# 人工智能中的常用搜索策略

  人工智能中的搜索策略大体分为两种：无信息搜索和有信息搜索。无信息搜索是指我们不知道接下来要搜索的状态哪一个更加接近目标的搜索策略，因此也常被成为盲目搜索；而有信息搜索则是用启发函数f（n）来衡量哪一个状态更加接近目标状态，并优先对该状态进行搜索，因此与无信息搜索相比往往能够更加高效得解决问题。  
  要衡量一个搜索策略的好坏，我们需要从四个方面对其进行判断：完备性、时间复杂度、空间复杂度和最优性。因此以下通过这四个方面来比较常见搜索策略之间的优劣。

# 无信息搜索策略

### 宽度优先搜索（BFS）

  首先扩展根节点，然后扩展根节点的所有后继，接着再扩展它们的后继，从而一层一层的对节点进行扩展。BFS是一个简单的搜索策略，在搜索过程中会对所有状态进行遍历，因此它是完备的；假设搜索树每个节点有b个后继，深度为d，则时间复杂度和空间复杂度均为O(bd)；最后考虑最优性，因为我们总会在最浅那一层找到目标状态，因此当且仅当每一步的代价都一致的时候，BFS可以得到最优解。

### 一致代价搜索

  在BFS的基础上，一致代价搜索不在扩展深度最浅的节点，而是通过比较路径消耗g（n），并选择当前代价最小的节点进行扩展，因此可以保证无论每一步代价是否一致，都能够找到最优解。

### 深度优先搜索（DFS）

  DFS扩展根节点的一个后继，然后扩展它的一个后继，直到到达搜索树的最深层，那里的节点没有后继，于是DFS回溯到上一层，扩展另外一个未被扩展的节点。在有限状态空间中，DFS是完备的，因为它可以把所有空间遍历一遍；而在无限空间中，DFS则有可能会进入深度无限的分支，因此是不完备的。DFS的时间复杂度为为O(bd)，而空间复杂度仅为O（d），因为我们只需要保存当前分支的状态，因此空间复杂度远远好于BFS。然而DFS并不能保证找到最优解。

### 深度受限搜索

  深度受限搜索设定一个最大深度dmax，当搜索深度大于dmax的时候立即回溯，从而避免了在无穷状态空间中陷入深度无限的分支。

### 迭代加深的深度有限搜索

  迭代加深的深度有限搜索也设定一个最大深度dmax，开始我们把dmax设为1，然后进行深度受限搜索，如果么有找到答案，则让dmax加一，并再次进行深度有限搜索，以此类推直到找到目标。这样既可以避免陷入深度无限的分支，同时还可以找到深度最浅的目标解，从而在每一步代价一致的时候找到最优解，再加上其优越的空间复杂度，因此常常作为首选的无信息搜索策略。

# 有信息搜索

### 贪婪最佳优先搜索

贪婪最佳优先搜索总是扩展距离目标最近的节点，其启发函数f（n）=h（n）其中:

f（n）=节点n到目标节点的最小代价路径的估计值

贪婪最佳优先搜索的最大问题是它往往不能找到最优解。

### A\*

为了找到最优解，A\*算法对一个节点的评估结合了h（n）和g（n）从开始节点到节点n的路径代价，即f（n）=g（n）+h（n）

f（n）=经过节点n的最小代价解的估计代价

因为A\*搜索总是搜索f（n）最小的点，因此它总能找到最优解。

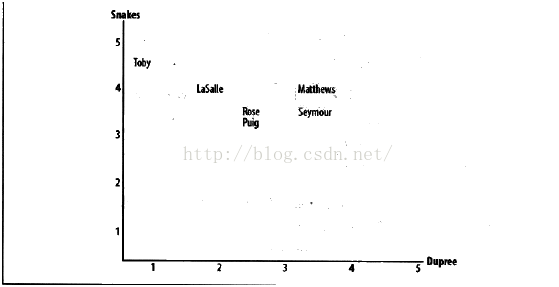
# 相似度算法之欧几里得距离

2016年07月13日 23:27:38

阅读数：18343

**在计算用户相似度的过程中，欧几里得距离是比较直观，常见的一种相似度算法。**

**根据两用户之间共同评价的Item为维度，建立一个多维的空间，那么通过用户对单一维度上的评价Score组成的坐标系X（s1,s2,s3……，si）即可定位该用户在这个多维度空间中的位置，那么任意两个位置之间的距离Distance(X,Y)（即：欧式距离）就能在一定程度上反应了两用户兴趣的相似程度。**



**上图即二维空间中6位用户对Snakes 和 Dupree 这两Item评价的直观体现**

**就其意义而言，欧氏距离越小，两个用户相似度就越大，欧氏距离越大，两个用户相似度就越小。**

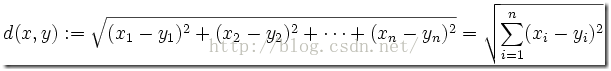
**在日常使用中，一般习惯于将相似度与1类比，相似度在数值上反映为0<=Similarity(X,y)<=1，越接近1，相似度越高；**

**那么我们在使用欧几里得距离时，可以通过 1/（1+Distance(X,Y)）来贯彻上一理念。**

**1.定义**

    欧几里得度量（euclidean metric）（也称欧氏距离）是一个通常采用的距离定义，指在m维空间中两个点之间的真实距离，或者向量的自然长度（即该点到原点的距离）。在二维和三维空间中的欧氏距离就是两点之间的实际距离。

**2.公式**

****

**3.注意事项**

a.因为计算是基于各维度特征的绝对数值，所以欧氏度量需要保证各维度指标在相同的刻度级别，比如对身高（cm）和体重（kg）两个单位不同的指标使用欧式距离可能使结果失效。

b.欧几里得距离是数据上的直观体现，看似简单，但在处理一些受主观影响很大的评分数据时，效果则不太明显；比如，U1对Item1,Item2 分别给出了2分，4分的评价;U2 则给出了4分，8分的评分。通过分数可以大概看出，两位用户褒Item2 ,贬Item1，也许是性格问题，U1 打分更保守点，评分偏低，U2则更粗放一点，分值略高。在逻辑上，是可以给出两用户兴趣相似度很高的结论。如果此时用欧式距离来处理，得到的结果却不尽如人意。即评价者的评价相对于平均水平偏离很大的时候欧几里德距离不能很好的揭示出真实的相似度。

**4.代码实现**

#得到两者共同评分项

defGetSameItem(UL,p1,p2):

        si = {}

        for item in UL[p1]:

               if item in UL[p2]:

                       si[item] = 1

        return si

#欧几里得相似度算法

defEuclidSimilarity(UL,p1,p2):

        si = GetSameItem(UL,p1,p2)

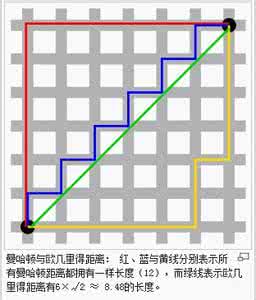
        if len(si) == 0:

               return 0

        sum\_of\_squares = sum([pow(UL[p1][item] -UL[p2][item] , 2) for item in si])

        return 1/(1+math.sqrt(sum\_of\_squares))

### 概念

曼哈顿距离——两点在南北方向上的距离加上在东西方向上的距离，即d（i，j）=|xi-xj|+|yi-yj|。对于一个具有正南正北、正东正西方向规则布局的城镇街道，从一点到达另一点的距离正是在南北方向上旅行的距离加上在东西方向上旅行的距离，因此，曼哈顿距离又称为出租车距离。   
——引用自百度   


### 简析

* 就曼哈顿距离的概念来说，只能上、下、左、右四个方向进行移动，而且两点之间的曼哈顿距离是两点之间的最短距离（在只能向上、下、左、右四个方向进行移动的前提下）。为什么呢？假设从一点到达另一点（只能向上、下、左、右四个方向进行移动，下同），要使路程最短，就只能每一步都有用（使之与另一点的南北距离或东西距离缩短），所以我们最先想到的是图中的红线，它的长度就是两点之间的曼哈顿距离。而红线可以通过平移转化为蓝线、黄线等线，它们的长度都与红线相等。（再解释不下去了，体会一下）
* 那么我们可以利用曼哈顿距离解决什么问题呢？曼哈顿距离可以代替一个广搜，不过这个广搜是有条件限制的：   
  ①只能上、下、左、右四个方向进行移动（很多迷宫问题其实都有这个特性）   
  ②只求两点之间最短路径的长度，不求路径过程   
  为了让大家更清晰地体会，举一例题：[最少联通代价](http://blog.csdn.net/cqbzlytina/article/details/75162183)   
  此题本是要用深搜+广搜解决，但巧法可只用一个小深搜+曼哈顿距离思想