一个关于遗传算法的java小实验（吃豆人）

原创

飚蕭蕭 于 2018-09-26 22:56:09 发布 4302 收藏 8

分类专栏： # 机器学习 文章标签： 遗传算法 GA java

我们不生产知识，我们只是互联网的搬运工

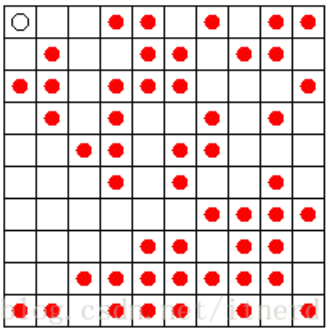
本文链接：https://blog.csdn.net/itnerd/article/details/82860638

遗传算法就是利用遗传学中父代基因杂交产生子代的原理，通过良种配对、遗传进化来改良子代的算法。

遗传算法非常适合用来处理离散型的**决策问题**，通过对父代中优良模型的参数进行杂交，产生下一代模型，淘汰其中效果较差的杂种，保留表现优良的杂种，重复此过程，直到培育出满意的模型。

遗传算法的关键不仅在于良种配对，更在于杂交过程中随机出现的**少量突变**，正是由于这种偏离父代的突变存在使得子代有机会比父代更强！始终群进一步演化，避免停滞不前！

下面用一个简单吃豆人实验来说明。源码在这里！



上图展示的是实验场景，可以看成“吃豆人”，或者在有些地方称为“扫地机器人”。**黑色空心圆**代表机器人，它的目标是在有限步数内，消灭网格中更多的**红色实心圆**。

> 机器人可以执行的动作有7种，分别是 *原地不动*、*向左*、*向右*、*向下*、*向上*、*随机移动*、*吃豆（捡垃圾）*。
>
> 机器人的视野只有5格，即 *东西南北中*。
>
> 游戏的规则为：撞墙**扣5分**，吃豆（前提是当前位置有豆）**得10分**、瞎jb吃**扣1分**。

图中展示的是一个利用遗传算法，从模型参数为随机初始化的祖先，在规模为100的种群中，择优杂交繁衍约1000代左右得到的一个优良品种。看它那沉着稳健的步伐，还是相当OK的。

下面来说说具体实现的步骤吧！具体代码（训练模型+图形界面）可以在这里下载！

首先，模型中定义了这些属性+常量：

```
public class Model {
 private int rowNum,colNum; // 场地的行数、列数
 private int currX,currY;   // 机器人的当前位置
 private boolean fruits[][]; // 判断每个位置是否有豆子的布尔矩阵
 private double percent;    // 每个位置有豆子的概率
 private String strategy;   // 机器人在每个位置对应的状态下采取的策略
 public static final int STRATEGY_LEN = 243;  // 策略的字符串长度
 public static final int ACTION_NUM = 7;      // 机器人的动作数
 public static final int DIRECTION[][] = {{0,0},{0,1},{0,-1},{1,0},{-1,0}}; // 不动、东、西、南、北
 ……
}
```

如何定义机器人当前的状态？ 机器人的视野只有5格：*东*、*西*、*南*、*北*、*中*。每一格可以有3中状态：*墙*、*豆*、*空*。所以机器人的所有可能状态有

$3 = 243$ 种。事实上，并不是所有状态都存在，比如当前位置有墙、或者四面都是墙等等。但这并不影响算法

你可能发现了上面模型中定义的策略String的长度也是243：

```
public static final int STRATEGY_LEN = 243;  // 策略的字符串长度
```

实际上每个字符都在{0,1,2,3,4,5,6}中取值，对应机器人在那个状态下的动作！

```
* 0-不动
* 1-东
* 2-西
* 3-南
* 4-北
* 5-随机移动
* 6-吃豆
```

这是判断机器人当前状态的函数，返回一个0~242之间的数：

```
public int getState(){
 int tempX,tempY;
 int state,wholeState=0;
 for(int i=0; i<5;++i) {  // 遍历视野中的 5 个格子
  tempX = currX + DIRECTION[i][0];
  tempY = currY + DIRECTION[i][1];
  if(tempX<0 || tempY<0 || tempX>=rowNum || tempY>=colNum) {
   state = 0;   //如果是 墙（出界）
  }else {
   if(fruits[tempX][tempY]) {
    state = 1; //发现豆子
   }else {
    state = 2;  //什么也没有
   }
  }
  wholeState = 3*wholeState+state;  // 乘 3相当于在 3进制中进位，用位数为5的3进制数表示当前状态
 }
 return wholeState;
}
```

获取当前策略下的动作（intent），之所以成为intent是因为有些动作是不会被执行的（撞墙、瞎吃）。

```
public int getIntent(int state) {
 return strategy.charAt(state)-'0';
}
```

然后根据intent进行动作并得到每一步的分数

```
public int checkAndMove(int intent) {
 int score = 0;
 int tempX,tempY;
 Random rand = new Random();
 if(intent == 5) { //随机移动，等价于转化为动作1-4之一
  intent = rand.nextInt(4)+1; // 5==>{1,2,3,4}
 }
 if(intent == 6) { // 吃豆
  if(fruits[currX][currY]) { // 吃到豆
   score = 10;
   fruits[currX][currY] = false;
  }else {      // 瞎吃
   score = -1;
  }
 }else{
  tempX = currX + DIRECTION[intent][0];
  tempY = currY + DIRECTION[intent][1];
  if(tempX<0 || tempY<0 || tempX>=rowNum || tempY>=colNum) { // 撞墙
   score = -5;
  }else {
   currX = tempX;
   currY = tempY;
  }
 }
 return score;
}
```

这样一来，我们已经完成**根据当前位置判断状态、根据状态获得动作、处理动作并得分**这几个基本模块。只要任意初始化一个Strategy，就可以通过规定步数内的得分来评估它的好坏！

终于来到重要一步，遗传进化！遗传进化需要考虑两个方面：1. 继承父辈的优良性状；2. 有所创新/突变。也就是人们常说的在**探索**和**开发**中保持平衡！

```
public static String combineStrategy(String a,String b) {
 StringBuilder sb = new StringBuilder();
 Random rand = new Random();
 if(rand.nextInt(10)<1) { // 随机劈腿！！！
  b = genStrategy();
 }
 for(int i=0; i<STRATEGY_LEN; ++i) {
  if(rand.nextInt(20)<1) { // 杂交出错！！！
   sb.append((char)(rand.nextInt(ACTION_NUM)+'0'));
  }else {
   sb.append(rand.nextBoolean()?a.charAt(i):b.charAt(i)); // 正常杂交
  }
 }
 return sb.toString();
}
```

笔者在实验中发现，如果没有 *劈腿*、*出错* 这种事情发生，种群的基因会迅速的趋同，因为暂时最好的那些杂种凭借一时领先，获得垄断地位的交配机会，种群便永久地陷入半吊子水平，不得进步！！这有些类似神经网络在训练中陷入局部最优的烂坑！

可见**出错是进化的关键**！

接下来的事情就简单了（其实都很简单）：

1. 创建一批随机祖先，即随机初始化一批（100个） Model 的 strategy，让他们在10*10且豆子概率为50%的网格中放飞自我；
2. 挑选出最优的一批Model进行杂交；
3. 评估子代Model的得分；
4. 重复2、3，直到结果满意。

实际实验中，1000代左右的模型在200步内就能达到460左右的高分，如文章开篇展示的动图所示。而理论上的最高分为500：

$$500分 = 10行 * \quad * 50\% \quad$$