

**机器学习第十四次实验报告**

**遗传算法**

**组 名：重 案 六 组**

**姓 名：李阳帆 吴楠楠 申丹丹**

**任课老师：冯 筠（教 授）**

**分工情况：共 同 合 作 完 成**

**二○一七年六月**

# 遗传算法

## 问题描述

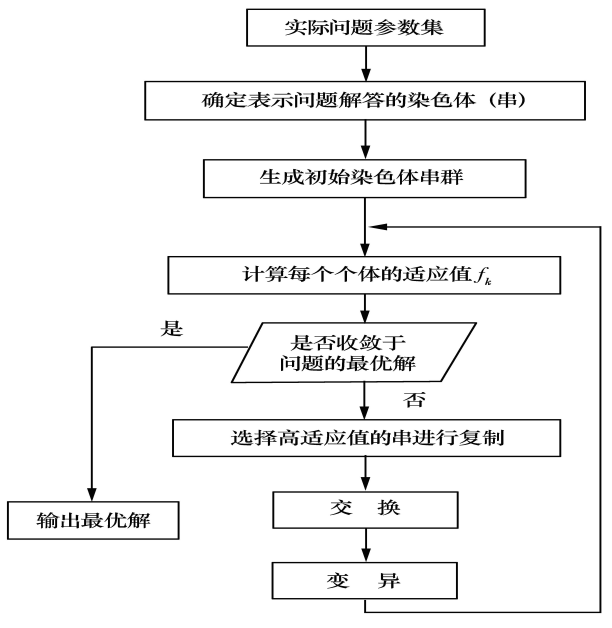
利用遗传算法，在软木塞数据集上选择出分类准确率最高的几维特征，并用分类器验证准确率。

要求：确定编码方案，适应度函数，选择，遗传，交叉的方法

## 模型建立和求解过程

### 2.1 遗传算法

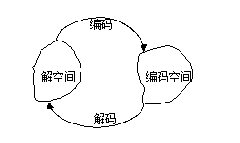
1. 遗传算法的定义：它是一类借鉴生物界的进化规律（适者生存，优胜劣汰遗传机制）演化而来的启发式算法。
2. 遗传算法的五个基本要素：
3. 参数编码：把一个问题的可行解从其解空间转换到遗传算法的搜索空间的转换方法。
4. 初始群体的设定：种群和规模的选择
5. 适应度函数的设计
6. 遗传操作设计：选择、交叉、变异
7. 控制参数设定：进化代数、交叉概率、变异概率
8. 遗传算法的流程图



3.编码与解码

要实现遗传算法首先需要弄清楚如何对求解问题进行编码和解码。对于函数优化问题，一般来说，有两种编码方式，一是实数编码，一是二进制编码，两者各有优缺点，二进制编码具有稳定性高、种群多样性大等优点，但是需要的存储空间大，需要解码过程并且难以理解；而实数编码直接用实数表示基因，容易理解并且不要解码过程，但是容易过早收敛，从而陷入局部最优。本文以最常用的二进制编码为例，说明遗传编码的过程。

从遗传算法求解的过程来看，需要处理好两个空间的问题，一个是编码空间，另一个是解空间，如下图所示。从解空间到编码空间的映射过程成为编码过程；从编码空间到解空间的映射过程成为解码过程。



1. 二进制编码

用若干二进制数表示一个个体（解），将原问题的解空间映射到位串空间 B={0，1}上，然后在位串空间上进行遗传操作。

假设某一参数的取值范围是，用长度为的二进制编码符号串来表示该参数，则它总共能产生种不同的编码。二进制编码的公式如（1）式，其中为二进制编码的编码精度。

 （1）

1. 解码

假设某一个体的编码是，则对应的解码公式为：

 （2）

4.初始化种群

在开始遗传算法迭代过程之前，需要对种群进行初始化。设种群大小为pop\_size，每个染色体或个体的长度为chromo\_size，种群的大小决定了种群的多样性，而染色体的长度则是由前述的编码过程决定的。一般随机生成初始种群，但是如果知道种群的实际分布，也可以按照此分布来生成初始种群。假设生成的初始种群为(v1, v2, …, vpop\_size)。

在软木塞数据中，我们首先读取数据，设置种群大小pop\_size=10， 染色体长度chromo\_size=10，对数据进行初始化分。划分后的pop矩阵只包含0和1，关键代码如下：

%初始化种群

for i=1:pop\_size

for j=1:chromo\_size

pop(i,j) = round(rand);

end

end

1. 适应度函数：

适应度用于评价个体的优劣程度,适应度越大个体越好,反之适应度越小则个体越差；根据适应度的大小对个体进行选择,以保证适应性能好的个体有更多的机会繁殖后代,使优良特性得以遗传。因此,遗传算法要求适应度函数值必须是非负数。

在本次实验中，我们采用KNN函数作为适应度函数，将分类的准确率归一化后作为适应度值，关键代码如下：

%计算适应度

for j =1:size(Test\_Data,1)

for i=1:pop\_size

label(:,i) = knnclassify(Test\_Data(:,[find(pop(i,:)==1)]),Train\_Data(:,[find(pop(i,:)==1)]),Train\_label,5,'euclidean','nearest');

end

end

for i = 1 :pop\_size

count(i) = sum(Test\_label(:,1)==label(:,i));

value(i,1) = count(i)/size(Test\_Data,1);

end

%计算适应度值

for i=1:pop\_size

fitness\_value(i,1) = 0;

end

%对求得的准确率进行归一化

for i = 1 :pop\_size

fitness\_value(i,1) = value(i,1)/sum(value);

end

然后，对个体按照适应度大小进行排序，并且保存最佳个体。

6.选择操作

选择操作即从前代种群中选择个体到下一代种群的过程。一般根据个体适应度的分布来选择个体。以初始种群(v1, v2, …, vpop\_size)为例，假设每个个体的适应度为(fitness(v1), fitness(v2),…, fitness(vpop\_size))，一般适应度可以按照解码的过程进行计算。以轮盘赌的方式选择个体。

随机转动一下轮盘，当轮盘停止转动时，若指针指向某个个体，则该个体被选中。很明显，具有较高适应度的个体比具有较低适应度的个体更有机会被选中。但是这种选择具有随机性，在选择的过程中可能会丢失掉比较好的个体，所以可以使用精英机制，将前代最优个体直接选到下一代中。

轮盘赌选择具体算法如下：

for i=1:pop\_size

r = rand \* fitness\_table(pop\_size); %随机生成一个随机数，在0和总适应度之间，因为fitness\_table(pop\_size)为最后一个个体的累积适应度，即为总适应度 first = 1;

last = pop\_size;

mid = round((last+first)/2);

idx = -1;

%下面按照排中法选择个体

while (first <= last) && (idx == -1)

if r > fitness\_table(mid)

first = mid;

elseif r < fitness\_table(mid)

last = mid;

else

idx = mid;

break;

end

mid = round((last+first)/2);

if (last - first) == 1

idx = last;

break;

end

end

for j=1:chromo\_size

pop\_new(i,j)=pop(idx,j);

end

end

%是否精英选择

if elitism

p = pop\_size-1;

else

p = pop\_size;

end

for i=1:p

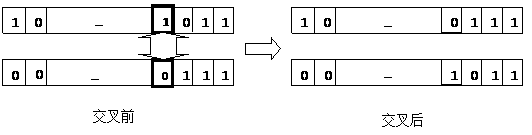
for j=1:chromo\_size

pop(i,j) = pop\_new(i,j); %若是精英选择，则只将pop\_new前pop\_size-1个个体赋给pop，最后一个为前代最优个体保留

end

end

7.交叉操作是对任意两个个体进行的（在这里我们实现的算法是直接对相邻的两个个体进行的）。随机选择两个个体，如下图所示：



然后随机生成一个实数0<=r<=1, 如果r<cross\_rate， 0<cross\_rate<1为交叉概率，则对这两个个体进行交叉，否则则不进行。如果需要进行交叉，再随机选择交叉位置(rand\*chromo\_size)，如果等于0或者1，将不进行交叉。否则将交叉位置以后的二进制串进行对换（包括交叉位置）。（注意：有时候还可以进行多点交叉，但是这里只讨论单点交叉的情况）

单点交叉具体代码如下：

for i=1:2:pop\_size

if(rand < cross\_rate) %cross\_rate为交叉概率

cross\_pos = round(rand \* chromo\_size); %交叉位置

if or (cross\_pos == 0, cross\_pos == 1) %交叉位置为0或1，不进行交叉

continue;

end

for j=cross\_pos:chromo\_size %交换

temp = pop(i,j);

pop(i,j) = pop(i+1,j);

pop(i+1,j) = temp;

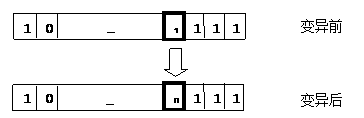
end

end

end

1. 变异操作

变异操作是对单个个体进行的。首先生成一个随机实数0<=r<=1, 如果r<mutate\_rate，则对此个体进行变异操作， 0<mutate\_rate<1为变异概率，一般为一个比较小的实数。对每一个个体，进行变异操作，如下图所示：



如个体需要进行变异操作，首先需要确定变异位置(rand\*chromo\_size)，若为0则不进行变异，否则则对该位置的二进制数字进行变异：1变成0, 0变成1.（当然也可以选择多点进行变异）。单点变异的具体代码如下：

for i=1:pop\_size

if rand < mutate\_rate %mutate\_rate为变异概率

mutate\_pos = round(rand\*chromo\_size); %变异位置

if mutate\_pos == 0

continue; %若变异位置为0，则不进行变异

end

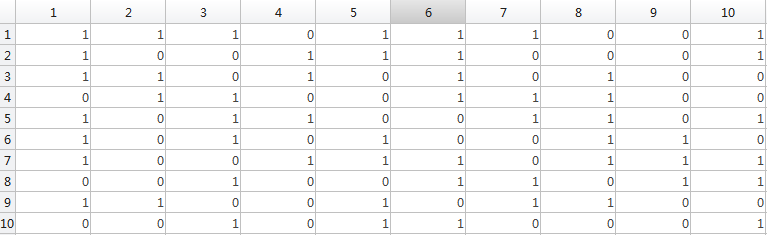
pop(i,mutate\_pos) = 1 - pop(i, mutate\_pos); %将变异位置上的数字置反

end

end

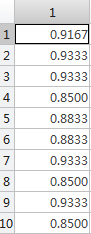
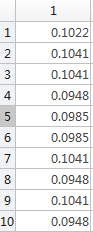
### 2.2 实验结果

1. **初始化群体**：随机生成10\*10的0\1特征矩阵



1. **适应度函数**

根据10\*10的0\1矩阵，我们选择特征，如第一行选择值为1的特征，即选择特征1、2、4，利用KNN进行分类得到的准确率作为适应度值，此处的适应度函数为KNN函数，再将适应度值进行归一化，可以得到fitness\_value。如下表所示，其中左边为KNN分类的准确率右边为归一化后的fitness\_value：

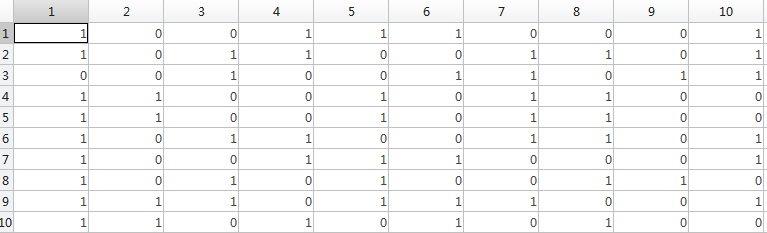
1. **排序**

对个体按适应度大小进行排序，并且保存最佳个体，将其保存在best\_individual中，将fitness\_value中的值，累加求和，得到fitness\_table，如下表所示，每个概率值组成一个区域，全部概率值之和为1，它将作为选择复制中的转盘用于选择个体。



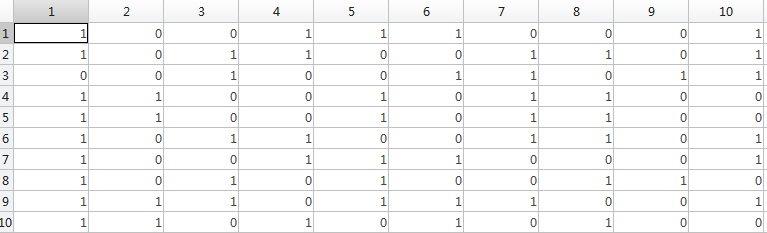
1. **选择**

随机产生一个0—1之间的随机数，依据该随机数出现在上述的哪一个概率区域内来确定各个个体出现在上述哪一个概率区域内，来确定各个个体被选中的次数。这里，我们采用了排中法选择个体，选择完毕后，会得到一个最新的编码矩阵pop\_new，另外，我们在代码中加入了精英判断，若是精英选择，则只将pop\_new前pop\_size-1个个体赋给pop，最后一个为前代最优个体保留，其中选择完毕后的种群矩阵pop\_new如下表所示，考虑到保存前一代的最优个体后，得到的种群矩阵pop。



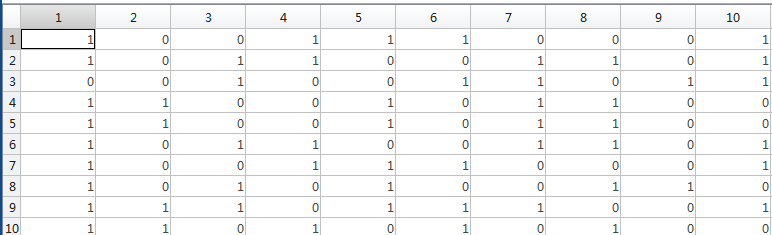
1. **交叉**

本实验采用单点交叉法，它首先对群体进行随机配对，其次随机设置交叉点位置，最后再相互交换配对染色体之间的部分基因。设置cross\_rate=0.6为交叉概率，若交叉位置都为0或1，则不进行交叉，反之，1—>0，0—>1。



1. **变异**

本实验采用单点变异操作，变异操作是对单个个体进行的。最后变异生成的种群矩阵pop。



**7.特征选择结果：**

1次迭代得到的最佳特征选择集合为：



迭代10次后得到最优的特征选择集合如下表所示：



迭代10次后，适应度函数的准确率为0.9333，达到稳定状态，此时的特征选择集合为最好的解。



## 实验结论

通过本次实验，了解到遗传算法得到的并不是一个最优解，它是对未知问题的模拟，为了防止过早的收敛得不到很好的解，我们在选择程序中加入了精英选择判断，把每一次得到的最好的解放入下一次迭代中。虽然我们找出的不一定是最优解，但一定是最好的解。遗传算法可以解决浩瀚空间中搜索的问题，以及局部最优的问题。当然，遗传算法还有很多应用，本实验只是将遗传算法应用到了特征选择。