**一、具体思想**

遗传算法是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法。遗传算法是从代表问题可能潜在的解集的一个种群开始的，而一个种群则由经过基因编码的一定数目的个体组成。每个个体实际上是染色体带有特征的实体。染色体作为遗传物质的主要载体，即多个基因的集合，其内部表现（即基因型）是某种基因组合，它决定了个体的形状的外部表现，如黑头发的特征是由染色体中控制这一特征的某种基因组合决定的。因此，在一开始需要实现从表现型到基因型的映射即编码工作。由于仿照基因编码的工作很复杂，我们往往进行简化，如二进制编码，初代种群产生之后，按照适者生存和优胜劣汰的原理，逐代演化产生出越来越好的近似解，在每一代，根据问题域中个体的适应度大小选择个体，并借助于自然遗传学的遗传算子进行组合交叉和变异，产生出代表新的解集的种群。这个过程将导致种群像自然进化一样的后生代种群比前代更加适应于环境，末代种群中的最优个体经过解码，可以作为问题近似最优解。

1. **算法步骤**
2. 初始化，t=0，随机的产生一个包含L条不同染色体的种群M（0）；
3. 计算当前种群的M（t）中每一条染色体的适应度f（m）；
4. 按照选择概率p（f（m））对中群众的染色体进行采样，由采样出的染色体经过一定的操作繁殖出下一代染色体，组成下一代的种群M（t+1）；
5. 回到（2），直到达到终止条件，输出适应度最大的染色体最为找到的最优解。
6. **数据描述**

Sonar数据2类，60维，208个样本。

1. **实现过程**
2. **种群适应度函数**

在本次大作业中，我们有两种种群适应度函数的定义方式。

应为我们的目标是为了使得选出的样本特征实现更好的分类，所以最简单直接的方式便是将每一次结果直接进行knn分类，取分类结果为种群适应度函数。

还有一种方法是基于类别可分性判据来设计种群适应度函数。可供选择的有基于类内类间距离的类别可分性判据、基于概率的类别可分性判据以及基于熵的类别可分性判据。

我再此次实验中选择了基于类内类间距离的类别可分性判据，其定义主要如下：





其中为类内距离，为类间距离，为可分性判据。

1. **种群选择**  
   在本次实验中，我采用了轮盘赌的方法，来选择可以进入下一代种群的染色体。

又称比例选择方法．其基本思想是：各个个体被选中的概率与其适应度大小成正比。具体操作如下：

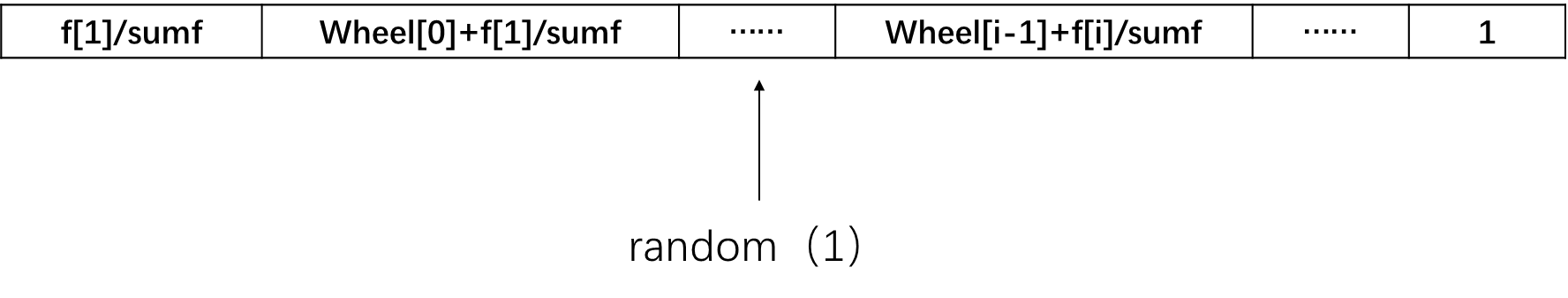
(1)计算出群体中每个个体的适应度f(i=1，2，…，M)，M为群体大小；

(2)计算出每个个体被遗传到下一代群体中的概率；

(3)计算出每个个体的累积概率；（q[i]称为染色体x[i] (i=1, 2, …, n)的积累概率）

(4)在[0，1]区间内产生一个均匀分布的伪随机数r；

(5)若r<q[1]，则选择个体1，否则，选择个体k，使得：q[k-1]<r≤q[k] 成立；

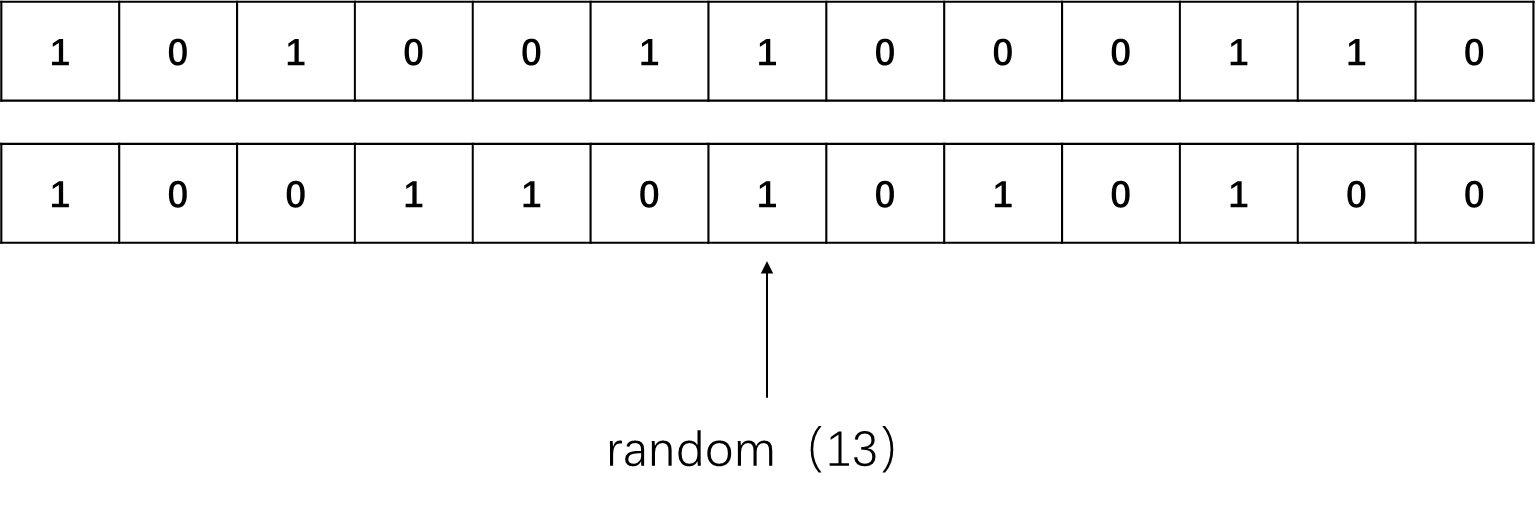


(6)重复(4)、(5)共M次

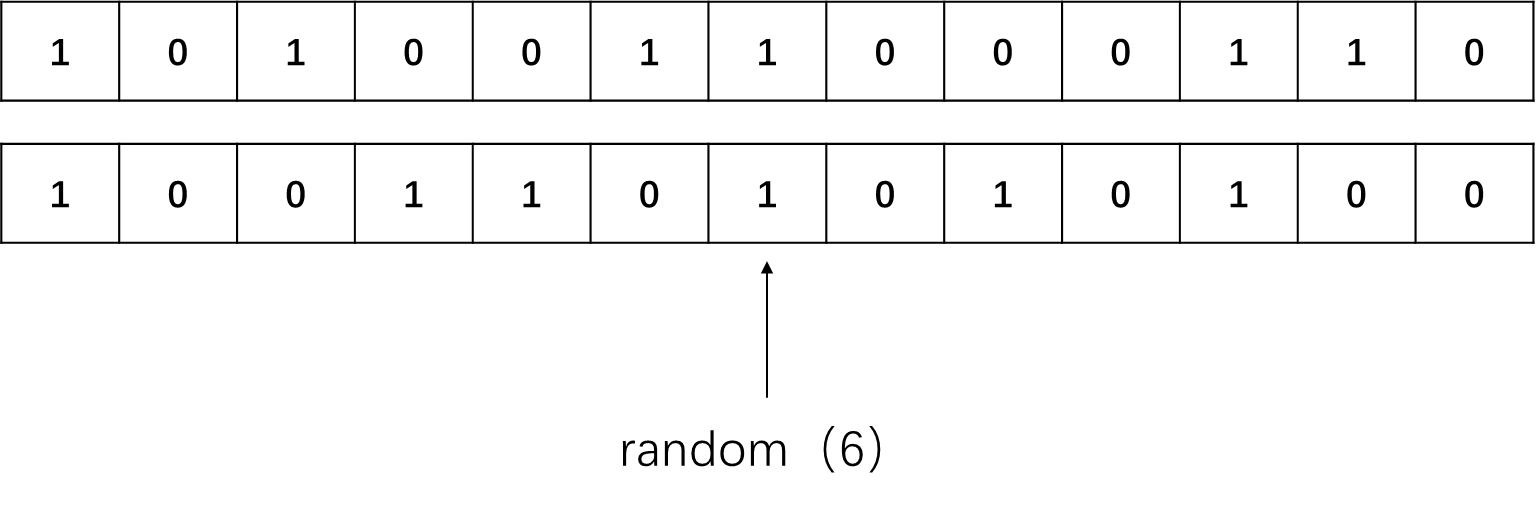
1. **基因重组**

在进行基因重组方式共有两种选择方案。

一是保持每次突变后选择的特征数目改变，这种方法较为简单，在染色体长度内随机产生一个随机数，从对应位置交换染色体片段即可。



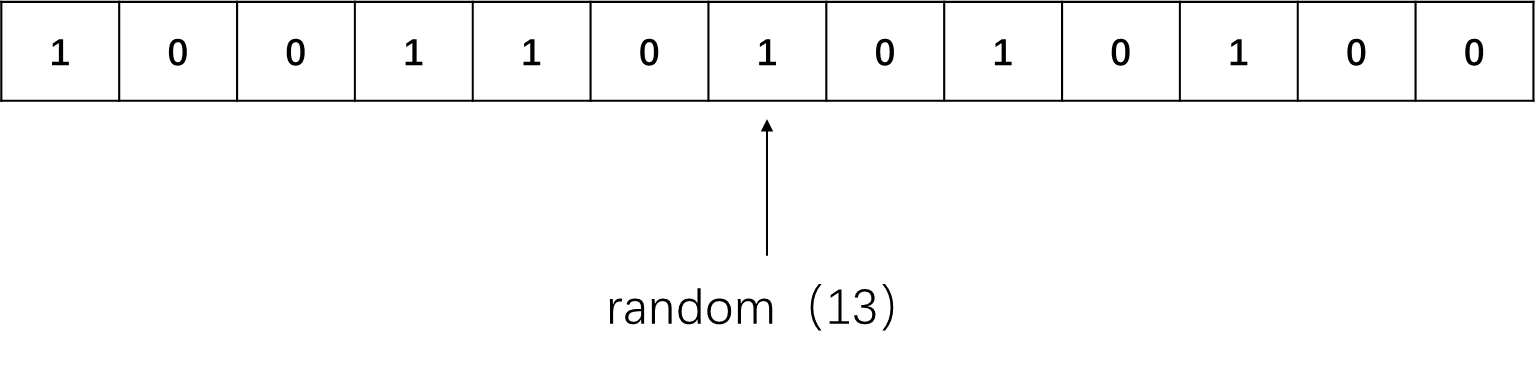
第二种方法则是保证重组前后选择特征数目不变。实现方法是先统计两条染色体中基因内容不同的位数n，然后再n内生成随机数，然后交换前n位不同的基因，来保证重组选择的特征数目不改变。



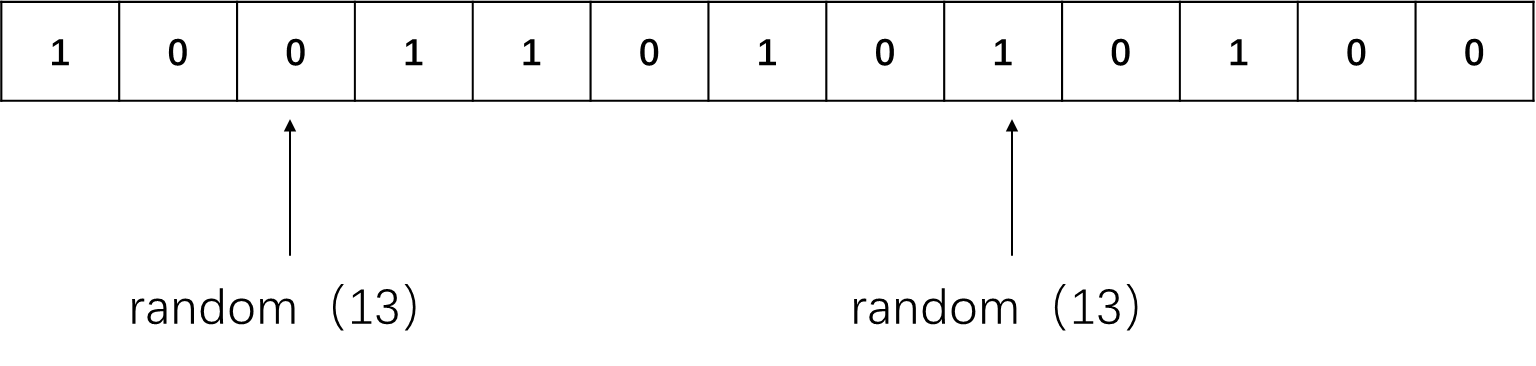
1. **基因突变**

基因突变同样对应两种实现方式。

当特征数目可以改变时，随机生成一个基因位置，该位取反即可。



当特征数目保持不变时，应先随机生成一个基因位置之后，再随机生成一个与其相反的基因位，然后同时取反。



1. **终止条件**

终止条件也有两种选择。

一是基于迭代次数，简单粗暴，但是可能由于不知道程序效果而耗费多余时间。

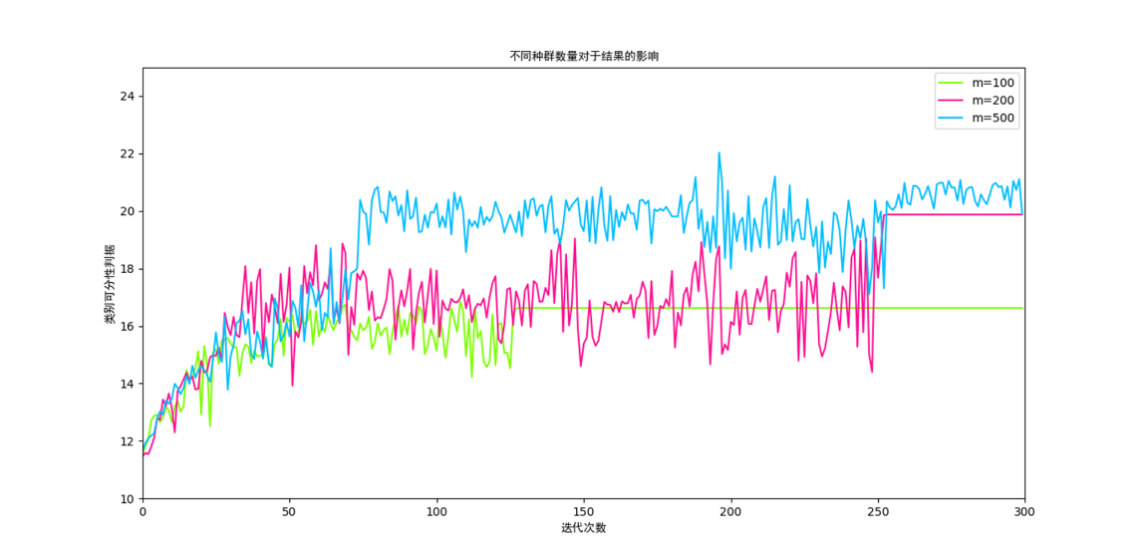
二是基于种群适应度函数，这种终止条件有可能产生对于适应度函数值所代表的实际意义概念不强的问题，所以需要先验知识。

为了更好的呈现实验结果，这次试验我采用的是基于迭代次数的终止条件。

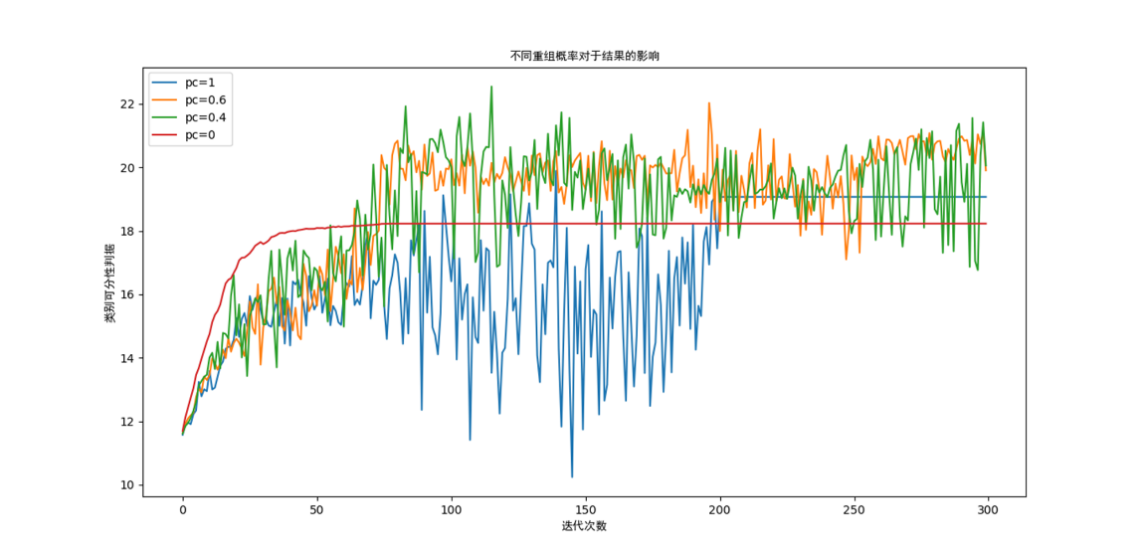
1. **实验结果**

本次实验的问题情境是60位特征中选择30个特征，默认种群数量为500，迭代次数300代，重组概率0.6，突变概率0.001，以下为在改变一个参数时种群的进化过程。

1. **种群数量发生改变时**



1. **重组概率发生改变时**



1. **结果分析**

由图可知，种群数量改变时，种群越小进化速度越快，收敛越快，越容易陷入局部最优解。种群越大进化速度越慢，收敛越慢，但容易趋向最优解。

重组概率改变时，重组概率越小，曲线越平滑，但进化效果差。重组概率越大，波动越大，参数过大时同样影响进化效果。

1. **源代码（python实现）**

import numpy as np

import random

#读入数据

def **readfile**():

global alldata,M,R

read=open(*'sonar.all-data'*)

data=[1]\*(alldata.shape[1]-1)

while True:

s=read.readline()

s=s.split(*','*)

if len(s) == 1:

break

data[0]=float(s[0])

for i in range(0,alldata.shape[1]-1):

data[i]=float(s[i])

if s[60]==*'R\n'*:

R=np.row\_stack((R, data))

if s[60]==*'M\n'*:

M=np.row\_stack((M, data))

alldata=np.row\_stack((alldata, s))

read.close()

#返回该DNA链的基于类内类间距离的类别可分性判据

def **getf**(l):

#生成特征向量

Xm=np.zeros(shape=(0,D))

Xr=np.zeros(shape=(0,D))

Mr=[0]\*D

Mm=[0]\*D

for i in range(0,Nr):

temp=[]

for j in range (0,L\_LENGTH):

if l[j]==1:

temp.append(R[i][j])

Xr=np.row\_stack((Xr,temp))

Mr=np.add(Mr,temp)

for i in range(0,Nm):

temp=[]

for j in range (0,L\_LENGTH):

if l[j]==1:

temp.append(M[i][j])

Xm=np.row\_stack((Xm,temp))

Mm=np.add(Mm,temp)

#计算类别可分性判别函数

Mr=np.multiply(1/Nr,Mr)

Mm=np.multiply(1/Nm,Mm)

Mall=np.multiply(Pr,Mr)+np.multiply(Pm,Mm)

Sb=np.multiply(Pr,np.dot(np.mat(Mr-Mall),np.mat(Mr-Mall).T))+np.multiply(Pm,np.dot(np.mat(Mm-Mall),np.mat(Mm-Mall).T))

Sr=np.zeros(shape=(D,D))

Sm=np.zeros(shape=(D,D))

for i in range(0,Nr):

Sr=Sr+np.dot(np.mat(Xr[i]-Mr),np.mat(Xr[i]-Mr).T)

Sr=np.multiply(1/N,Sr)

for i in range(0, Nm):

Sm=Sm+np.dot(np.mat(Xm[i]-Mm),np.mat(Xm[i]-Mm).T)

Sm=np.multiply(1/N,Sm)

Sw=Sr+Sm

return np.trace(np.mat(Sb+Sw))

#产生新的染色体

def **born**(l1,l2):

pb=random.random()

#突变过程

if(pb<pm):

i=random.randint(0,L\_LENGTH)

j=i

while(l1[j]==l1[i]):

i=random.randint(0,L\_LENGTH)

l1[i]=l1[i]^1

l1[j]=l1[j]^1

i = random.randint(0, L\_LENGTH)

j = i

while (l2[j]==l2[i]):

i=random.randint(0, L\_LENGTH)

l2[i]=l2[i]^1

l2[j]=l2[j]^1

if(pb>pc):

return l1,l2

#重组过程

else:

double\_1=0

singl\_0=[-1]\*L\_LENGTH

singl\_1=[-1]\*L\_LENGTH

for i in range (0,L\_LENGTH):

if l1[i]==1 and l2[i]==1:

double\_1=double\_1+1

elif l1[i]==0 and l2[i]==1:

singl\_0[i]=i

elif l1[i]==1 and l2[i]==0:

singl\_1[i]=i

exchange\_1=random.randint(0,(D-double\_1))

cot=0

for i in range(0,L\_LENGTH):

if singl\_0[i]!=-1:

l1[i]=1

l2[i]=0

cot=cot+1

if cot==exchange\_1:

break

cot=0

for i in range(L\_LENGTH-1,-1,-1):

if singl\_1[i]!=-1:

l1[i]=0

l2[i]=1

cot=cot+1

if cot==exchange\_1:

break

return l1,l2

#GA

#读入数据与初始化变量

alldata=np.zeros(shape=(0, 61)) #全体数据带标签

R=np.zeros(shape=(0,60)) #R

M=np.zeros(shape=(0,60)) #M

readfile()

M\_SIZE=1000 #种群容量

L\_LENGTH=alldata.shape[1]-1 #染色体长度

D=20 #提取的特征数目

pc=0.6 #重组概率

pm=0.001 #突变概率

T=300 #迭代次数

N=alldata.shape[0] #全体样本个数

Nr=R.shape[0] #R样本个数

Nm=M.shape[0] #M样本个数

Pr=Nr/N #R先验概率

Pm=Nm/N #M先验概率

ansf=0 #最优可分性判据

ansl=[] #最优解

F=[] #可分性判据

#随机产生初始种群

m=[]

for i in range(0,M\_SIZE):

#随机产生一条染色体

l=[0]\*L\_LENGTH

cot=0;

while cot<D:

index=random.randint(0,L\_LENGTH-1)

if l[index]==0:

l[index]=1

cot=cot+1

m.append(l)

t=0

while t<T:

t=t+1

#print("The %d-th generation:"%(t))

#计算当前种群每一条染色体的种群适应度并得到最大的适应度

f=[0]\*M\_SIZE

sumf=0

maxf=-1

maxi=-1

for i in range(0,M\_SIZE):

f[i]=getf(m[i])

sumf=sumf+f[i]

if f[i]>maxf:

maxf=f[i]

maxi=i

if (maxf>ansf):

ansf=maxf

ansl=m[i]

print(*"%f,%f"* % (maxf, sumf / M\_SIZE))

F.append(sumf / M\_SIZE)

#计算当前种群每一条染色体的选择概率及概率轮盘

max=0

min=10

p=[0]\*M\_SIZE

for i in range(0,M\_SIZE):

p[i]=f[i]/sumf

if p[i]>max:

max=p[i]

if p[i]<min:

min=p[i]

wheel=[0]\*(M\_SIZE+1)

for i in range(1,M\_SIZE+1):

wheel[i]=wheel[i-1]+p[i-1]

#print(max)

#print(min)

#产生新的种群

newm=[0]\*M\_SIZE

i=0;

while i<M\_SIZE:

#按照p(f(l))对种群中的染色体进行采样

temp=random.random()

for j1 in range(1,M\_SIZE+1):

if temp>wheel[j1-1] and temp<=wheel[j1]:

break

temp=random.random()

for j2 in range(1,M\_SIZE+1):

if temp>wheel[j2-1] and temp<=wheel[j2]:

break

#繁殖下一代

newl1,newl2=born(m[j1-1],m[j2-1])

newm[i]=newl1

i=i+1

newm[i]=newl2

i=i+1

m=newm

print(*"The result of selection:"*,ansl)