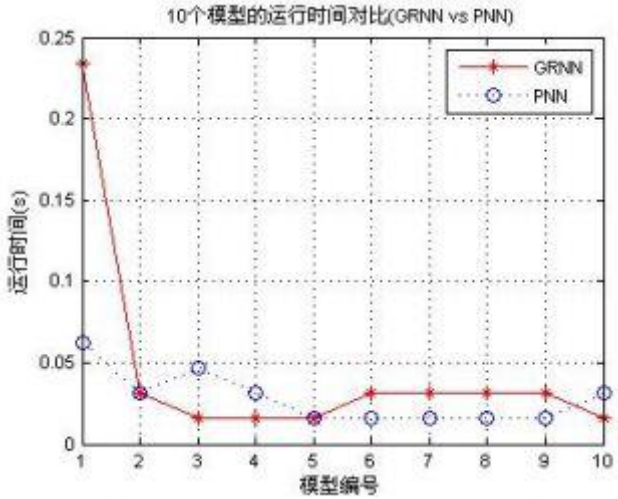
```
for i = 1:4
    for j = i:4
        p_train = P_train(i:j,:);
        p_test = P_test(i:j,:);
       %% GRNN 创建及仿真测试
        t = cputime;
        % 创建网络
        net_grnn = newgrnn(p_train,T_train);
        % 仿真测试
        t_sim_grnn = sim(net_grnn,p_test);
        T_sim_grnn = round(t_sim_grnn);
        t = cputime - t;
        time_grnn = [time_grnn t];
        result_grnn = [result_grnn T_sim_grnn'];
       %% PNN 创建及仿真测试
        t = cputime;
        Tc_train = ind2vec(T_train);
        % 创建网络
        net_pnn = newpnn(p_train,Tc_train);
        % 仿真测试
        Tc_test = ind2vec(T_test);
        t_sim_pnn = sim(net_pnn,p_test);
        T_sim_pnn = vec2ind(t_sim_pnn);
        t = cputime - t;
        time_pnn = [time_pnn t];
        result_pnn = [result_pnn T_sim_pnn'];
    end
end

%% 性能评价

% 正确率 accuracy
accuracy_grnn = [];
accuracy_pnn = [];
time = [];
for i = 1:10
    accuracy_1 = length(find(result_grnn(:,i) == T_test'))/length(T_test);
    accuracy_2 = length(find(result_pnn(:,i) == T_test'))/length(T_test);
    accuracy_grnn = [accuracy_grnn accuracy_1];
    accuracy_pnn = [accuracy_pnn accuracy_2];
end
% 结果对比
result = [T_test' result_grnn result_pnn]
accuracy = [accuracy_grnn;accuracy_pnn]
```

```
time = [time_grnn;time_pnn]
%% 绘图
figure(1)
plot(1:30,T_test,'bo',1:30,result_grnn(:,4),'r-*',1:30,result_pnn(:,4),'k:^')
grid on
xlabel('测试集样本编号')
ylabel('测试集样本类别')
string = {'测试集预测结果对比(GRNN vs PNN)';['正确率:' num2str(accuracy_grnn(4)*100)
'%(GRNN) vs ' num2str(accuracy_pnn(4)*100) '%(PNN)']};
title(string)
legend('真实值','GRNN 预测值','PNN 预测值')
figure(2)
plot(1:10,accuracy(1,:),'r-*',1:10,accuracy(2,:),'b:o')
grid on
xlabel('模型编号')
ylabel('测试集正确率')
title('10 个模型的测试集正确率对比(GRNN vs PNN)')
legend('GRNN','PNN')
figure(3)
plot(1:10,time(1,:),'r-*',1:10,time(2,:),'b:o')
grid on
xlabel('模型编号')
ylabel('运行时间(s)')
title('10 个模型的运行时间对比(GRNN vs PNN)')
legend('GRNN','PNN')
```

4、运行结果：







第 27 章　**无导师学习神经网络的分类——矿井突水水源判别**

1、案例背景

如第 25 章及第 26 章所述，对于有导师学习神经网络，事先需要知道与输入相对应的期望输出，根据期望输出与网络输出间的偏差来调整网络的权值和阈值。然而，在大多数情况下，由于人们认知能力以及环境的限制，往往无法或者很难获得期望的输出，在这种情况下，基于有导师学习的神经网络往往是无能为力的。

与有导师学习神经网络不同，无导师学习神经网络在学习过程中无需知道期望的输出。其与

真实人脑中的神经网络类似，可以通过不断地观察、分析与比较，自动揭示样本中的内在规律和本质，从而可以对具有近似特征（属性）的样本进行准确地分类和识别。本章将详细介绍竞争神经网络与自组织特征映射（SOFM）神经网络的结构及原理，并以实例说明其具体的应用范围及效果。

近年来，国内煤矿事故时有发生，严重危害了人们的生命和财产安全。其中，由于煤矿突水造成的事故不容忽视。因此，不少专家和学者致力于研究矿井突水事故的预防，突水水源的判别对预测矿井突水事故的发生有着重要的意义。

相关研究表明，可以利用水化学法判别矿井的突水水源，其基本依据是：由于受到含水层的沉积期、地层岩性、建造和地化环境等诸多因素的影响，使储存在不同含水层中的地下水主要化学成分有所不同。为了准确地判别突水水源，需要综合多种因素，用的比较多的是"7大离子"溶解氧、硝酸根离子等。

目前，有很多种判别突水水源的方法，如模糊综合评判、模糊聚类分析、灰色关联度法等，然而这些方法都要事先假定模式或主观规定一些参数，致使评价的结果主观性较强。

现采集到某矿的 39 个水源样本，分别来自于 4 个主要含水层：二灰和奥陶纪含水层、八灰含水层、顶板砂岩含水层和第四系含水层（砂砾石成分以石灰岩为主）。以每个水源样本中的 等 7 种离子的含量作为判别因素，试利用竞争神经网络和 SOFM 神经网络分别建立判别模型，并对模型的性能进行综合评价。

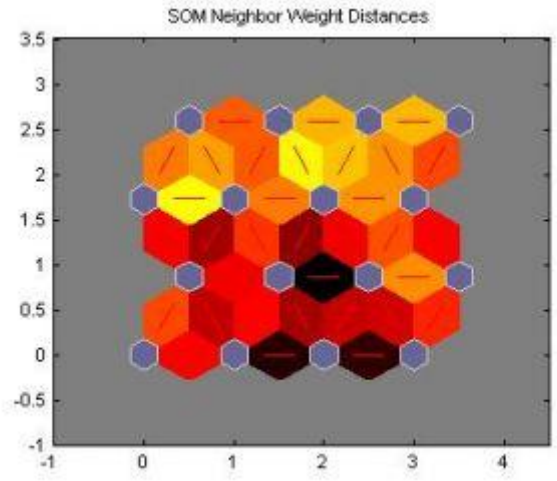
2、案例目录：

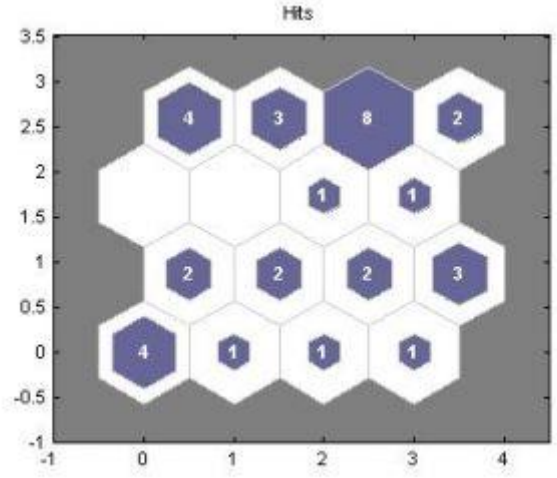
3、主程序：

```
%% 清空环境变量
clear all
clc

%% 训练集/测试集产生

% 导入数据
load water_data.mat
% 数据归一化
attributes = mapminmax(attributes);
% 训练集——35 个样本
P_train = attributes(:,1:35);
T_train = classes(:,1:35);
% 测试集——4 个样本
P_test = attributes(:,36:end);
T_test = classes(:,36:end);

%% 竞争神经网络创建、训练及仿真测试

% 创建网络
net = newc(minmax(P_train),4,0.01,0.01);
% 设置训练参数
net.trainParam.epochs = 500;
% 训练网络
net = train(net,P_train);
% 仿真测试
% 训练集
```

```
t_sim_compet_1 = sim(net,P_train);
T_sim_compet_1 = vec2ind(t_sim_compet_1);
% 测试集
t_sim_compet_2 = sim(net,P_test);
T_sim_compet_2 = vec2ind(t_sim_compet_2);

%% SOFM 神经网络创建、训练及仿真测试

% 创建网络
net = newsom(P_train,[4 4]);
% 设置训练参数
net.trainParam.epochs = 200;
% 训练网络
net = train(net,P_train);
% 仿真测试
% 训练集
t_sim_sofm_1 = sim(net,P_train);
T_sim_sofm_1 = vec2ind(t_sim_sofm_1);
% 测试集
t_sim_sofm_2 = sim(net,P_test);
T_sim_sofm_2 = vec2ind(t_sim_sofm_2);

%% 结果对比

% 竞争神经网络
result_compet_1 = [T_train' T_sim_compet_1']
result_compet_2 = [T_test' T_sim_compet_2']
% SOFM 神经网络
result_sofm_1 = [T_train' T_sim_sofm_1']
result_sofm_2 = [T_test' T_sim_sofm_2']
```

4、运行结果：



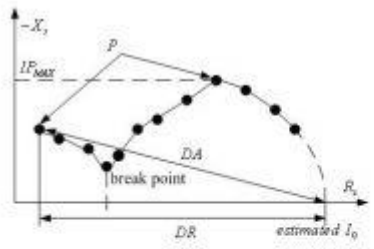


第 28 章　**支持向量机的分类——基于乳腺组织电阻抗特性的乳腺癌诊断**

1、案例背景

支持向量机（Support Vector Machine，SVM）是一种新的机器学习方法，其基础是 Vapnik 创建的统计学习理论（Statiscal Learning Theory，STL）。统计学习理论采用结构风险最小化（Structural Risk Minimization，SRM）准则，在最小化样本点误差的同时，最小化结构风险，提高了模型的泛化能力，且没有数据维数的限制。在进行线性分类时，将分类面取在离两类样本距离较大的地方；进行非线性分类时通过高维空间变换，将非线性分类变成高维

空间的线性分类问题。
本章将详细介绍支持向量机的分类原理，并将其应用于基于乳腺组织电阻抗频谱特性的乳腺癌诊断。
乳腺是女性身体的重要器官，乳腺疾病类别繁多、病因复杂，其中，乳腺癌是乳腺疾病的一种，逐渐成为危害女性健康的主要恶性肿瘤之一。近年来，乳腺癌等乳腺疾病发病率呈明显上升趋势，被医学界称为"女性健康第一杀手"。
相关研究结果表明，在直流状态下不同生物组织表现出不同的电阻特性，生物组织电阻抗随着外加电信号频率的不同而表现出较大的差异。常见的电阻抗测量方法有：电阻抗频谱法(Impedance Spectroscopy)、阻抗扫描成像法(Electrical Impedance Scanning，EIS)、电阻抗断层成像法（Electrical Impedance Tomography，EIT）等。电阻抗频谱法的测量依据是生物组织的电阻抗随着外加电信号频率的不同而呈现出较大的差异；阻抗扫描成像法的原理是癌变组织与正常组织及良性肿瘤组织的电导（阻）率相比，存在着显著性的差异，从而使得均匀分布在组织外的外加电流或电压场产生畸变；电阻抗断层成像法则利用设于体表外周的电极阵列及微弱测量电流，提取相关特征并重新构造出截面的电阻抗特性图像。
尽管目前的电阻抗测量结果还存在一些偏差，但相关研究已经证实癌变组织与正常组织的电阻抗特性存在显著地差异。因此，乳腺组织的电阻抗特征可以应用于乳腺癌的检查与诊断中。由于电阻抗测量法具有无创、廉价、操作简单、医生与病人易于接受等优点，随着测量技术的不断发展，电阻抗测量系统精度的日益提高，基于乳腺组织电阻抗特性的乳腺癌诊断技术势必会在临床检查与诊断中发挥其特有的作用。



2、案例目录：

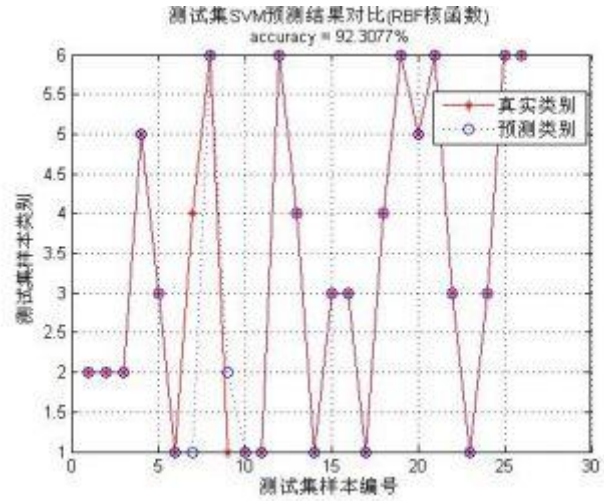
3、主程序：

```
%% 清空环境变量
clear all
clc

%% 导入数据
load BreastTissue_data.mat
% 随机产生训练集和测试集
% n = randperm(size(matrix,1));
load n.mat
```

```
% 训练集——80 个样本
train_matrix = matrix(n(1:80),:);
train_label = label(n(1:80),:);
% 测试集——26 个样本
test_matrix = matrix(n(81:end),:);
test_label = label(n(81:end),:);

%% 数据归一化
[Train_matrix,PS] = mapminmax(train_matrix');
Train_matrix = Train_matrix';
Test_matrix = mapminmax('apply',test_matrix',PS);
Test_matrix = Test_matrix';

%% SVM 创建/训练(RBF 核函数)

% 寻找最佳 c/g 参数——交叉验证方法
[c,g] = meshgrid(-10:0.2:10,-10:0.2:10);
[m,n] = size(c);
cg = zeros(m,n);
eps = 10^(-4);
v = 5;
bestc = 1;
bestg = 0.1;
bestacc = 0;
for i = 1:m
    for j = 1:n
        cmd = ['-v ',num2str(v),' -t 2',' -c ',num2str(2^c(i,j)),' -g ',num2str(2^g(i,j))];
        cg(i,j) = svmtrain(train_label,Train_matrix,cmd);
        if cg(i,j) > bestacc
            bestacc = cg(i,j);
            bestc = 2^c(i,j);
            bestg = 2^g(i,j);
        end
        if abs( cg(i,j)-bestacc )<=eps && bestc > 2^c(i,j)
            bestacc = cg(i,j);
            bestc = 2^c(i,j);
            bestg = 2^g(i,j);
        end
    end
end
cmd = [' -t 2',' -c ',num2str(bestc),' -g ',num2str(bestg)];
% 创建/训练 SVM 模型
model = svmtrain(train_label,Train_matrix,cmd);
```

```
%% SVM 仿真测试
[predict_label_1,accuracy_1] = svmpredict(train_label,Train_matrix,model);
[predict_label_2,accuracy_2] = svmpredict(test_label,Test_matrix,model);
result_1 = [train_label predict_label_1];
result_2 = [test_label predict_label_2];

%% 绘图
figure
plot(1:length(test_label),test_label,'r-*')
hold on
plot(1:length(test_label),predict_label_2,'b:o')
grid on
legend('真实类别','预测类别')
xlabel('测试集样本编号')
ylabel('测试集样本类别')
string = {'测试集 SVM 预测结果对比(RBF 核函数)';
          ['accuracy = ' num2str(accuracy_2(1)) '%']};
title(string)
```

4、运行结果：



第 29 章　**支持向量机的回归拟合——混凝土抗压强度预测**

1、案例背景

与传统的神经网络相比，SVM 具有以下几个优点：
（1）SVM 是专门针对小样本问题而提出的，其可以在有限样本的情况下获得最优解；
（2）SVM 算法最终将转化为一个二次规划问题，从理论上讲可以得到全局最优解，从而解决了传统神经网无法避免局部最优的问题；
（3）SVM 的拓扑结构由支持向量决定，避免了传统神经网络需要反复试凑确定网络结构的问题；
（4）SVM 利用非线性变换将原始变量映射到高维特征空间，在高维特征空间中构造线性分类函数，这既保证了模型具有良好的泛化能力，又解决了"维数灾难"问题。
同时，SVM 不仅可以解决分类、模式识别等问题，还可以解决回归、拟合等问题。因此，其在各个领域中都得到了非常广泛的利用。
本章将详细介绍 SVM 回归拟合的基本思想和原理，并以实例的形式阐述其在混凝土抗压强度预测中的应用。
随着技术的不断发展，混凝土抗压强度检测手段也愈来愈多，基本上可以分为局部破损法和非破损法两类，其中局部破损法主要是钻芯法，非破损法主要包括回弹法和超声法。工程上常采用钻芯法修正回弹法并结合《回弹法检测混凝土抗压强度技术规程》、《建筑结构检测技术标准》等规定的方法来推定混凝土的抗压强度。按照传统的方法，通常需要先对混凝土试件进行 28 天标准养护，然后通过测试获得。若能够提前预测出混凝土的 28 天抗压强度，则对于提高施工的质量和进度都具有重要的参考意义和实用价值。
此外，不少专家和学者将投影寻踪回归、神经网络、灰色理论等方法引入到混凝土结构工程领域中，取得了不错的效果，对混凝土抗压强度的预测有着一定的指导意义。
相关研究成果表明，混凝土的 28 天立方米抗压强度与混凝土的组成有很大的关系，即与每立方米混凝土中水泥、炉石、飞灰、水、超增塑剂、碎石及砂用量的多少有显著的关系。
现采集到 103 组混凝土样本的立方米抗压强度及其中上述 7 种成分的含量大小，要求利用支持向量机建立混凝土的 28 天立方米抗压强度与其组成间的回归数学模型，并对模型的性能进行评价。

2、案例目录：

3、主程序：

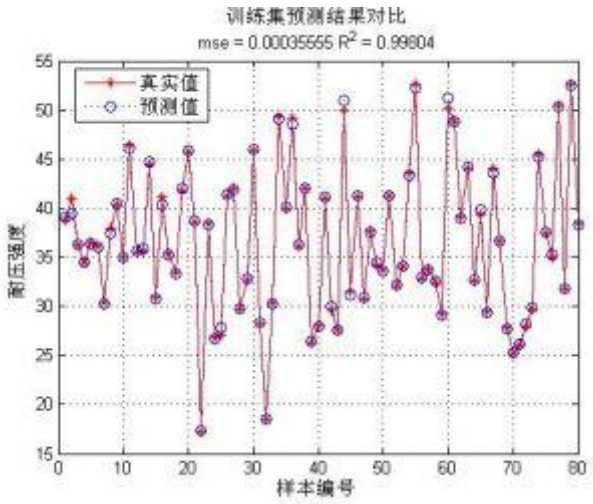
```
%% 清空环境变量
clear all
clc

%% 导入数据
load concrete_data.mat
% 随机产生训练集和测试集
n = randperm(size(attributes,2));
% 训练集——80 个样本
p_train = attributes(:,n(1:80))';
t_train = strength(:,n(1:80))';
% 测试集——23 个样本
p_test = attributes(:,n(81:end))';
t_test = strength(:,n(81:end))';

%% 数据归一化

% 训练集
[pn_train,inputps] = mapminmax(p_train');
pn_train = pn_train';
```

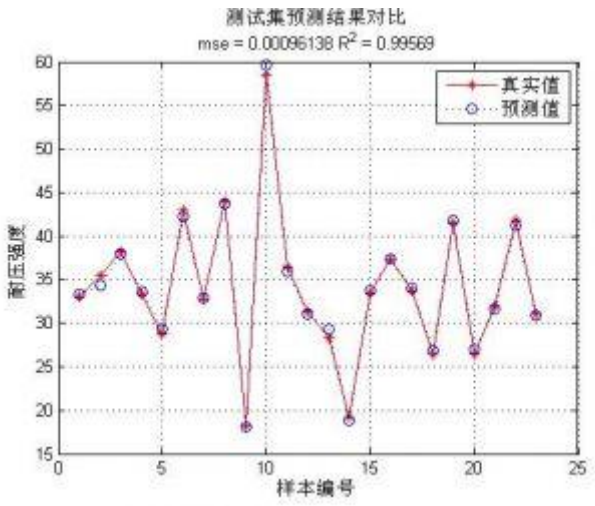
```
pn_test = mapminmax('apply',p_test',inputps);
pn_test = pn_test';
%  测试集
[tn_train,outputps] = mapminmax(t_train');
tn_train = tn_train';
tn_test = mapminmax('apply',t_test',outputps);
tn_test = tn_test';

%% SVM 模型创建/训练

%  寻找最佳 c 参数/g 参数
[c,g] = meshgrid(-10:0.5:10,-10:0.5:10);
[m,n] = size(c);
cg = zeros(m,n);
eps = 10^(-4);
v = 5;
bestc = 0;
bestg = 0;
error = Inf;
for i = 1:m
    for j = 1:n
        cmd = ['-v ',num2str(v),' -t 2',' -c ',num2str(2^c(i,j)),' -g ',num2str(2^g(i,j) ),' -s 3 -p 0.1'];
        cg(i,j) = svmtrain(tn_train,pn_train,cmd);
        if cg(i,j) < error
            error = cg(i,j);
            bestc = 2^c(i,j);
            bestg = 2^g(i,j);
        end
        if abs(cg(i,j) - error) <= eps && bestc > 2^c(i,j)
            error = cg(i,j);
            bestc = 2^c(i,j);
            bestg = 2^g(i,j);
        end
    end
end
%  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] = svmpredict(tn_train,pn_train,model);
[Predict_2,error_2] = svmpredict(tn_test,pn_test,model);
%  反归一化
predict_1 = mapminmax('reverse',Predict_1,outputps);
```

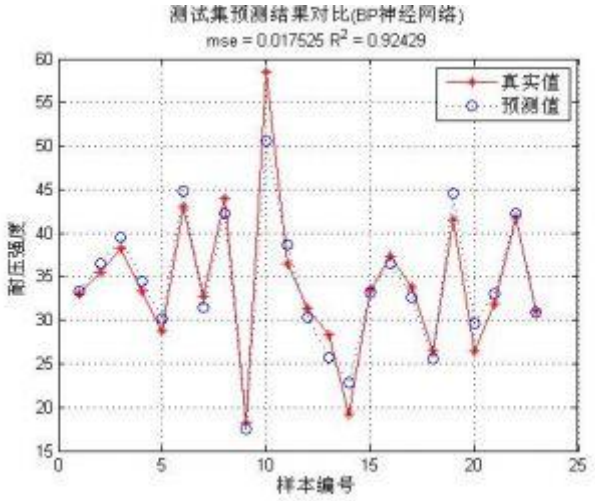
```
predict_2 = mapminmax('reverse',Predict_2,outputps);
% 结果对比
result_1 = [t_train predict_1];
result_2 = [t_test predict_2];

%% 绘图
figure(1)
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('耐压强度')
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(2)
plot(1:length(t_test),t_test,'r-*',1:length(t_test),predict_2,'b:o')
grid on
legend('真实值','预测值')
xlabel('样本编号')
ylabel('耐压强度')
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)

%% BP 神经网络

% 数据转置
pn_train = pn_train';
tn_train = tn_train';
pn_test = pn_test';
tn_test = tn_test';
% 创建 BP 神经网络
net = newff(pn_train,tn_train,10);
% 设置训练参数
net.trainParam.epcohs = 1000;
net.trainParam.goal = 1e-3;
net.trainParam.show = 10;
net.trainParam.lr = 0.1;
% 训练网络
net = train(net,pn_train,tn_train);
% 仿真测试
tn_sim = sim(net,pn_test);
% 均方误差
```

```
E = mse(tn_sim - tn_test);
% 决定系数
N = size(t_test,1);
R2=(N*sum(tn_sim.*tn_test)-sum(tn_sim)*sum(tn_test))^2/((N*sum((tn_sim).^2)-(sum(tn_sim))
^2)*(N*sum((tn_test).^2)-(sum(tn_test))^2));
% 反归一化
t_sim = mapminmax('reverse',tn_sim,outputps);
% 绘图
figure(3)
plot(1:length(t_test),t_test,'r-*',1:length(t_test),t_sim,'b:o')
grid on
legend('真实值','预测值')
xlabel('样本编号')
ylabel('耐压强度')
string_3 = {'测试集预测结果对比(BP 神经网络)';
            ['mse = ' num2str(E) ' R^2 = ' num2str(R2)]};
title(string_3)
```

4、运行结果：







第 30 章　极限学习机的回归拟合及分类——对比实验研究

1、案例背景

单隐含层前馈神经网络（Single-hidden Layer Feedforward Neural Network，SLFN）以其良好的学习能力在许多领域中得到了广泛的应用。然而，传统的学习算法（如 BP 算法等）固有的一些缺点，成为制约其发展的主要瓶颈。前馈神经网络大多采用梯度下降方法，该方法主要存在以下几个方面的缺点和不足：

（1）训练速度慢。由于梯度下降法需要多次迭代，从而达到修正权值和阈值的目的，因此训练过程耗时较长；

（2）容易陷入局部极小点，无法达到全局最小；

（3）学习率 的选择敏感。学习率 对神经网络的性能影响较大，必须选择合适的 ，才能获得较为理想的网络。若 太小，则算法收敛速度很慢，训练过程耗时较长；反之，若 太大，则训练过程可能不稳定（收敛）。

因此，探索一种训练速度快、获得全局最优解，且具有良好的泛化性能的训练算法是提升前馈神经网络性能的主要目标，也是近年来的研究热点和难点。

本章将介绍一个针对 SLFN 的新算法——极限学习机（Extreme Learning Machine，ELM），该算法随机产生输入层与隐含层间的连接权值及隐含层神经元的阈值，且在训练过程中无需调整，只需要设置隐含层神经元的个数，便可以获得唯一的最优解。与传统的训练方法相比，该方法具有学习速度快、泛化性能好等优点。

同时，在介绍 ELM 算法的基础上，本章以实例的形式将该算法分别应用于回归拟合（第 25 章——基于近红外光谱的汽油辛烷值预测）和分类（第 26 章——鸢尾花种类识别）中。

为了评价 ELM 的性能，试分别将 ELM 应用于基于近红外光谱的汽油辛烷值测定和鸢尾花种类识别两个问题中，并将其结果与传统前馈网络（BP、RBF、GRNN、PNN 等）的性能和运行速度进行比较，并探讨隐含层神经元个数对 ELM 性能的影响。

2、案例目录：

3、主程序：

```
%% Part1:ELM 的回归拟合——基于近红外光谱的汽油辛烷值预测
clear all
clc

%% 训练集/测试集产生
load spectra_data.mat
% 随机产生训练集和测试集
temp = randperm(size(NIR,1));
% 训练集——50 个样本
P_train = NIR(temp(1:50),:)';
T_train = octane(temp(1:50),:)';
% 测试集——10 个样本
P_test = NIR(temp(51:end),:)';
T_test = octane(temp(51:end),:)';
```

```
N = size(P_test,2);

%% 数据归一化

% 训练集
[Pn_train,inputps] = mapminmax(P_train);
Pn_test = mapminmax('apply',P_test,inputps);
% 测试集
[Tn_train,outputps] = mapminmax(T_train);
Tn_test = mapminmax('apply',T_test,outputps);

%% ELM 创建/训练
[IW,B,LW,TF,TYPE] = elmtrain(Pn_train,Tn_train,30,'sig',0);

%% ELM 仿真测试
tn_sim = elmpredict(Pn_test,IW,B,LW,TF,TYPE);
% 反归一化
T_sim = mapminmax('reverse',tn_sim,outputps);

%% 结果对比
result = [T_test' T_sim'];
% 均方误差
E = mse(T_sim - T_test);
% 决定系数
N = length(T_test);
R2=(N*sum(T_sim.*T_test)-sum(T_sim)*sum(T_test))^2/((N*sum((T_sim).^2)-(sum(T_sim))^2)*(
N*sum((T_test).^2)-(sum(T_test))^2));

%% 绘图
figure(1)
plot(1:N,T_test,'r-*',1:N,T_sim,'b:o')
grid on
legend('真实值','预测值')
xlabel('样本编号')
ylabel('辛烷值')
string = {'测试集辛烷值含量预测结果对比(ELM)';['(mse = ' num2str(E) ' R^2 = ' num2str(R2)
')']};
title(string)

%% Part2:ELM 的分类——鸢尾花种类识别
clear all
clc

%% 训练集/测试集产生
```

```
load iris_data.mat
%  随机产生训练集和测试集
P_train = [];
T_train = [];
P_test = [];
T_test = [];
for i = 1:3
    temp_input = features((i-1)*50+1:i*50,:);
    temp_output = classes((i-1)*50+1:i*50,:);
    n = randperm(50);
    %  训练集——120 个样本
    P_train = [P_train temp_input(n(1:40),:)'];
    T_train = [T_train temp_output(n(1:40),:)'];
    %  测试集——30 个样本
    P_test = [P_test temp_input(n(41:50),:)'];
    T_test = [T_test temp_output(n(41:50),:)'];
end

%% ELM 创建/训练
[IW,B,LW,TF,TYPE] = elmtrain(P_train,T_train,20,'sig',1);

%% ELM 仿真测试
T_sim_1 = elmpredict(P_train,IW,B,LW,TF,TYPE);
T_sim_2 = elmpredict(P_test,IW,B,LW,TF,TYPE);

%%  结果对比
result_1 = [T_train' T_sim_1'];
result_2 = [T_test' T_sim_2'];
%  训练集正确率
k1 = length(find(T_train == T_sim_1));
n1 = length(T_train);
Accuracy_1 = k1 / n1 * 100;
disp(['训练集正确率 Accuracy = ' num2str(Accuracy_1) '%(' num2str(k1) '/' num2str(n1) ')'])
%  测试集正确率
k2 = length(find(T_test == T_sim_2));
n2 = length(T_test);
Accuracy_2 = k2 / n2 * 100;
disp(['测试集正确率 Accuracy = ' num2str(Accuracy_2) '%(' num2str(k2) '/' num2str(n2) ')'])

%%  绘图
figure(2)
plot(1:30,T_test,'bo',1:30,T_sim_2,'r-*')
grid on
xlabel('测试集样本编号')
```

```
ylabel('测试集样本类别')
string = {'测试集预测结果对比(ELM)';['(正确率 Accuracy = ' num2str(Accuracy_2) '%)' ]};
title(string)
legend('真实值','ELM 预测值')
```

4、运行结果：