分组准确度分析文档

# 聚类算法得到的结果文本说明:

**adjacency.txt:**

其实就是一个矩阵A：行数=列数=本次模拟数据的总人数

第i行第j列的元素表示：在所有聚类（总的聚类次数在该数据文件中不会显示）中，将第i个人和第j个人分在同一组的聚类次数。该值越大，说明两人越有可能在同一组。

第i行第i列的元素全部人为设置成了0（实际应该等于总的聚类次数），因为讨论同一个人的所属组没意义。

可以看到adjacency.txt文件中的每个位置上的数据即a\_ij并不是整数，这是因为我的程序运行了20次，每次选取文件中200条坐标记录做聚类，一条记录做一次聚类，即程序运行一次得到的矩阵A是这200次聚类的结果，程序重复运行了20次，我把这20个矩阵A求了平均，最后得到的结果就是这个adjacency.txt文件。

为什么程序要重复运行20次？因为每次运行的200条记录都是从所有文件中随机选取的，为了避免随机性，重复运行20次求平均，应该是比较合理的方法。

**groupid.txt:**

这个文档显示的是本次模拟实验的数据，每个人的真实所属组号，这是模拟前就设定好的，并不是聚类的结果。

第一列表示人的ID号，即1,2,3.....,第二列是对应的人的所属组号。

# 分组准确度分析类:

**GroupIDReader**类(关键性的成员方法和成员变量):

public void read(String fileName): 用来从groupid.txt文件中读取原始的分组信息(即模拟器里给定的分组情况).

private void addIntoGroupList(String lineString): 处理一行的信息,将其添加到原始分组数组里

private ArrayList<ArrayList<String>> groupList: 一个groupList是一个群组列表, 每个列表项是一个ArrayList, 里面存放着一个组里的所有人的独立的ID, 用String的形式存放. ID字面上是连续的正整数, 从1开始. 而群组的组号则是从0开始的连续非负整数.

**ClassifyGroup**类(关键性的成员方法和成员变量):

private ArrayList<ArrayList<String>> groupList: 和上面的groupList一样, 只不过这里存放的是经过分析adjacency.txt后的得到的分组信息.

private void analyzeSingleLine(String lineString): 分析adjacency.txt中单行数据, 对groupList进行数据的添加.

public static double analyzeAccuracy (ArrayList<ArrayList<String>> origin, ArrayList<ArrayList<String>> result): 对原始分组和结果分组信息进行对比, 得出分组准确率等信息.

# ClassifyGroup类分析adjacency.txt和计算分组准确度算法的详细说明:

**analyzeSingleLine()分析adjacency.txt:**

因为adjacency.txt里的矩阵第i行第j列的元素代表人i和人j在同一组的可能性相对大小, 因此比较该元素值与周围元素的相差, 来决定是否将i和j归入同一组.

显然adjacency.txt里的矩阵是对称矩阵, 因而只需分析右上半部分即可.

分析单行时, 首先检查行号i是否已在groupList中, 如果在, 就把这一行上和i同组的列号加入到i所在的组. 如果i不在groupList中, 则分析和i在同组的列号中, 有没有在groupList中的列号, 如果有, 则把i和其他列号都加入到该组, 如果没有, 则在groupList中新建一组, 将i和与其相关的列号放进去.

分析完矩阵后, 就得到了结果组的矩阵.

**analyzeAccuracy()计算分组准确度:**

首先确定一个可容忍的误差率errorRate(双精度类型), 以及一个表示成功分组数量的整型变量hits=0.

将原始组表中每组的每个人, 在结果组表中所在组的组号, 进行记录. 原始组表每组的这种信息放在相应的一个数组里, 并排序. 然后分析原始组表, 计算每组人员中, 被归到同一组的最大人数. 用这个数除以该原始组的总人数, 就得到一个比率, 如果这个比率大于(1-errorRate), 说明这个原始组在容忍范围内被很好地聚类出来, hits自加1, 否则说明该组的聚类失败, hits不变.

分析完原始组表中所有的组后, 用hits除以原始组表的组数就得到了一个准确率a.

这个准确率a在大部分情况下能比较准确地反映分组结果的正确率, 但是, 还要注意:

1. a需要和errorRate同时考虑, errorRate越小, 可容忍的误差越小, a的值可能越小, 反之同理.
2. a的值还需要和原始组以及结果组的组的数量进行比较. 原始组和结果组的组数如果相差很大, 即使a再高也没用, 分类结果还是失败的. 只有在原始组和结果组的组数接近的情况下(组数偏差不超过20%), a才能在可容忍误差范围内准确地反映分组准确率.

于是, 在a的基础上我们又得出一个新的准确率ag(考虑分组结果中组数偏差的准确率). 如果组数偏差在errorRate范围内, a=ag, 如果超出errorRate, 则:

**ag = a \* ( | 结果组数 - 原始组数 | - ( errorRate \* 原始组数 ) ) / 结果组数**

(ag存在问题，目前不做考虑)

这样, ag就能放映出特殊情况下的分组正确率, 能保证极端情况下(如所有人全被分到一组中)我们能及时发现实际的分组情况, 不至于被a所骗. ag虽然能稳定地反应准确率, 但是其值大部分情况下偏低, 无法准确表示实际的分组正确率, 所以, 我们应该同时考虑a和ag两个准确率, 以a为主, 以ag为辅(用作参考, 排除极端情况).