

数学与计算机学院

人工智能课程设计

——基于Kmeans算法实现亚洲足球成绩评估

|  |  |
| --- | --- |
| 专 业 班 级 | 信科11602 |
| 学生学号 | 1604270309 |
| 学生姓名 | 罗 勇 |
| 任 课 教 师 | 叶 威 |
| 实验指导教师 | 叶 威 |
| 实验地点 | A3015 |
| 填写时间 | 2018.11.13 |

目录

[**1.课程设计介绍** 3](#_Toc529871360)

[**1.1 K均值聚类实现对亚洲足球水平的分类** 3](#_Toc529871361)

[**2.问题分析** 4](#_Toc529871362)

[**2.1 相异度计算** 4](#_Toc529871363)

[**2.2 标量** 4](#_Toc529871364)

[**2.2.1 欧氏距离的定义与计算** 4](#_Toc529871365)

[**2.2.2 标量格式化** 5](#_Toc529871366)

[**2.3聚类问题** 6](#_Toc529871367)

[**3.Kmeans聚类实现对亚洲国家足球成绩聚类** 6](#_Toc529871368)

[**3.1初始化聚类中心** 6](#_Toc529871369)

[**3.2 数据集聚类** 7](#_Toc529871370)

[**3.2.1 相异度计算** 7](#_Toc529871371)

[**3.2.2 划分数据集** 8](#_Toc529871372)

[**3.3 中心簇更新** 9](#_Toc529871373)

[**3.4 kmeans迭代** 10](#_Toc529871374)

[**4 结果分析及展示** 10](#_Toc529871375)

[**5 算法分析** 12](#_Toc529871376)

[**5.1 复杂度分析** 12](#_Toc529871377)

[**5.2 算法收敛** 12](#_Toc529871378)

[**6 误差分析** 13](#_Toc529871379)

[**7.代码功能块介绍** 16](#_Toc529871380)

[**7.1 对数据0-1初始化处理** 16](#_Toc529871381)

[**7.2 kmeans 算法运行实体类** 17](#_Toc529871382)

**1.课程设计介绍**

**1.1 K均值聚类实现对亚洲足球水平的分类**

   K-means算法采用距离作为相似性的评价指标，即认为两个对象的距离越近，其相似度就越大。  
   该算法认为类簇是由距离靠近的对象组成的，因此把得到紧凑且独立的簇作为最终目标。它是数据点到原型的某种距离作为优化的目标函数，利用函数求极值的方法得到迭代运算的调整规则。  
   K-means算法以欧式距离作为相似度测度，它是求对应某一初始聚类中心向量V最优分类，使得评价指标最小。算法采用误差平方和准则函数作为聚类准则函数。以下表中给出的数据为亚洲国家在2015-2018中足球世界杯中的分数,现要求运用Kmeans算法思想,实现对亚洲国家足球水平的分类

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 2015 | 2017 | 2016 | 2015 |
| 伊朗 | 405.04 | 468.14 | 324.59 | 304.37 |
| 澳大利亚 | 426.07 | 360.57 | 431.26 | 136.76 |
| 日本 | 268.99 | 358.14 | 179.07 | 259.66 |
| 韩国 | 213.57 | 401.08 | 332.3 | 276.55 |
| 沙特阿拉 | 219.26 | 324.72 | 384.56 | 122.58 |
| 中国 | 300.71 | 213.32 | 145.74 | 262.48 |
| 巴勒斯坦 | 309.28 | 91.46 | 193.06 | 150.68 |
| 叙利亚 | 259.07 | 220.19 | 208.79 | 51 |
| 乌兹别克 | 164.9 | 331.29 | 223.89 | 215.14 |
| 阿拉伯联 | 183.25 | 256.84 | 269.52 | 219.18 |
| 伊拉克 | 253.64 | 91.69 | 249.9 | 179.75 |
| 黎巴嫩 | 310.25 | 66.56 | 135.7 | 114.23 |
| 阿曼 | 205.63 | 101.58 | 160. | 152.36 |
| 印度 | 215.81 | 171.36 | 99.71 | 8.6 |
| 卡塔尔 | 116.99 | 198.84 | 269.91 | 150.47 |
| 越南 | 180.89 | 112.48 | 121.13 | 122.63 |
| 约旦 | 148.26 | 109.37 | 208.54 | 140.28 |
| 土库曼斯 | 184.52 | 63.75 | 207.98 | 64.5 |
| 吉尔吉斯 | 148.75 | 92.41 | 246.8 | 66.96 |
| 朝鲜 | 114.02 | 124.31 | 283.58 | 37.91 |

**2.问题分析**

**2.1 相异度计算**

现对x ,y作如下定义

https://upload-images.jianshu.io/upload_images/6768913-29af88f9eb4b4a60.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240

  其中X，Y是两个元素项，各自具有n个可度量特征属性，那么X和Y的相异度定义为

https://upload-images.jianshu.io/upload_images/6768913-e3438f3bea01a25d.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240

  其中R为实数域。也就是说相异度是两个元素对实数域的一个映射，所映射的实数定量表示两个元素的相异度。下面介绍不同类型变量相异度计算方法。

**2.2 标量**

**2.2.1 欧氏距离的定义与计算**

  标量也就是无方向意义的数字，也叫标度变量。现在先考虑元素的所有特征属性都是标量的情况。例如，计算X={2,1,102}和Y={1,3,2}的相异度。一种很自然的想法是用两者的欧几里得距离来作为相异度，欧几里得距离的定义如下

https://upload-images.jianshu.io/upload_images/6768913-37555b5c7a0396f0.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240

  其意义就是两个元素在欧氏空间中的集合距离，因为其直观易懂且可解释性强，被广泛用于标识两个标量元素的相异度。

**2.2.2 标量格式化**

  在计算相异度的方式有一个问题，就是取值范围大的属性对距离的影响高于取值范围小的属性。而这样不利于真实反映真实的相异度，为了解决这个问题,一般要对属性值进行规格化。  
  所谓规格化就是将各个属性值按比例映射到相同的取值区间，这样是为了平衡各个属性对距离的影响。通常将各个属性均映射到[0,1]区间，映射公式为：

https://upload-images.jianshu.io/upload_images/6768913-bd4f9ad4341247a8.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240

其中max(ai)和min(ai)表示所有元素项中第i个属性的最大值和最小值。

综上，对以上表格中个国家成绩进行[0,1]格式化后结果如下(格式化过程见)：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 2018 | 2017 | 2016 | 2015 |
| 伊朗 | 0.9326069540137799 | 1.0 | 0.6782687377469461 | 1.0 |
| 澳大利亚 | 1.0 | 0.7339944113355918 | 1.0 | 0.4333096662947561 |
| 日本 | 0.49661913154943127 | 0.7279853606666832 | 0.23936057909817524 | 0.8488352436014472 |
| 韩国 | 0.3190193879186028 | 0.8341699844209798 | 0.7015231488463279 | 0.9059404266828955 |
| 沙特阿拉 | 0.33725364524915874 | 0.6453423675165064 | 0.8591464334187906 | 0.38536700814822333 |
| 中国 | 0.5982695080916519 | 0.36986572368258364 | 0.13883275524053693 | 0.8583696791425771 |
| 巴勒斯坦 | 0.6257330556000641 | 0.06852296050842008 | 0.28155632634595085 | 0.48037326300841876 |
| 叙利亚 | 0.46482935427014904 | 0.38685427433912806 | 0.32900015080681644 | 0.14335463366805287 |
| 乌兹别克 | 0.16305079314212467 | 0.6615890600657781 | 0.37454380938018395 | 0.6983128782499916 |
| 阿拉伯联 | 0.2218554718795065 | 0.47748460644427404 | 0.5121701100889761 | 0.7119721405145891 |
| 伊拉克 | 0.44742829674731616 | 0.06909171839066247 | 0.45299351530689186 | 0.578659093214322 |
| 黎巴嫩 | 0.6288415318058004 | 0.006948737604787463 | 0.1085507464937415 | 0.35713561213104783 |
| 阿曼 | 0.2935747476365967 | 0.09354830732708524 | 0.18184285929724026 | 0.4860533522669643 |
| 印度 | 0.326197724723602 | 0.2661045030787112 | 0.0 | 0.0 |
| 卡塔尔 | 0.009517705495914113 | 0.3340587057048889 | 0.5133464032574273 | 0.47966325185110054 |
| 越南 | 0.21429258131709658 | 0.12050248522465938 | 0.0646056401749359 | 0.38553605842377525 |
| 约旦 | 0.10972600544784487 | 0.11281188951259924 | 0.32824611672447596 | 0.4452108056936133 |
| 土库曼斯 | 0.22592533247876947 | 0.0 | 0.32655708038003317 | 0.18899820806707915 |
| 吉尔吉斯 | 0.11129626662393848 | 0.0708721778481169 | 0.4436434926858695 | 0.19731548162423504 |
| 朝鲜 | 0.0 | 0.1497564232547788 | 0.5545769868798069 | 0.09909727152855258 |

**2.3聚类问题**

  所谓聚类问题，就是给定一个元素集合D，其中每个元素具有n个可观察属性，使用某种算法将D划分成k个子集，要求每个子集内部的元素之间相异度尽可能低，而不同子集的元素相异度尽可能高。其中每个子集叫做一个簇。

**3.Kmeans聚类实现对亚洲国家足球成绩聚类**

**3.1初始化聚类中心**

  本次试验中，我们预计将21个国家分成三个水平段，即初始化聚类中心为3个，k=3  
  聚类中心的选择可以随机，但是该方法在有些情况下的效果较差，为此可尽量遵循以下两种方法  
  1).选择彼此距离尽可能远的K个点  
  2)先对数据用层次聚类算法或者Canopy算法进行聚类，得到K个簇之后，从每个类簇中选择一个点，该点可以是该类簇的中心点，或者是距离类簇中心点最近的那个点为保证试验的准确性，选取3个  
  综上所述，遵循彼此尽可能远的原则，选取的三个初始点  
澳大利亚(1.0 0.73 1.0 0.43)，朝鲜(0.0 0.15 0.55 0.1)，黎巴嫩(0.45 0.07 0.45 0.58)，这里值为保留2为小数的结果，实际计算以以上数据为准。

#### 3.2 数据集聚类

##### 3.2.1 相异度计算

  中心簇确定后，分别计算剩下的元素到k个簇中心的相异度,将这些元素分别划归到相异度最低的簇，这里就以第一次计算的结果为例进行说明

|  |  |  |  |
| --- | --- | --- | --- |
|  | Cluster1 | Cluster2 | Cluster3 |
| 伊朗 | 1.3476933868255538 | 1.555504814938461 | 0.7070712314829007 |
| 澳大利亚 | 1.2111310337452303 | 1.285081168918857 | 0.0 |
| 日本 | 0.8923319785790955 | 1.1146513862920968 | 1.0023284203065108 |
| 韩国 | 1.197116421098692 | 1.1148023711655355 | 0.8867006325512223 |
| 沙特阿拉 | 1.0279887251446345 | 0.7307929408111575 | 0.6850037178877967 |
| 中国 | 0.6203192367698562 | 1.0750406804405832 | 1.1028427638435183 |
| 巴勒斯坦 | 0.2211775167456911 | 0.7861630944074157 | 1.0494304138333406 |
| 叙利亚 | 0.515294515174758 | 0.5701976378443554 | 0.9701694193729713 |
| 乌兹别克 | 0.9125071193616101 | 0.8246390373512267 | 1.0803465736615188 |
| 阿拉伯联 | 0.8221065627434258 | 0.7307801144235243 | 0.9934469986301134 |
| 伊拉克 | 0.4522012608065473 | 0.6685788083416322 | 1.0333313314810477 |
| 黎巴嫩 | 0.0 | 0.825444654817664 | 1.2111310337452303 |
| 阿曼 | 0.37668926806365177 | 0.6148261657085301 | 1.257525693225462 |
| 印度 | 0.5459707822128316 | 0.6613000327732459 | 1.364070286258954 |
| 卡塔尔 | 0.8181903177041594 | 0.424956870055824 | 1.1747272519735894 |
| 越南 | 0.43299310454257817 | 0.6073682437550734 | 1.3678279935236912 |
| 约旦 | 0.5802682205652083 | 0.4294472458841903 | 1.2766559697471915 |
| 土库曼斯 | 0.48804373443626964 | 0.3654374098048476 | 1.2849719562146842 |
| 吉尔吉斯 | 0.6401319196878827 | 0.20140169471671437 | 1.2628346058552768 |
| 朝鲜 | 0.825444654817664 | 0.0 | 1.285081168918857 |

##### 3.2.2 划分数据集

   由以上数据可以看出，表中三个为0的点恰为所选取的初始中心点，其余数剧皆表示当前点到中心簇的欧式距离。

  根据当前点到k个中心簇的距离，将其划分到离其最近的中心簇中，即minLength{Cluster0，Cluster1......Clustern},以下为第一次进行聚类时，Cluster1簇中数据成员

Cluster

Cluster\_id=1, center:{Point\_id=11,name=黎巴嫩 clusterId=1,dist=0.0

[0.6288415, 0.0069487374, 0.10855075, 0.35713562]}

Point\_id=2,name=日本,clusterId=1,dist=0.89233196 [0.49661914, 0.7279854, 0.23936059, 0.84883523]

Point\_id=5,name=中国,clusterId=1,dist=0.62031925 [0.5982695, 0.36986572, 0.13883275, 0.8583697]

Point\_id=6,name=巴勒斯坦,clusterId=1,dist=0.22117752 [0.6257331, 0.06852296, 0.28155634, 0.48037326]

Point\_id=7,name=叙利亚,clusterId=1,dist=0.5152945 [0.46482936, 0.38685426, 0.32900015, 0.14335464]

Point\_id=10,name=伊拉克,clusterId=1,dist=0.45220125 [0.4474283, 0.069091715, 0.4529935, 0.5786591]

Point\_id=11,name=黎巴嫩,clusterId=1,dist=0.0 [0.6288415, 0.0069487374, 0.10855075, 0.35713562]

Point\_id=12,name=阿曼,clusterId=1,dist=0.37668926 [0.29357475, 0.093548305, 0.18184286, 0.48605335]

Point\_id=13,name=印度,clusterId=1,dist=0.5459708 [0.3261977, 0.2661045, 0.0, 0.0]

Point\_id=15,name=越南,clusterId=1,dist=0.4329931 [0.21429259, 0.12050249, 0.06460564, 0.38553604]

其中  
"Cluster\_id"为中心簇编号  
"Point\_id"为数据编号,这里便指国家编号  
“dist”表示数据点到当前中心簇欧氏距离，并携带了数据信息

#### 3.3 中心簇更新

  在3.2中，我们通过计算数据集元素到中心簇的欧式距离，实现了对数据里的第一次聚类，已经将所有的数据集划分为三个簇。接下来我们需要重新确定新的中心簇，用以进行下一次聚类，直到达到所规定的最大迭代次数或者当中心簇不在变化时，我们即可认为当前的聚类是稳定可靠的。  
  此处我们就第一次聚类为例，计算新的中心簇，新的中心簇的计算遵循以下步骤：  
1）对同一簇内的所有点进行对应的各个维度求和  
2）对同一簇各个维度和计算平均值  
3）计算两个新旧中心的距离，如果任意一个簇中心的移动个距离大于某一规定值，则继续进行迭代。

Point\_id=-1,name=null,clusterId=0,dist=0.0 [0.45508733, 0.23438042, 0.19963807, 0.459813]

 以上为Cluster1重新确定中心簇后的数据，可以看出，新的中心簇中心不属于已知的元素中任意一个。其他中心簇的确定类似。

#### 3.4 kmeans迭代

  由于中心点的选取以及其他数据产生的影响，会导致聚类的结果会产生误差，因此要尽可能使迭代趋于收敛，反应在本设计中的表现就是中心簇中点趋于稳定。

### 4 结果分析及展示

  本次结果是在人为规定初始中心簇中心的情况下进行的，故存在误差或结果差异是必然的，此后会在问题分析中对结果进行分析

单次迭代运行次数：3

Cluster

Cluster\_id=2, center:{Point\_id=-1,name=null,clusterId=0,dist=0.0 [0.12019593, 0.25808185, 0.4361549, 0.40293857]}

Point\_id=8,name=乌兹别克,clusterId=2,dist=0.50566447 [0.1630508, 0.6615891, 0.37454382, 0.6983129]

Point\_id=9,name=阿拉伯联,clusterId=2,dist=0.39969024 [0.22185548, 0.4774846, 0.51217014, 0.7119721]

Point\_id=14,name=卡塔尔,clusterId=2,dist=0.17282173 [0.009517705, 0.3340587, 0.5133464, 0.47966325]

Point\_id=16,name=约旦,clusterId=2,dist=0.1861296 [0.109726004, 0.11281189, 0.32824612, 0.4452108]

Point\_id=17,name=土库曼斯,clusterId=2,dist=0.3681944 [0.22592533, 0.0, 0.32655707, 0.1889982]

Point\_id=18,name=吉尔吉斯,clusterId=2,dist=0.27832282 [0.11129627, 0.07087218, 0.44364348, 0.19731548]

Point\_id=19,name=朝鲜,clusterId=2,dist=0.36403954 [0.0, 0.14975642, 0.554577, 0.099097274]

Cluster

Cluster\_id=0, center:{Point\_id=-1,name=null,clusterId=0,dist=0.0 [0.6170998, 0.7882984, 0.69565976, 0.7146905]}

Point\_id=0,name=伊朗,clusterId=0,dist=0.47546425 [0.93260694, 1.0, 0.67826873, 1.0]

Point\_id=1,name=澳大利亚,clusterId=0,dist=0.5668859 [1.0, 0.7339944, 1.0, 0.43330967]

Point\_id=2,name=日本,clusterId=0,dist=0.4943248 [0.49661914, 0.7279854, 0.23936059, 0.84883523]

Point\_id=3,name=韩国,clusterId=0,dist=0.3571653 [0.31901938, 0.83417, 0.7015231, 0.9059404]

Point\_id=4,name=沙特阿拉1,clusterId=0,dist=0.48366535 [0.33725366, 0.64534235, 0.8591464, 0.385367]

Cluster

Cluster\_id=1, center:{Point\_id=-1,name=null,clusterId=0,dist=0.0 [0.44989586, 0.17267983, 0.19467275, 0.41118518]}

Point\_id=5,name=中国,clusterId=1,dist=0.5137987 [0.5982695, 0.36986572, 0.13883275, 0.8583697]

Point\_id=6,name=巴勒斯坦,clusterId=1,dist=0.2326008 [0.6257331, 0.06852296, 0.28155634, 0.48037326]

Point\_id=7,name=叙利亚,clusterId=1,dist=0.3686065 [0.46482936, 0.38685426, 0.32900015, 0.14335464]

Point\_id=10,name=伊拉克,clusterId=1,dist=0.32482874 [0.4474283, 0.069091715, 0.4529935, 0.5786591]

Point\_id=11,name=黎巴嫩,clusterId=1,dist=0.26424742 [0.6288415, 0.0069487374, 0.10855075, 0.35713562]

Point\_id=12,name=阿曼,clusterId=1,dist=0.1909658 [0.29357475, 0.093548305, 0.18184286, 0.48605335]

Point\_id=13,name=印度,clusterId=1,dist=0.48062474 [0.3261977, 0.2661045, 0.0, 0.0]

Point\_id=15,name=越南,clusterId=1,dist=0.2753302 [0.21429259, 0.12050249, 0.06460564, 0.38553604]

以上结果显示,这里还尚未展示图形化功能，根据数据信息可以看出

亚洲一流:伊朗 澳大利亚 日本 韩国 沙特阿拉

亚洲二流:中国 巴勒斯坦 叙利亚 伊拉克 黎巴嫩 阿曼 印度 越南

亚洲三流:乌兹别克 阿拉伯 卡塔尔 约旦 土库曼斯 吉尔吉斯 朝鲜

### 5 算法分析

#### 5.1 复杂度分析

本实验重复3.2，直到这k个中线点不再变化（收敛了），或执行了足够多的迭代  
时间复杂度：O(I\*n\*k\*m)空间复杂度：O(n\*m)  
其中m为每个元素字段个数，n为数据量，I为跌打个数。一般I,k,m均可认为是常量，所以时间和空间复杂度可以简化为O(n)，即线性的。

#### 5.2 算法收敛

从kmeans的算法可以发现，SSE其实是一个严格的坐标下降（Coordinate Decendet）过程。设目标函数SSE如下：

https://upload-images.jianshu.io/upload_images/6768913-ced05f8c46d81c57.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240

采用欧式距离作为变量之间的聚类函数。每次朝一个变量Ci 其中m是c\_i所在的簇的元素的个数也就是当前聚类的均值就是当前方向的最优解（最小值），这与kmeans的每一次迭代过程一样。所以，这样保证SSE每一次迭代时，都会减小，最终使SSE收敛。

由于SSE是一个非凸函数（non-convex function），所以SSE不能保证找到全局最优解，只能确保局部最优解。但是可以重复执行几次kmeans，选取SSE最小的一次作为最终的聚类结果

### 6 误差分析

  固定初始中心点中心簇，所产生误差是必然的，本部分，我们随机选取中心簇中心点对结果加以校正。随机选取过程中，如若中心簇中心点太近，实验结果也是会产生误差。

随机测试1：

单次迭代运行次数：2

Cluster

Cluster\_id=1, center:{Point\_id=-1,name=null,clusterId=0,dist=0.0 [0.42557815, 0.15708022, 0.16092595, 0.30874217]}

Point\_id=6,name=巴勒斯坦,clusterId=1,dist=0.30317208 [0.6257331, 0.06852296, 0.28155634, 0.48037326]

Point\_id=7,name=叙利亚,clusterId=1,dist=0.33157012 [0.46482936, 0.38685426, 0.32900015, 0.14335464]

Point\_id=11,name=黎巴嫩,clusterId=1,dist=0.2625653 [0.6288415, 0.0069487374, 0.10855075, 0.35713562]

Point\_id=12,name=阿曼,clusterId=1,dist=0.23095015 [0.29357475, 0.093548305, 0.18184286, 0.48605335]

Point\_id=13,name=印度,clusterId=1,dist=0.37812915 [0.3261977, 0.2661045, 0.0, 0.0]

Point\_id=15,name=越南,clusterId=1,dist=0.2472942 [0.21429259, 0.12050249, 0.06460564, 0.38553604]

Cluster

Cluster\_id=0, center:{Point\_id=-1,name=null,clusterId=0,dist=0.0 [0.15064894, 0.12276515, 0.43656054, 0.3314907]}

Point\_id=10,name=伊拉克,clusterId=0,dist=0.39028335 [0.4474283, 0.069091715, 0.4529935, 0.5786591]

Point\_id=14,name=卡塔尔,clusterId=0,dist=0.30399698 [0.009517705, 0.3340587, 0.5133464, 0.47966325]

Point\_id=16,name=约旦,clusterId=0,dist=0.16259776 [0.109726004, 0.11281189, 0.32824612, 0.4452108]

Point\_id=17,name=土库曼斯,clusterId=0,dist=0.23052698 [0.22592533, 0.0, 0.32655707, 0.1889982]

Point\_id=18,name=吉尔吉斯,clusterId=0,dist=0.14931399 [0.11129627, 0.07087218, 0.44364348, 0.19731548]

Point\_id=19,name=朝鲜,clusterId=0,dist=0.3022552 [0.0, 0.14975642, 0.554577, 0.099097274]

Cluster

Cluster\_id=2, center:{Point\_id=-1,name=null,clusterId=0,dist=0.0 [0.5085844, 0.681304, 0.5629807, 0.7302634]}

Point\_id=0,name=伊朗,clusterId=2,dist=0.6061447 [0.93260694, 1.0, 0.67826873, 1.0]

Point\_id=1,name=澳大利亚,clusterId=2,dist=0.72348666 [1.0, 0.7339944, 1.0, 0.43330967]

Point\_id=2,name=日本,clusterId=2,dist=0.3480109 [0.49661914, 0.7279854, 0.23936059, 0.84883523]

Point\_id=3,name=韩国,clusterId=2,dist=0.3306952 [0.31901938, 0.83417, 0.7015231, 0.9059404]

Point\_id=4,name=沙特阿拉1,clusterId=2,dist=0.48714998 [0.33725366, 0.64534235, 0.8591464, 0.385367]

Point\_id=5,name=中国,clusterId=2,dist=0.5489535 [0.5982695, 0.36986572, 0.13883275, 0.8583697]

Point\_id=8,name=乌兹别克,clusterId=2,dist=0.3953624 [0.1630508, 0.6615891, 0.37454382, 0.6983129]

Point\_id=9,name=阿拉伯联,clusterId=2,dist=0.35591024 [0.22185548, 0.4774846, 0.51217014, 0.7119721]

随机测试2：

Cluster

Cluster\_id=1, center:{Point\_id=-1,name=null,clusterId=0,dist=0.0 [0.18291941, 0.5296187, 0.5648017, 0.5688288]}

Point\_id=4,name=沙特阿拉1,clusterId=1,dist=0.39687288 [0.33725366, 0.64534235, 0.8591464, 0.385367]

Point\_id=8,name=乌兹别克,clusterId=1,dist=0.26603594 [0.1630508, 0.6615891, 0.37454382, 0.6983129]

Point\_id=9,name=阿拉伯联,clusterId=1,dist=0.16581334 [0.22185548, 0.4774846, 0.51217014, 0.7119721]

Point\_id=14,name=卡塔尔,clusterId=1,dist=0.28090924 [0.009517705, 0.3340587, 0.5133464, 0.47966325]

Cluster

Cluster\_id=0, center:{Point\_id=-1,name=null,clusterId=0,dist=0.0 [0.669303, 0.73320305, 0.551597, 0.80929106]}

Point\_id=0,name=伊朗,clusterId=0,dist=0.4392325 [0.93260694, 1.0, 0.67826873, 1.0]

Point\_id=1,name=澳大利亚,clusterId=0,dist=0.67215204 [1.0, 0.7339944, 1.0, 0.43330967]

Point\_id=2,name=日本,clusterId=0,dist=0.35902962 [0.49661914, 0.7279854, 0.23936059, 0.84883523]

Point\_id=3,name=韩国,clusterId=0,dist=0.4058471 [0.31901938, 0.83417, 0.7015231, 0.9059404]

Point\_id=5,name=中国,clusterId=0,dist=0.55663526 [0.5982695, 0.36986572, 0.13883275, 0.8583697]

Cluster

Cluster\_id=2, center:{Point\_id=-1,name=null,clusterId=0,dist=0.0 [0.31344044, 0.12227395, 0.2792339, 0.30561215]}

Point\_id=6,name=巴勒斯坦,clusterId=2,dist=0.3618877 [0.6257331, 0.06852296, 0.28155634, 0.48037326]

Point\_id=7,name=叙利亚,clusterId=2,dist=0.34889185 [0.46482936, 0.38685426, 0.32900015, 0.14335464]

Point\_id=10,name=伊拉克,clusterId=2,dist=0.35429952 [0.4474283, 0.069091715, 0.4529935, 0.5786591]

Point\_id=11,name=黎巴嫩,clusterId=2,dist=0.38021722 [0.6288415, 0.0069487374, 0.10855075, 0.35713562]

Point\_id=12,name=阿曼,clusterId=2,dist=0.20799963 [0.29357475, 0.093548305, 0.18184286, 0.48605335]

Point\_id=13,name=印度,clusterId=2,dist=0.4384294 [0.3261977, 0.2661045, 0.0, 0.0]

Point\_id=15,name=越南,clusterId=2,dist=0.24957275 [0.21429259, 0.12050249, 0.06460564, 0.38553604]

Point\_id=16,name=约旦,clusterId=2,dist=0.25195056 [0.109726004, 0.11281189, 0.32824612, 0.4452108]

Point\_id=17,name=土库曼斯,clusterId=2,dist=0.19608189 [0.22592533, 0.0, 0.32655707, 0.1889982]

Point\_id=18,name=吉尔吉斯,clusterId=2,dist=0.28681543 [0.11129627, 0.07087218, 0.44364348, 0.19731548]

Point\_id=19,name=朝鲜,clusterId=2,dist=0.46632865 [0.0, 0.14975642, 0.554577, 0.099097274]

由此可见，初始中心点的选取以及数据处理需要尽可能的避免造成跟大的误差。

**7.代码功能块介绍**

  此处就几个重要模块加以展示，详细代码不做展示。查看源代码可到<https://github.com/luo111/kmeans>

**7.1 对数据0-1初始化处理**

*/\*\**

*\* 0均值\标准差归一化 公式：X(norm) = (X - μ) / σ*

*\* X(norm) = (X - 均值) / 标准差*

*\**

*\* @param points 原始数据*

*\* @return 归一化后的数据*

*\*/*

public static double[][] normalize4ZScore(double[][] points) {

if (points == null || points.length < 1) {

return points;

}

double[][] p = new double[points.length][points[0].length];

double[] matrixJ;

double avg;

double std;

for (int j = 0; j < points[0].length; j++) {

matrixJ = getMatrixCol(points, j);

avg = average(matrixJ);

std = standardDeviation(matrixJ);

for (int i = 0; i < points.length; i++) {

p[i][j] = std == 0 ? points[i][j] : (points[i][j] - avg) / std;

}

}

return p;

}

**7.2 kmeans 算法运行实体类**

public class KMeansRun {

private int kNum; *//簇的个数*

private int iterNum = 10; *//迭代次数*

private int iterMaxTimes = 100000; *//单次迭代最大运行次数*

private int iterRunTimes = 0; *//单次迭代实际运行次数*

private float disDiff = (float) 0.01; *//单次迭代终止条件，两次运行中类中心的距离差*

private List<float[]> original\_data =null; *//用于存放，原始数据集*

private static List<Point> pointList = null; *//用于存放，原始数据集所构建的点集*

private DistanceCompute disC = new DistanceCompute();

private int len = 0; *//用于记录每个数据点的维度*

public KMeansRun(int k, List<float[]> original\_data) throws IOException {

this.kNum = k;

this.original\_data = original\_data;

this.len = original\_data.get(0).length;

*//检查规范*

check();

*//初始化点集。*

init();

}