

一、实验要求

1、每人完成聚类算法 K-Means 的代码编写，并在数据集上测试，聚类结果需要采用不少于 1 种可视化方法表示；

2、完成最基本的 K-Means 算法及 1 种可视化表示成绩为 70-85 分，完成二分 K-Means 算法并进行分步骤过程可视化成绩最高可以到 95 分，对已有的 K-Means 算法（或可视化方法）自行做改进或优化成绩最高可以到 100 分；

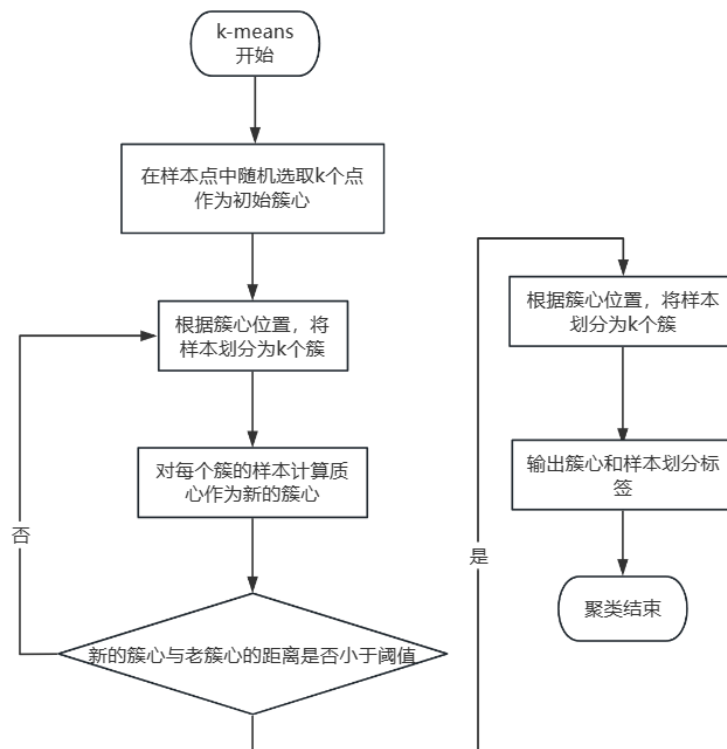
3、完成算法所用的语言不限；

4、测试数据集可以自行构造，也可以采用 UCI 上的公开数据集；

5、提交作业包括：1）源代码， 2）可执行文件， 3）测试样例；4）实验报告：说明采用的算法，主要的函数、测试说明、改进之处（如有）、心得体会等。

二、实验原理

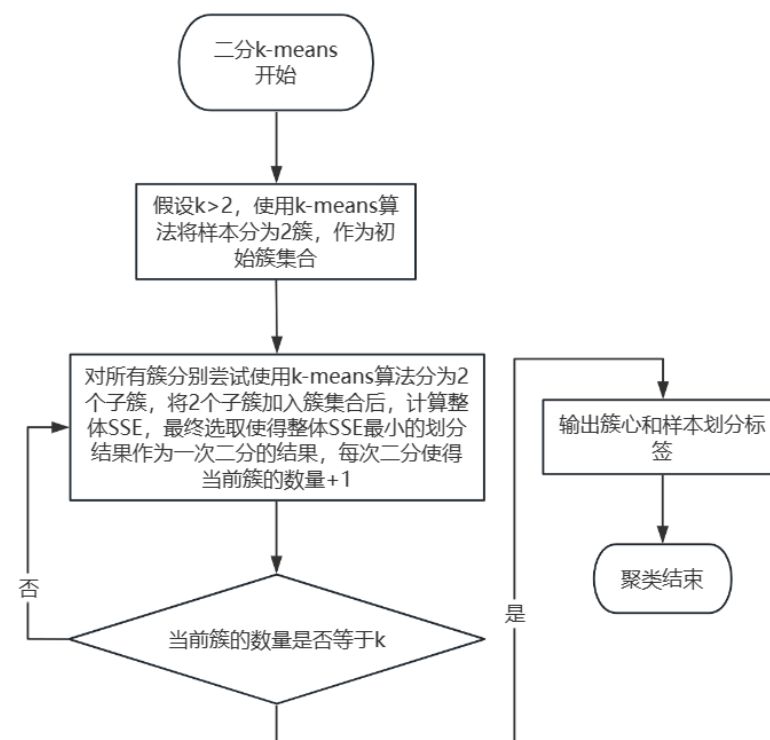
1. K-means 算法：



在本次实验的程序中，除了收敛阈值外，还限制了迭代次数（限制为 200），用于演示

聚类过程的 K-means 算法只会尝试一次聚类,用于二分 K-means 算法的 K-means 算法会尝试 5 次聚类, 选取其中 SSE 最小的作为最终结果。

2. 二分 K-means 算法:



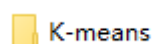
本次实验程序采用的是使整体 SSE 最优的二分,有的算法会直接选取当前 SSE 最大的簇来进行划分。

3. 可视化相关算法:

使用 PCA 算法将高维数据降维到 2 维, 在平面坐标系中展示聚类过程和结果, 值得注意的是 PCA 会使得簇心位置发生偏移, 使用原簇心降维后得到的簇心不利于展示, 因此绘图时需要再次计算降维后每个簇的簇心。

三、主要函数、文件说明

程序使用 Python 开发, IDE 为 Pycharm, OS 为 Win10
项目文件夹为:



K-means 文件夹下的文件有

.idea	2024-04-19 12:57	文件夹	
.venv	2024-04-15 15:04	文件夹	
__pycache__	2024-04-19 12:18	文件夹	
build	2024-04-19 12:05	文件夹	
dataset	2024-04-19 12:29	文件夹	
bi_k_means.py	2024-04-19 11:31	PY 文件	5 KB
data_process.py	2024-04-19 12:57	PY 文件	1 KB
k_means.py	2024-04-19 12:18	PY 文件	6 KB
log.txt	2024-04-18 12:17	文本文档	1 KB
main.exe	2024-04-19 12:24	应用程序	69,722 KB
main.py	2024-04-19 12:15	PY 文件	5 KB
main.spec	2024-04-19 12:23	SPEC 文件	1 KB
test.py	2024-04-19 8:46	PY 文件	1 KB
tools.py	2024-04-18 10:09	PY 文件	2 KB

其中可忽略文件夹与文件为：

文件夹：.idea，.venv，__pycache__，build

文件：log.txt，main.spec

文件说明：

1. 文件夹 dataset：

dataset 文件夹用于存放默认数据集，可供测试的数据集有（格式已统一）：

iris.data	2023-05-22 22:20	DATA 文件	5 KB
iris_top1.data	2024-04-19 11:17	DATA 文件	1 KB
iris_top2.data	2024-04-19 11:17	DATA 文件	1 KB
iris_top3.data	2024-04-19 11:17	DATA 文件	1 KB
wine.data	2023-05-22 22:22	DATA 文件	11 KB
wq.data	2024-04-19 11:53	DATA 文件	341 KB

程序会提供更改数据集路径的选项：

2. 可执行文件 main.exe：

需要注意的是，运行后将出现控制台，需要等待数秒后按回车出现用户交互提示

源代码说明：

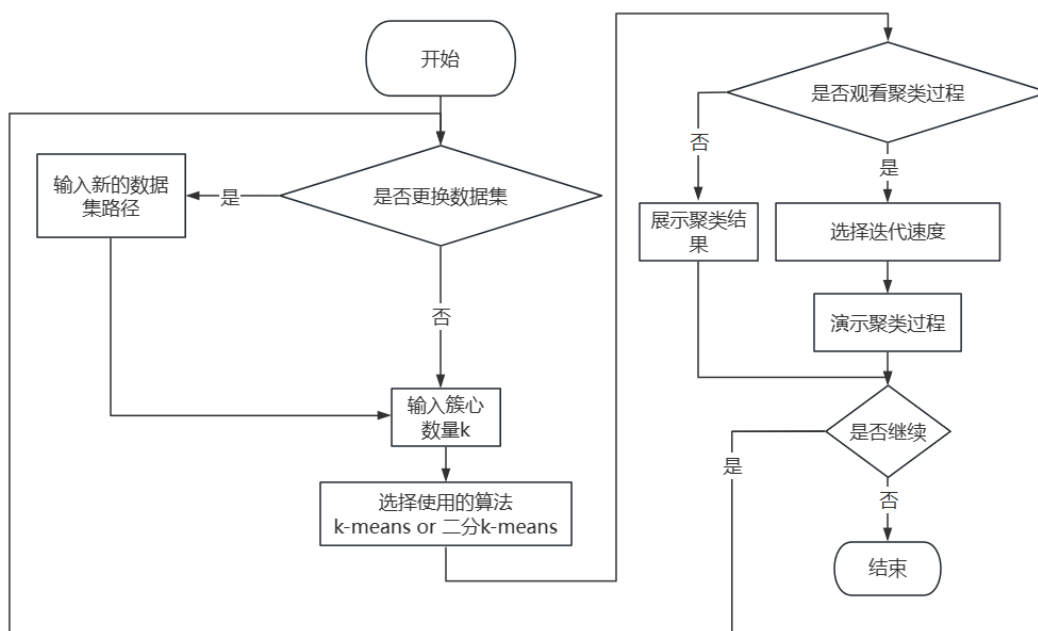
1. main.py：

主函数所在文件，包含如迭代次数限制等基本参数，提供用户交互

主要函数：main()，负责用户交互逻辑管理

```
def main():
    plt.ion()
    print(note)
    sys.stdout.flush()
    data_dir = default_data_dir
    while True:
        # 数据集选择
        input_str = "当前数据集为：" + data_dir
        print(input_str)
        input_str = "是否更换数据集？输入1是，输入0否：\n"
```

本程序用户交互逻辑为：



2. data_process.py:

包含数据处理函数，用于导入数据集为 np.array 类型数据

主要函数：get_data_feature(datadir)

```
# 读取数据集，返回属性数据矩阵（去掉最后一列label）
1 usage
def get_data_feature(datadir):
    f = open(datadir)
    datastr = f.read()
    f.close()
    dataraw = datastr.strip('\n').split('\n')
    row = len(dataraw)
    col = len(dataraw[0].strip('\n').split(','))
    datafeature = np.zeros(shape=(row, col-1), dtype=float)
    count = 0
    for line in dataraw:
        tmp = line.strip('\n').split(',')
        datafeature[count,:] = list(map(float, tmp[0:col-1]))
        count += 1
    return datafeature
```

3. tools.py:

包含用户输入规范函数以及调试相关函数

主要函数：暂停函数 pause()，整数输入函数 int_get(low, up, input_str)

4. k_means.py:

包含 k-means 算法相关函数

主要函数:

(1) get_sse(data, label, center)

计算 SSE 的函数

输入:

data: 样本

label: 样本划分标签, 同一簇的样本具有相同的标签

center: 聚类簇心

输出: 该聚类的 SSE

```
# 计算sse函数
8 usages
def get_sse(data, label, center):
    k = center.shape[0]
    sse = 0
    for i in range(k):
        data_i = data[i == label]
        dst = np.linalg.norm(data_i - center[i, :], axis=1)
        sse += np.sum(dst, axis=0)
    sse = round(sse, 2)
    return sse
```

(2) k_means_init_center(data, k)

簇心初始化函数

输入:

data: 样本

k: 需要初始化的簇心数

输出:

初始化的簇心 center

```
# 从数据集中随机选择k个点作为初始簇心
2 usages
def k_means_init_center(data, k):
    n = data.shape[0]
    if k > n:
        center = data[np.random.choice(data.shape[0], n, replace=False)]
    else:
        center = data[np.random.choice(data.shape[0], k, replace=False)]
    return center
```

(3) k_means_get_label(data, center)

根据簇心位置, 获取样本划分标签函数

输入:

data: 样本

center: 簇心

输出:

样本划分标签 center

```
# 计算数据与各个簇心的距离，并且返回每个样本离得最近簇序号（从0开始）
5 usages
def k_means_get_label(data, center):
    dst = np.linalg.norm(data[:, np.newaxis] - center, axis=2)
    label = np.argmin(dst, axis=1)
    return label
```

(4) k_means_get_center(data, label, k)

根据样本划分，计算新的簇心

输入：

data: 样本
label: 样本划分标签
k: 簇心个数

输出：

新的簇心 center

```
# 根据当前的簇划分，计算新的簇心
3 usages
def k_means_get_center(data, label, k):
    center = np.array([data[label == i].mean(axis=0) for i in range(k)])
    return center
```

(5) k_means(data, k, T, E)

k-means 算法函数

输入：

data: 样本
k: 簇心个数
T: 最大迭代次数
E: 收敛误差

输出：

聚类结果的簇心 center，样本划分标签 label，以及迭代次数 t

```
# k-means算法，迭代直到收敛，T为最大迭代次数，E为收敛误差
def k_means(data, k, T, E):
    n = data.shape[0]
    > if n == 1: ...
    center = k_means_init_center(data, k)
    t = 0
    > while t < T: ...
    label = k_means_get_label(data, center)
    return center, label, t
```

(6) k_means_with_try(data, k, T, E, try_num)

在 k-means 函数的基础上，尝试多次聚类，选取 SSE 最小的聚类结果

输入：

data: 样本
k: 簇心个数
T: 最大迭代次数
E: 收敛误差
try_num: 尝试次数（默认设置为 5）

输出:

聚类结果的簇心 center, 样本划分标签 label, 以及迭代次数 t

```
# 会尝试不同初始随机点的k-means算法, 选取SSE最小的结果返回
4 usages
> def k_means_with_try(data, k, T, E, try_num):...
```

(7) draw(data, k, label, p_time, title)

绘图函数, 将聚类结果通过 PCA 降维到 2 维后, 通过平面坐标系展示

输入:

data: 样本
k: 簇心个数
label: 样本划分标签
p_time: 绘图时间（用于控制迭代间隔时间）
title: 图像标题

输出:

绘制图形

```
# 绘图函数, 给出当前的簇标签和簇心, 绘制相应图像
6 usages
> def draw(data, k, label, p_time, title):...
```

4. bi_k_means.py:

包含二分 k-means 算法相关函数

主要函数:

bi_k_means(data, k, T, E, try_num)

二分 k-means 算法函数

输入:

data: 样本
k: 簇心个数
T: 最大迭代次数
E: 收敛误差
try_num: k-means 算法尝试聚类次数

输出:

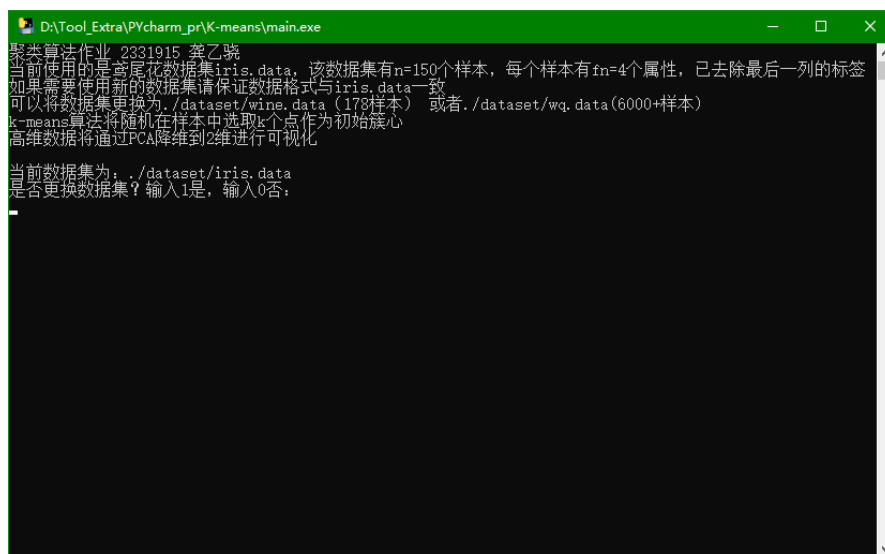
簇心 center, 划分标签 label, 重排后的样本 data

```
# 二分k_means算法函数，每次迭代都对所有簇进行二分，选取整体SSE最小的划分方式
> def bi_k_means(data, k, T, E, try_num):
```

四、测试说明

此处以二分 k-means 算法对鸢尾花数据集进行聚类为例，簇心数设置为 6：

1. 确保 main.exe 和 dataset 文件夹在同一目录中
2. 运行 main.exe，出现控制台后等待数秒，按下回车出现用户交互提示：



```
D:\Tool_Extra\PYcharm_pr\K-means\main.exe
聚类算法作业 2331915 袁乙晓
当前使用的是鸢尾花数据集iris.data, 该数据集有n=150个样本, 每个样本有fn=4个属性, 已去除最后一列的标签
如果需要使用新的数据集请保证数据格式与iris.data一致
可以将数据集更换为 ./dataset/wine.data (178样本) 或者 ./dataset/wq.data (6000+样本)
k-means算法将随机在样本中选取k个点作为初始簇心
高维数据将通过PCA降维到2维进行可视化
当前数据集为: ./dataset/iris.data
是否更换数据集? 输入1是, 输入0否:
```

3. 输入 0，选择不更换数据集

```
当前数据集为: ./dataset/iris.data
是否更换数据集? 输入1是, 输入0否:
0
请输入簇心数k(1<k<16):
```

4. 输入簇心数 6

```
请输入簇心数k(1<k<16):
6
算法选择, 输入1是k-means, 输入0是二分k-means:
```

5. 输入 0，选择二分 k-means 算法

```
算法选择, 输入1是k-means, 输入0是二分k-means:
0
是否观看算法过程? 输入1是, 输入0否:
```

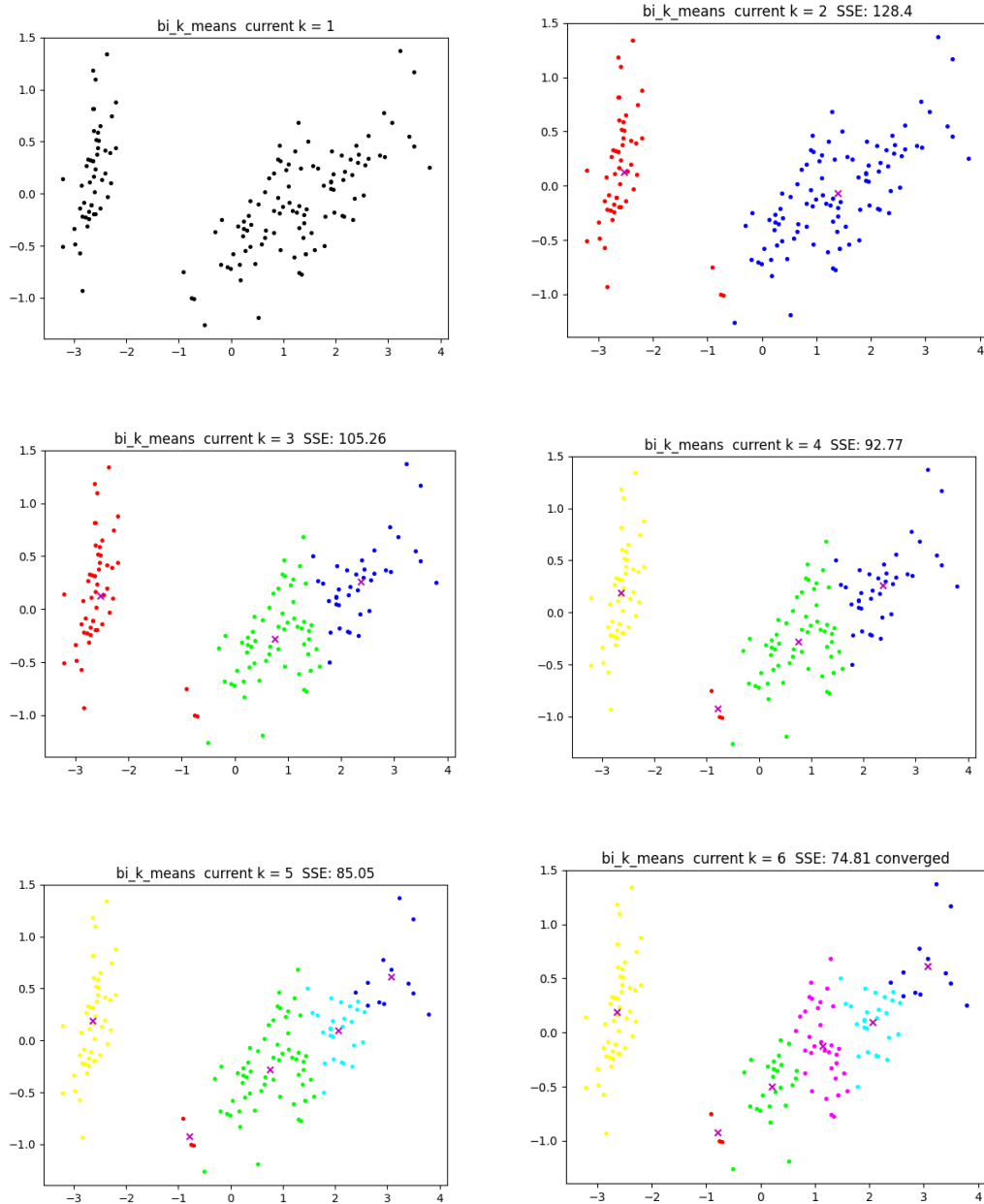
6. 输入 1，表示需要观看算法过程


```

0
是否观看算法过程？输入1是，输入0否：
1
请输入迭代间隔（单位：秒）t(1 <= t <= 3):

```

7. 输入 3，表示每次二分的间隔为 3 秒（方便截图），之后将弹出聚类图像：



终端也会给出提示

```

请输入迭代间隔（单位：秒）t(1 <= t <= 3):
3
已完成2个簇的划分
已完成3个簇的划分
已完成4个簇的划分
已完成5个簇的划分
已完成6个簇的划分
按回车键继续...

```

按回车继续

是否退出程序？输入1退出，输入0继续：

8. 输入 1，退出程序