实验五：层次聚类

姓名：孙铭

学号：1711377

专业：计算机科学与技术

日期：2020年5月8日

目录

摘要

本次作业主题是层次聚类，实验涉及到的基本、中级、高级三类要求已全部实现。在此声明本次作业数据集，代码及实验报告均由个人独立完成，本次实验实现的具体功能参下。

> *数据集要求：*
>
> > *本次实验使用数据集为基于sklearn包，scikit中的make_blobs方法所生成的数据集，该方法常被用来生成聚类算法的测试数据。数据集包含2000个测试样例。每个测试样例的前三列表示特征，第四列表示标签。*
>
> *实验基本要求：*
>
> 1. *实现了single-linkage层次聚类算法，并绘制图像展示实验结果。*
> 2. *实现了complete-linkage层次聚类算法，并绘制图像展示实验结果。*
>
> *实验中级要求：*
>
> 3. *实现了average-linkage层次聚类算法，并绘制图像展示实验结果。*
>
> *实验提高要求：*
>
> 4. *对比上述三种算法，给出实验结论。*
> 5. *通过变换聚类簇的个数（$k = 3, 4, 5$），测试上述三种算法的性能，并给出实验结果图及相应分析。*

以上是本次实验摘要部分内容，接下来按阶段展开讲述。

环境配置

本次作业使用语言版本为`Python 3.6.5 64bit`，处理器为`Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80GHz`，项目工程文件如下。

```
| data                    # 数据集存储文件
|   | data_x.py           # 样本特征存储文件
|   | data_y.py           # 样本标签存储文件
| pic                     # 实验结果图存储文件
| hie_clustering.py       # 层次聚类算法实现
```

实验过程中用到的包如下。

```
import os
import numpy as np
from matplotlib import pyplot as plt
from sklearn.datasets.samples_generator import make_blobs
```

以上是本次实验环境配置部分。

数据处理

本次实验要求自己生成数据集，考虑到聚类算法对数据集要求的特殊性，我选择利用`sklearn.datasets.samples_generator`中的`make_blobs`方法实现数据集的生成。

`scikit`中的`make_blobs`方法常被用来生成聚类算法的测试数据，直观地说，`make_blobs`会根据用户指定的特征数量、中心点数量、范围等来生成几类数据，这些数据可用于测试聚类算法的效果。`make_blobs`方法如下。

```
sklearn.datasets.make_blobs(n_samples=100,
n_features=2,centers=3, cluster_std=1.0, center_box=
(-10.0, 10.0), shuffle=True, random_state=None)[source]
```

其中各参数的含义为:

```
n_samples         # 待生成的样本总数
n_features        # 每个样本的特征数
centers           # 类别数
cluster_std       # 每个类别的方差
center_box        # 中心确认之后的数据边界
shuffle           # 随机打乱
random_state      # 设置随机数种子防止每次生成数据都修改
```

这里我们将生成数据的方法封装成函数，代码如下。

```
def generate_samples(count, centers, std):
    X, Y = make_blobs(n_samples=count, centers=centers,
cluster_std=std)
    np.savetxt(os.path.dirname(__file__) +
'/data/data_x.dat', X)
    np.savetxt(os.path.dirname(__file__) +
'/data/data_y.dat', Y)
    print("Finished generating samples in /data/data_x.dat
and /data/data_y.dat")
    return X, np.array(Y, dtype=int)
```

在调用函数生成数据时，各参数初始化情况如下。

```
count = 2000
centers = [[1,1,1],[1,3,3],[3,6,5],[2,6,8]]
std = 0.5
```

于是，我们可以得到数据总量为2000，有四个样本中心点，方差为0.5的数据集样本。以上是本次实验数据处理部分，接下来将对算法实现进行讲述。

实验基本要求

层次聚类试图在不同层次对数据集进行划分，从而形成树形的聚类结构，数据集的划分可采用自底向上的聚合策略或自顶向下的聚合策略。

本次实验我采用的是AGNES算法，AGNES算法是一种采用自底向上聚合策略的层次聚类算法。该算法先将数据集中每个样本看作一个初始数据簇，然后在算法运行时的每一步中找出距离最近的两个聚类簇进行合并，该过程不断重复，直到达到预设的聚类簇个数。这里的关键是如何计算聚类簇之间的距离。而实际上，每个簇是一个样本集合，因此，只需采用关于集合的某种距离即可。

关于计算距离的方法，有三种方法进行计算，分别是*single linkage(最小距离)*、*complete linkage(最大距离)*、*average linkage(平均距离)*，关于实验基础部分只要求实现前两种，因此这里给出前两种距离的计算公式如下。

$$\text{最小距离}: d_{min}(C_i, C_j) = min_{x \in C_i, z \in C_j} dist(x, z)$$
$$\text{最大距离}: d_{max}(C_i, C_j) = max_{x \in C_i, z \in C_j} dist(x, z)$$

显而易见，最小距离由两个聚类簇的最近样本决定，最大距离由两个聚类簇的最远样本决定。这里关于距离计算方法`dist()`，我实现了两种方法，一种是欧氏距离，另一种是豪斯多夫距离，但豪斯多夫距离在对本实验的计算结果并不理想，可以使用`hie_clustering.py/dist_H()`函数来证实这一结果（注：豪斯多夫距离的代码实现并没有删掉，而是放在了作业源码中，以保留我个人的一些思考和探索过程）。

基于欧氏距离的*single linkage(最小距离)*、*complete linkage(最大距离)*实现如下。

```
def dist_emin(c_i, c_j):
    return min([np.sum((x - y) ** 2) for x in c_i for y in c_j])
def dist_emax(c_i, c_j):
    return max([np.sum((x - y) ** 2) for x in c_i for y in c_j])
```

接下来实现AGNES算法，该算法的伪代码参下。

```
输入：样本集D = {x1,x2,...,xm}
     聚类簇距离度量函数d
     聚类簇数k
过程：
for j = 1,2,...,m do
    Cj = {xj}
```

```
end for
for i = 1,2,...,m do
    for j = i + 1,...,m do
        M(i,j) = M(j,i) = d(Ci,Cj)
    end for
end for
设置当前聚类簇个数：q = m
while q > k do
    找出距离最近的两个聚类簇Ci*和Cj*
    合并Ci*和Cj*
    for j = j* + 1, j* + 2,...,q do:
        将聚类簇Cj重编号为Cj-1
    end for
    删除距离矩阵M的第j*行与第j*列
    for j = 1,2,...,q-1 do
        M(i*,j) = M(j,i*) = d(Ci*,Cj)
    end for
    q = q - 1
end while
输出：簇划分C = {C1,C2,...,Ck}
```

算法细节已在伪代码中给出，这里直接贴出算法实现代码如下。

```
def AGNES(X, Y, dist, k): # dist:距离度量函数 k:聚类簇数
    N = X.shape[0]
    CIndex = [[i] for i in range(N)] # 点的索引标号
    C = [[x] for x in X] # 初始化单样本聚类簇 C: [[array()]]

    # 初始化聚类簇距离矩阵
    MAX_DIS = 1e3
    M = np.zeros((N, N)) + MAX_DIS
    for i in range(N):
        for j in range(i + 1, N):
            M[i][j] = M[j][i] = dist(C[i], C[j])

    cluster_count = X.shape[0] # 当前聚类簇个数
    while cluster_count > k:
        _index = np.argmin(M)
        ci_index, cj_index = int(_index / cluster_count),
_index % cluster_count
        C[ci_index] += C[cj_index]
        CIndex[ci_index] += CIndex[cj_index]
        del C[cj_index]
        del CIndex[cj_index]
        M = np.delete(M, cj_index, axis=0)
        M = np.delete(M, cj_index, axis=1)
        if ci_index > cj_index:
            ci_index -= 1
        for j in range(M.shape[0]):
            if j != ci_index:
```

```
                M[ci_index][j] = M[j][ci_index] =
dist(C[ci_index], C[j])
        cluster_count -= 1
    return C, CIndex
```

以上是本次实验基本要求部分。

实验中级要求

本次实验中级要求部分比较简单，要求实现*average linkage(平均距离)*，可以知道平均距离由两个聚类簇的所有样本决定。距离计算公式为：

$$\text{平均距离}: d_{avg}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{z \in C_j} dist(x, z)$$

算法实现代码如下。

```
def dist_eave(c_i, c_j):
    d_sum = sum([np.sum((x - y) ** 2) for x in c_i for y
in c_j])
    return d_sum / (len(c_i) * len(c_j))
```

需要指出的一点是，关于*single linkage(最小距离)*、*complete linkage(最大距离)*、*average linkage(平均距离)*实验结果实验结果的展示请参见后文的分析。

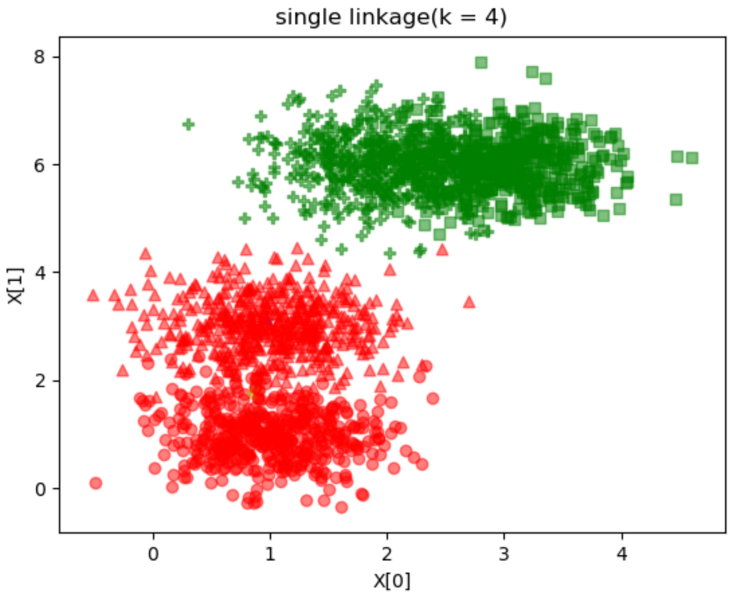
以上是本次实验中级要求部分。

实验高级要求

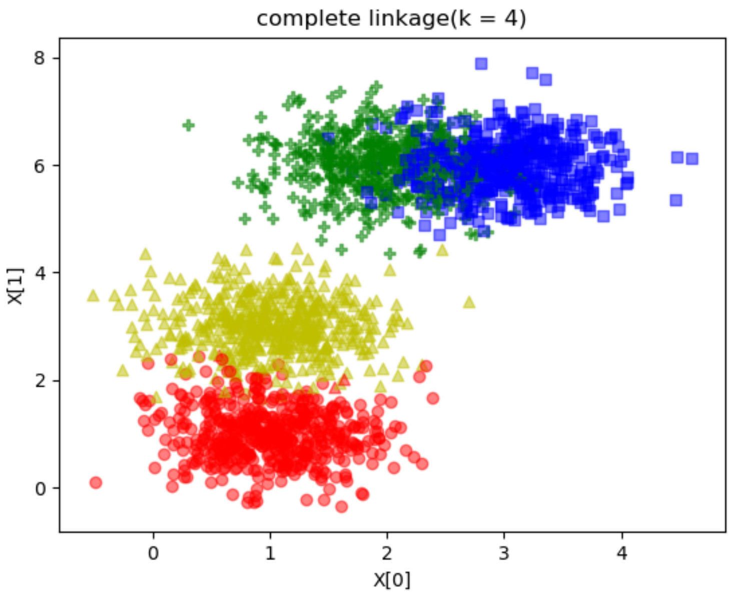
本次实验高级要求部分有两项任务，首先是对比*single linkage(最小距离)*、*complete linkage(最大距离)*、*average linkage(平均距离)*三种算法，给出相应的结论；其次是通过变换聚类簇的个数，测试上述三种算法的性能，并给出分析结果。这里我通过实验结果的图像表示展开分析。

首先，我以`k=4`为例，对应的*single linkage(最小距离)*、*complete linkage(最大距离)*、*average linkage(平均距离)*三种算法实验结果图如下。

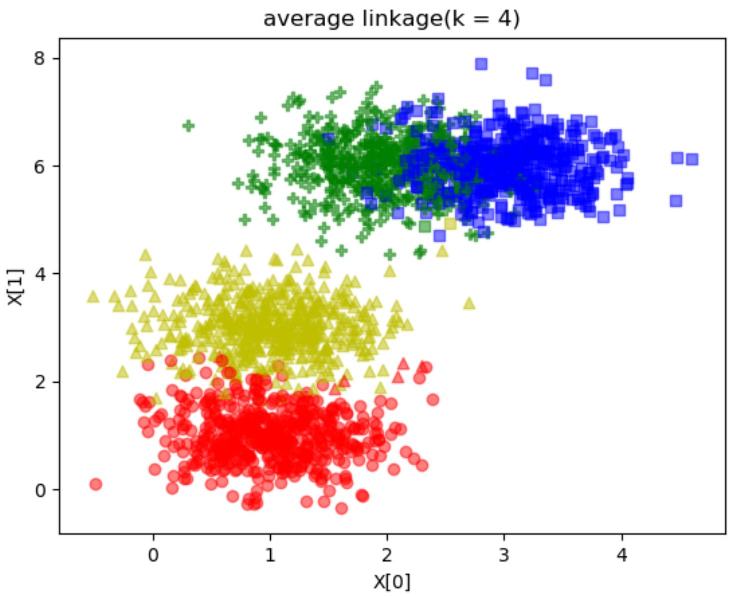
single linkage：



complete linkage:



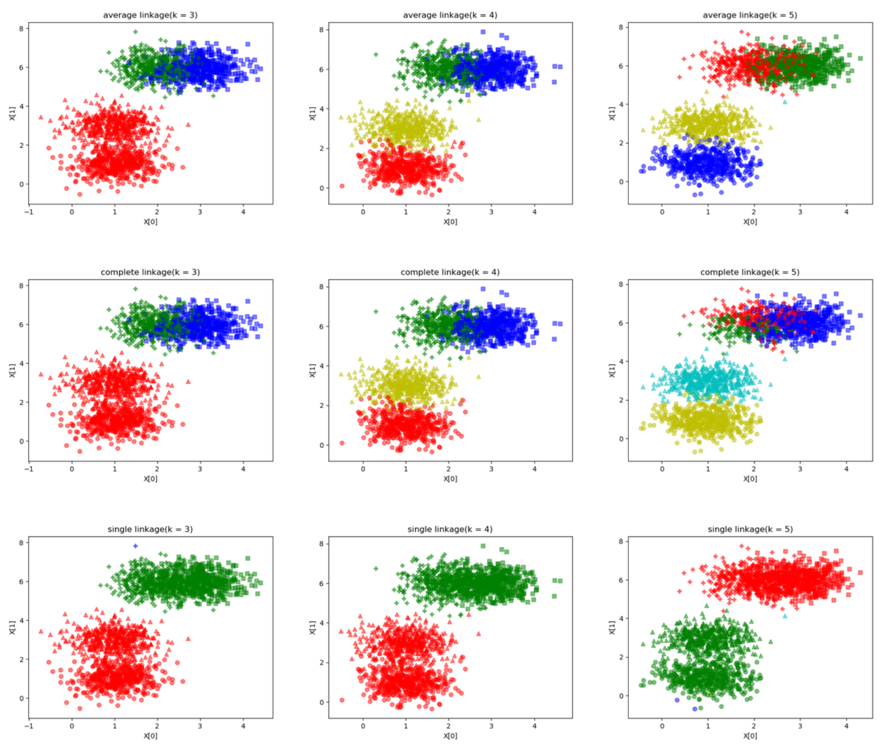
average linkage:



从图像中可以看出，基于*single linkage*实现的层次聚类算法在数据集上的表现最差，原本样本标签为四类，在聚类回归时几乎只聚成了两类。而*complete linkage*、*average linkage*算法在数据集上的表现最好，两者的聚类结果基本一致。

接下来是比较当聚类簇个数不同时，对上述三种算法性能的测试，这里我以聚类簇数量为3、4、5为例进行对比论证，以更好地进行分析。

下面是三种算法在`k=3、4、5`时聚类结果表现情况散点图。



对比上述三类算法在不同聚类簇数下的实验结果图，可以看出，无论是聚类簇数k取值为多少，*single linkage*算法的聚类结果在数据集上的表现都是最差的，而且聚类结果对聚类簇数k的值不敏感；而对比*complete linkage*算法和*average linkage*算法，可以看出随着k值的增加，*complete linkage*算法表现出对聚类簇数的高度敏感性，聚类结果也随之改变，相比于*average linkage*算法误差更大。而*average linkage*算法对聚类簇数k的值不敏感，聚类结果则更加精确。

因此，在三类算法中，当聚类簇数k偏离样本真实聚类簇数量时，使用平均距离实现的层级聚类，即*average linkage*算法的聚类结果是最精确的。

以上是本次实验高级要求部分。

以上是本次实验报告内容，感谢批阅！
