# 数据挖掘 聚类大作业

**5110369037**
**胡正浩**

## 一、 数据描述

这次聚类大作业我选用的是 Plants 数据集。总共 34781 条记录，每条记录分别表示一个植物及其对应的属性。每条记录可看做一条 70 维的数据，由 1 个名称属性和 69 个由 0 和 1 组成的数值属性组成，分别为植物名称和植物在某地区是否出现，1 表示出现，0 表示未出现。

## 二、 算法选择

我选择的聚类算法是 Jarvis-Patrick 算法，考虑因素如下：

（1） 数据集非常稀疏，且都是离散数据，因此不适用于传统的聚类算法，如：K-Means，DBSCAN 等。虽然在实际实现时，我对数据集做过一定的处理（详细处理过程会在稍后进行解释），但即使是处理过的数据集仍旧及其稀疏，因此，我一开始就考虑使用 KNN，SNN 等方法来实现聚类。

（2） 使用 KNN 和 SNN 时需要考虑距离矩阵（Proximity Matrix）的生成问题，即，如何衡量两个节点之间的距离。考虑到数据维数过大（70 维）以及数据组成特殊（0 和 1 组成）的问题，我认为传统的欧氏距离，SSE 都不适用。因此，在计算两个节点距离时，我选用了汉明距离作为指标。即：两个节点在相同地点出现的次数。

（3） K-Means 算法，MST 算法，Chameleon 算法若想达到理想的效果都需要预先设定分类的簇的个数，之后再对簇进行合并。我认为这样的算法效率十分低下，而且聚类的成功与否很大程度上受人为的主观因素影响，在算法效率与聚类效果之间存在 trade off。即，若想算法高效，则人为设定的簇的个数应尽量小，但这不利于聚类的结果；而若想要理想的聚类结果，则需人为设定较多的簇数，从而影响算法效率。因此，我选用 JP 算法，利用 SNN 算法生成的相似矩阵来进行聚类。不仅能一次性地得到合适的簇，而且算法效率较高，只要生成了相似矩阵，聚类的过程均可通过类似查表的过程利用相似矩阵来实现，查表的复杂度仅为 O(1)。

## 三、 算法实现

聚类算法是用 Python 来实现的。共有两个全局变量用于聚类时参数的调

整，分别为：KNN_Param（选取的最近邻的个数 k），以及 SNN_TH（判断两个节点之间是否相似的指标，为 0 到 1 之间的浮点数）。

整个算法分为 6 个阶段，预处理阶段，生成距离矩阵（Proximity Matrix）阶段，查找 K 最近邻阶段，生成相似矩阵（Similarity Matrix）阶段，聚类阶段，和评估阶段。

**预处理阶段**，将数据从文件中读入，并组织成二位列表（List）的形式，方便后续的处理，复杂度为 O(n)。

**生成距离矩阵（Proximity Matrix）阶段**，对每一对数据计算之间的距离，存储在一个二位字典（Dict）中，算法复杂度 $O(n^2)$。其中，需要特别指出的是，对于距离的计算，我原本是简单的采用汉明距离作为两点之间的距离，但后来发现这种方式聚类的结果效果并不理想。主要原因是，数据中每条数据地区个数的不同导致的不公平。经常出现的情况就是，比较两个节点 A 和 B 之间的距离时，可能 A 在 11 个地区出现，而 B 只在 2 个地区出现，导致 A 和 B 之间的距离最多只能为 2；若此时有个 C 节点和 A 的汉明距离也为 2，但却在 10 个地区出现，则在算法中 C 节点和 B 节点的地位是相同的，但这却是不合理的，实际上 B 节点的地位应该比 C 更高。因此，我对距离算法进行了修改，将距离表示为：

$$\text{距离（A，B）} = \frac{\text{汉明距离}}{\text{Min（A 地区个数， B 地区个数）}}$$

**查找 K 最近邻阶段**，遍历之前生成的距离矩阵，找出每个节点最近的 K 个节点作为邻居，存在一个二维字典（Dict）中，算法复杂度 $O(n^2)$，其中 K 通过全局变量 KNN_Param 来修改。

**生成相似矩阵（Similarity Matrix）阶段**，用 SNN 算法，计算两个节点之间的相似度，即：共同邻居的个数，若共同邻居个数小于 threshold 值，则两节点相似度记为 0，否则记为共同邻居个数，算法复杂度为 $O(n^2)$。Threshold 值由全局变量 SNN_TH 和 KNN_Param 共同计算得出，计算公式为：

$$\text{threshold} = SNN\_TH * KNN\_Param$$

**聚类阶段**，用广度优先的算法，先选定一个节点，从该节点开始遍历它的邻居节点，将所有邻居加入到列表中，之后再一个个遍历其邻居节点的邻居，直到列表不再增加为止。若一个节点遍历之后发现没有邻居节点，即列表中只有它一个节点，则将该节点标记为 outlier。算法复杂度小于 $O(n^2)$。

**评估阶段**，程序会将分类结果进行输出。输出包括：簇个数，outlier 个数，每个簇包含节点数，每个簇的地区属性值（每个节点的地区的并集），以及根据距离矩阵和相似矩阵生成的可视化矩阵图。

## 四、 Evaluation

为评价不同算法以及不同参数对聚类结果的影响，我进行了以下四组实验。

在生成距离矩阵时，使用普通汉明距离计算。最近邻 K 值取 20 个，SNN 阈值参数取 0.5，部分聚类结果截图如下：

```
Load Data Done.
Graph Generated!
Cluster Number:  85
Outlier Number:  34519
Cluster  0 :
---> Number of Nodes:  3
---> States included:  ['ca', 'az', 'nv']
Cluster  1 :
---> Number of Nodes:  2
---> States included:  ['co', 'ca', 'ga', 'ct', 'px', 'tx', 'la', 'tn', 'pa', 'pe', 'de', 'dc', 'hi', 'me', 'md', 'ma', 'mb', 'ut', 'mo', 'mn', 'mi', 'mt', 'qc',
Cluster  2 :
---> Number of Nodes:  3
---> States included:  ['ca', 'or']
Cluster  3 :
---> Number of Nodes:  2
---> States included:  ['va', 'al', 'ar', 'vt', 'il', 'ga', 'in', 'ia', 'ct', 'nh', 'nj', 'tx', 'la', 'nb', 'nc', 'ne', 'tn', 'ny', 'pa', 'ns', 'ri', 'md', 'de',
Cluster  4 :
---> Number of Nodes:  2
---> States included:  ['nm', 'tx', 'az']
Cluster  5 :
---> Number of Nodes:  6
---> States included:  ['px']
Cluster  6 :
---> Number of Nodes:  6
---> States included:  ['ca']
Cluster  7 :
---> Number of Nodes:  2
---> States included:  ['fl']
Cluster  8 :
---> Number of Nodes:  4
---> States included:  ['ut', 'ca', 'az', 'nv']
Cluster  9 :
---> Number of Nodes:  7
---> States included:  ['hi']
Cluster  10 :
---> Number of Nodes:  2
---> States included:  ['az', 'nm']
```

共产生 85 个簇，可以看到，很多簇都只包含 1 到 2 个地区，聚类的离散度较大。其中还发现 outlier 数目过多，原因在于数据稀疏度过大，导致绝大多数情况下节点很难找到足够多的邻居节点来通过 SNN 算法的检验，使得相似矩阵中充斥着相似度为 0 的节点对。

使用改进的距离算法后，结果如下（其他参数不变）：

```
Cluster Number:  10
Outlier Number:  34652
Cluster  0 :
---> Number of Nodes:  45
---> States included:  ['va', 'ab', 'co', 'ca', 'al', 'ar', 'il', 'ga', 'in', 'ia', 'az', 'id', 'sk', 'nj', 'nm', 'tx', 'la', 'nc', 'nd', 'ne', 'tn', 'pa', 'nt',
Cluster  1 :
---> Number of Nodes:  3
---> States included:  ['mt', 'wy']
Cluster  2 :
---> Number of Nodes:  22
---> States included:  ['va', 'ca', 'al', 'bc', 'ar', 'vt', 'il', 'ga', 'in', 'ia', 'ct', 'nh', 'ni', 'tx', 'la', 'nb', 'nc', 'ne', 'tn', 'ny', 'pa', 'pe', 'ns',
Cluster  3 :
---> Number of Nodes:  3
---> States included:  ['co', 'nm', 'ny', 'ut', 'wy']
Cluster  4 :
---> Number of Nodes:  30
---> States included:  ['va', 'vi', 'al', 'dc', 'ar', 'il', 'ga', 'in', 'fl', 'pr', 'tx', 'la', 'mo', 'tn', 'hi', 'ms', 'ky']
Cluster  5 :
---> Number of Nodes:  16
---> States included:  ['dengl', 'ab', 'co', 'wa', 'bc', 'ak', 'gl', 'vt', 'wy', 'on', 'lb', 'mb', 'fraspm', 'nf', 'or', 'sk', 'qc', 'nt', 'nu']
Cluster  6 :
---> Number of Nodes:  2
---> States included:  ['va', 'mn', 'md', 'wi', 'wv', 'vt', 'il', 'ga', 'in', 'ia', 'ct', 'nh', 'on', 'nj', 'ma', 'oh', 'nb', 'nc', 'mi', 'tn', 'ny', 'qc', 'pa',
Cluster  7 :
---> Number of Nodes:  4
---> States included:  ['va', 'md', 'de', 'al', 'me', 'wv', 'ar', 'il', 'in', 'ct', 'nh', 'on', 'nj', 'ma', 'oh', 'nc', 'mi', 'tn', 'ny', 'qc', 'pa', 'sc', 'ky']
Cluster  8 :
---> Number of Nodes:  2
---> States included:  ['on', 'lb', 'mb', 'bc', 'nf', 'sk', 'qc', 'gl', 'nt', 'nu']
Cluster  9 :
---> Number of Nodes:  2
---> States included:  ['ab', 'co', 'bc', 'mb', 'id', 'sk', 'mt', 'nt', 'yt']
```

簇的集中度明显提升，共产生 10 个簇。但在 outlier 过多的问题上，可以看出 outlier 数目有所减少，但依旧过多。因此，为了使聚类结果更加理想，我提高了邻居节点的数目，即 K 值，来增加两节点共同邻居重合的概率。在以下实验中，我将 KNN_Param 值设置为 200，使用改进的距离算法，SNN 阈值参数依旧为 0.5，程序总共运行时间 17 小时 30 分钟，聚类结果如下：

```
2014年 06月 28日 星期六 16:30:12 CST
Load Data Done.
Graph Generated!
Cluster Number:  8
Outlier Number:  32687
Cluster  0 :
---> Number of Nodes:  2075
---> States included:  ['co', 'va', 'ca', 'ga', 'gl', 'ct', 'pr', 'lb', 'tx', 'la', 'tn', 'pa', 'pe', 'mn', 'de', 'dc', 'hi', 'vt', 'me', 'md', 'ma', 'mb', 'ut', 'n
Cluster  1 :
---> Number of Nodes:  2
---> States included:  ['ab', 'co', 'ak', 'vt', 'il', 'ia', 'mn', 'id', 'ct', 'nh', 'nj', 'lb', 'nm', 'nb', 'nd', 'nf', 'ny', 'pa', 'pe', 'ns', 'nt', 'ri', 'nv', 'n
Cluster  2 :
---> Number of Nodes:  4
---> States included:  ['ks', 'ar', 'ok', 'il', 'la', 'mo', 'in']
Cluster  3 :
---> Number of Nodes:  3
---> States included:  ['oh', 'mn', 'mi', 'wi', 'pa', 'il', 'in', 'ia']
Cluster  4 :
---> Number of Nodes:  2
---> States included:  ['va', 'ab', 'co', 'vt', 'il', 'in', 'ia', 'mn', 'id', 'ct', 'nh', 'nj', 'nm', 'nb', 'nd', 'nf', 'ny', 'pa', 'pe', 'ns', 'nt', 'ri', 'wa', 'h
Cluster  5 :
---> Number of Nodes:  4
---> States included:  ['va', 'al', 'ar', 'vt', 'il', 'ga', 'in', 'ia', 'ct', 'nh', 'nj', 'la', 'nc', 'nd', 'tn', 'ny', 'pa', 'ri', 'md', 'de', 'dc', 'wi', 'wv', 'c
Cluster  6 :
---> Number of Nodes:  2
---> States included:  ['va', 'co', 'wa', 'vt', 'ct', 'nh', 'on', 'ma', 'oh', 'ut', 'mi', 'ny']
Cluster  7 :
---> Number of Nodes:  2
---> States included:  ['co', 'wa', 'vt', 'il', 'in', 'ct', 'me', 'on', 'nj', 'ma', 'mi', 'nf', 'ny', 'qc', 'pa', 'ns', 'nt', 'or']
2014年 06月 29日 星期日 10:03:38 CST
```

可以看到，提高 K 参数起了作用，聚类结果更为集中，只产生了 8 个簇，同时 outlier 的数量也有明显的减少，但聚类效果仍旧不理想。相比之下，即便是增加到了 200 个邻居，每个簇的邻居数目占整个数据集的比例还是只有不到 0.5%，因此，为了增大共同邻居的重合概率，需要提升的不是 K 参数，而是 K 占整个数据集的比例。

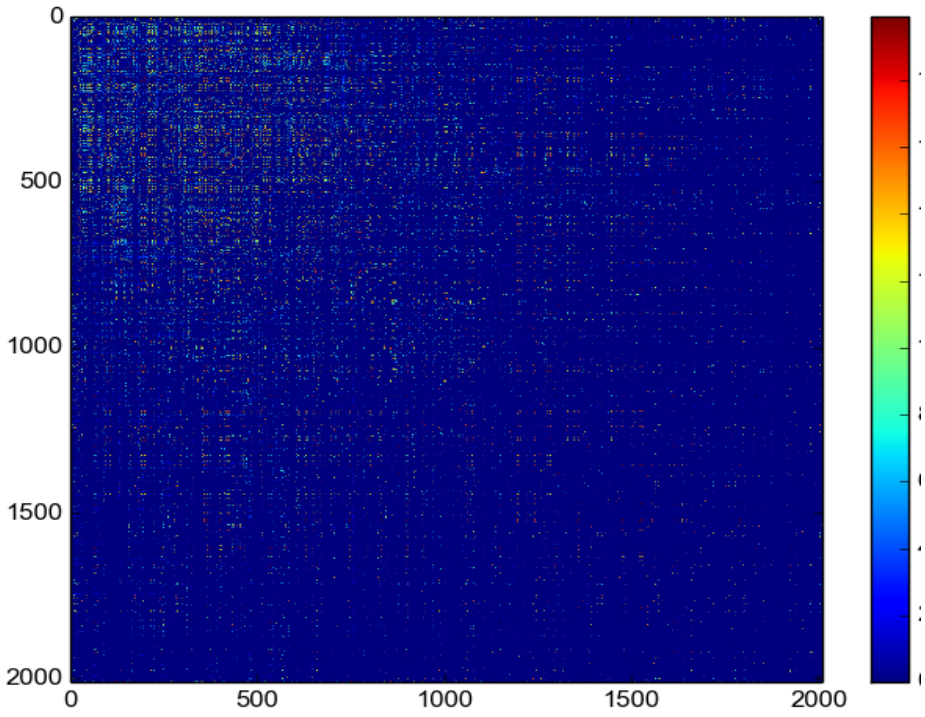
在最后一组实验中，我将 K 的比例提升到 10%，即 KNN_Param / 所有数据个数 = 10%，发现，得到有效聚类的数据数目增加到了近 80%，outlier 的数目下降到了 20%左右。聚类结果如下：

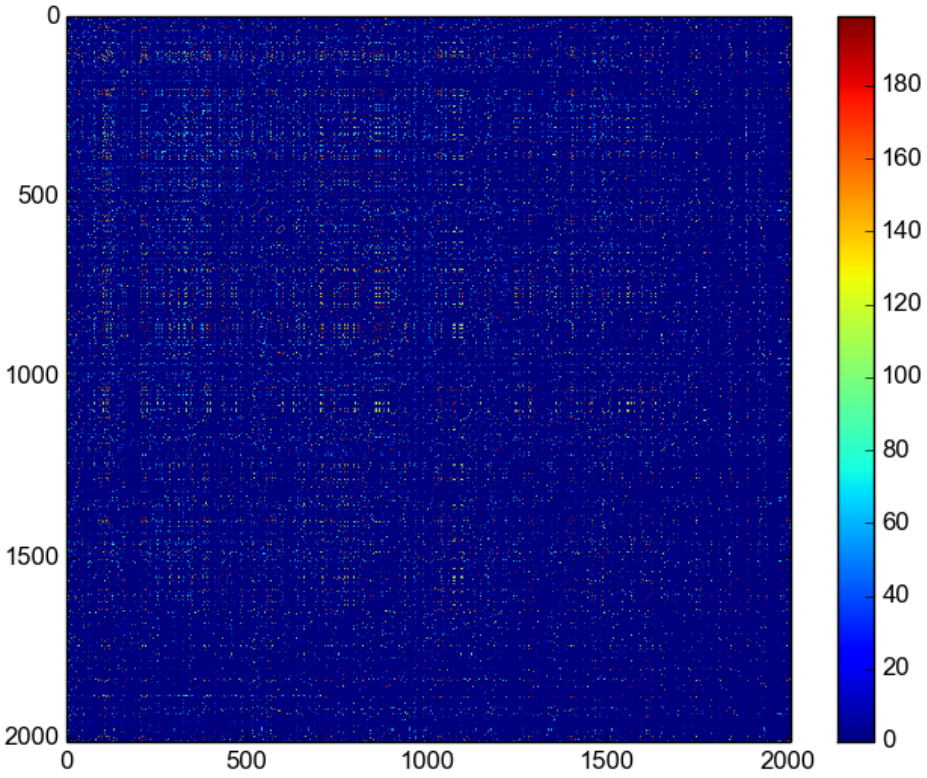
```
Cluster Number:  7
Outlier Number:  446
Cluster  0 :
---> Number of Nodes:  1544
---> States included:  ['co', 'dengl', 'ca', 'ga', 'gl', 'ct', 'pr', 'lb', 'tx', 'la', 'tn', 'pa', 'pe', 'mn', 'de', 'dc', 'hi', 'yt', 'me', 'md', 'ma', 'mb', 'ut',
Cluster  1 :
---> Number of Nodes:  3
---> States included:  ['mi', 'ca', 'wi', 'vt', 'il', 'ia', 'ct', 'nh', 'on', 'nj', 'ma', 'oh', 'ut', 'nb', 'mb', 'nd', 'nf', 'ny', 'qc', 'pa']
Cluster  2 :
---> Number of Nodes:  10
---> States included:  ['co', 'ca', 'ga', 'gl', 'ct', 'pr', 'lb', 'tx', 'la', 'tn', 'pa', 'pe', 'de', 'dc', 'hi', 'yt', 'me', 'md', 'ma', 'mb', 'ut', 'mo', 'mn', 'm
Cluster  3 :
---> Number of Nodes:  2
---> States included:  ['va', 'co', 'ak', 'ar', 'vt', 'il', 'ga', 'in', 'ia', 'id', 'ct', 'nh', 'nj', 'nb', 'nc', 'nd', 'ne', 'tn', 'ny', 'pa', 'ns', 'wa', 'bc', 'd
Cluster  4 :
---> Number of Nodes:  2
---> States included:  ['me', 'ut', 'ca', 'il', 'in', 'ia']
Cluster  5 :
---> Number of Nodes:  2
---> States included:  ['ab', 'co', 'vt', 'il', 'in', 'ia', 'az', 'id', 'ct', 'nh', 'ks', 'nj', 'nm', 'nd', 'ne', 'ny', 'pa', 'ri', 'mn', 'bc', 'wi', 'wy', 'me', 'd
Cluster  6 :
---> Number of Nodes:  2
---> States included:  ['va', 'mn', 'ca', 'vt', 'il', 'in', 'ct', 'nh', 'nj', 'lb', 'nb', 'nc', 'nf', 'ny', 'pa', 'pe', 'ns', 'ri', 'wa', 'bc', 'de', 'wi', 'me', 'd
```

通过距离矩阵，和相似矩阵生成的可视化矩阵图如下：

（上方的图为距离矩阵生成的可视化图，下方的图为相似矩阵生成的可视化图）

由图可见，即便是经过处理的数据，稀疏度依旧很高，因此 K-Means 等算法不适用。




## 五、 遇到问题及解决方式

在程序运行的过程中，算法的效率是主要的瓶颈。其中有很大一部分原因源自我使用的语言。

在一开始生成距离矩阵时就发现我的笔记本电脑根本跑不动，而在后续的操作中，程序的内存占用量更是直线上升。造成内存使用率过高的罪魁祸首就是 Python 里的字典。由于 JP 算法中需要大量使用查表方法，因此，我使用了 Python 中的字典数据结构来简化查表的过程，但在 Python 的机制中，一个字典不仅仅只是建立了一个数据表，它还为每条记录计算了一个哈希值作为索引。因此，由于数据数量庞大，导致矩阵庞大，进而导致哈希索引表的庞大。而这一切数据都需要保存在内存中，因此，频繁的内存页换入换出最终拖垮了我的电脑。

最终解决方法是，我把聚类程序放到服务器上跑，利用服务器的大内存减少内存页的交换。

由于在服务器上运行程序，从而引发了我的第二个问题。我通过 ssh 连接访问服务器，并运行程序，因此对于图形化的支持很差。而在我的实验过程中，生成可视化矩阵的那一步必须图形化界面的支持。对此，生成可视化矩阵的那一步分解成两个函数，ServerSide 和 ClientSide。ServerSide 是程序在服务器上运行时调用的函数，它将距离矩阵和相似矩阵导出到文件。而我可以将该文件下载到本地电脑，之后通过调用 ClientSide 函数，从文件中间距离矩阵和相似矩阵导入到内存中，再进行可视化的处理。

## 六、 附录文件

**Clustering.py**：聚类程序文件
**Plants.data**：原始数据文件
**Result/result_Origin_20.txt**：使用原始汉明距离算法的结果文件
**Result/result_20.txt**：使用改进距离算法的结果文件
**Result/result_200.txt**：提高邻居数目后的结果文件
**Result/result_200_in_2000.txt**：提高邻居数目比例后的结果文件