中山大学计算机学院

人工智能

本科生实验报告

（2022学年春季学期）
课程名称：Artificial Intelligence

教学班级	专业(方向)	学号	姓名
计科三班	计算机科学与技术	21307185	张礼贤

一、实验题目

Kmeans算法实现

二、实验内容

1. 算法原理

k-means算法

- K-means算法是一种聚类算法，用于将一个数据集分成K个不同的组或类别。先从样本集中随机选取 k个样本作为簇中心，并计算所有样本与这k个“簇中心”的距离，对于每一个样本，将其划分到与其距离最近的“簇中心”所在的簇中，对于新的簇计算各个簇的新的“簇中心”其原理如下：
    - 首先随机选择K个点作为聚类中心（Centroid）。
    - 将每个数据点分配给最近的聚类中心（距离的计算通常使用欧几里得距离），形成K个聚类。
    - 对于每个聚类，计算其聚类中心。

    - 将每个数据点重新分配到离其最近的聚类中心。重新分配后，可能会出现新的聚类中心。
    - 重复步骤3和4，直到聚类中心不再变化或达到预定的迭代次数。
- 最终得到K个聚类，每个聚类中心代表该聚类

---

k-means++算法

- k-means++算法是k-means算法的一种改进，它能够更好地初始化聚类中心，从而提高算法的聚类效果。其核心思想是在聚类中心的初始化过程中，通过一定的概率分布来选择合适的初始聚类中心，使得初始聚类中心更加分散，降低了随机初始化可能带来的不好结果。下面是k-means++算法的具体步骤：
    - 从数据集中随机选择一个样本点作为第一个聚类中心。
    - 对于每一个样本点x_i，计算它与已经选择的聚类中心之间的最短距离$D(x_i)$，也就是样本点到最近聚类中心的距离。
    - 对所有样本点的$D(x_i)$进行归一化，得到概率分布$P(x_i) = \frac{D(x_i)^2}{\sum_j D(x_j)^2}$。
    - 以概率分布$P(x_i)$来选择下一个聚类中心。
    - 重复步骤2~4，直到选择了k个聚类中心。
    - 使用标准的k-means算法进行聚类。
- 由于k-means++算法在初始化聚类中心时考虑了样本点之间的距离，因此可以得到更加分散的初始聚类中心，进而获得更好的聚类效果

- 手肘分析
- 随着聚类数k的增大，样本划分会更加精细，每个簇的聚合程度会逐渐提高，那么误差平方和SSE自然会逐渐变小。
- 当k小于真实聚类数时，由于k的增大会大幅增加每个簇的聚合程度，故SSE的下降幅度会很大，

- 而当k到达真实聚类数时，再增加k所得到的聚合程度回报会迅速变小，所以SSE的下降幅度会骤减，然后随着k值的继续增大而趋于平缓
- 也就是说SSE和k的关系图是一个手肘的形状，而这个时部对应的k值就是数据的真实聚类数。

2.关键代码及注释

```
import numpy as np
import matplotlib.pyplot as plt
import random

def distance(x1, x2):
    """计算两个向量的欧氏距离"""
    x1 = np.array(x1)
    x2 = np.array(x2)
    return np.sqrt(np.sum((x1 - x2)**2))

def read_file(filename):
    """读取并分离输入文件中的信息"""
    data = np.loadtxt(filename, delimiter=',',skiprows=1, dtype=float)  #跳过第一行读取，类型为float
    return data

def classify(data, center, k):
    """分类函数,将数据集合进行分类"""
    dis = np.zeros((len(data), len(center)))  #初始化距离数组
    for i in range(len(data)):
        for j in range(len(center)):
            dis[i][j] = np.linalg.norm(data[i] - center[j]) #求出数据样本距离给定的数据中心的距离
    min_dis = np.argmin(dis, axis=1)    #返回距离最小索引的列表
    cluster = [[] for i in range(k)]
    for i in range(len(min_dis)):
        cluster[min_dis[i]].append(data[i]) #对于每一个样本中心，根据前面求得的min_dis列表的值进行归类
    new_center = []
    for i in range(k):
        new_center.append(np.mean(cluster[i], axis=0).tolist()) #分类之后再根据得到的分类结果求出每个数据集合的中心
    return cluster, new_center

def init_center(data,k):
    """随机选取样本初始中心"""
    return random.sample(data.tolist(),k)

def init_center_plus(data,k):
    """根据kmeans++选择初始样本中心"""
```

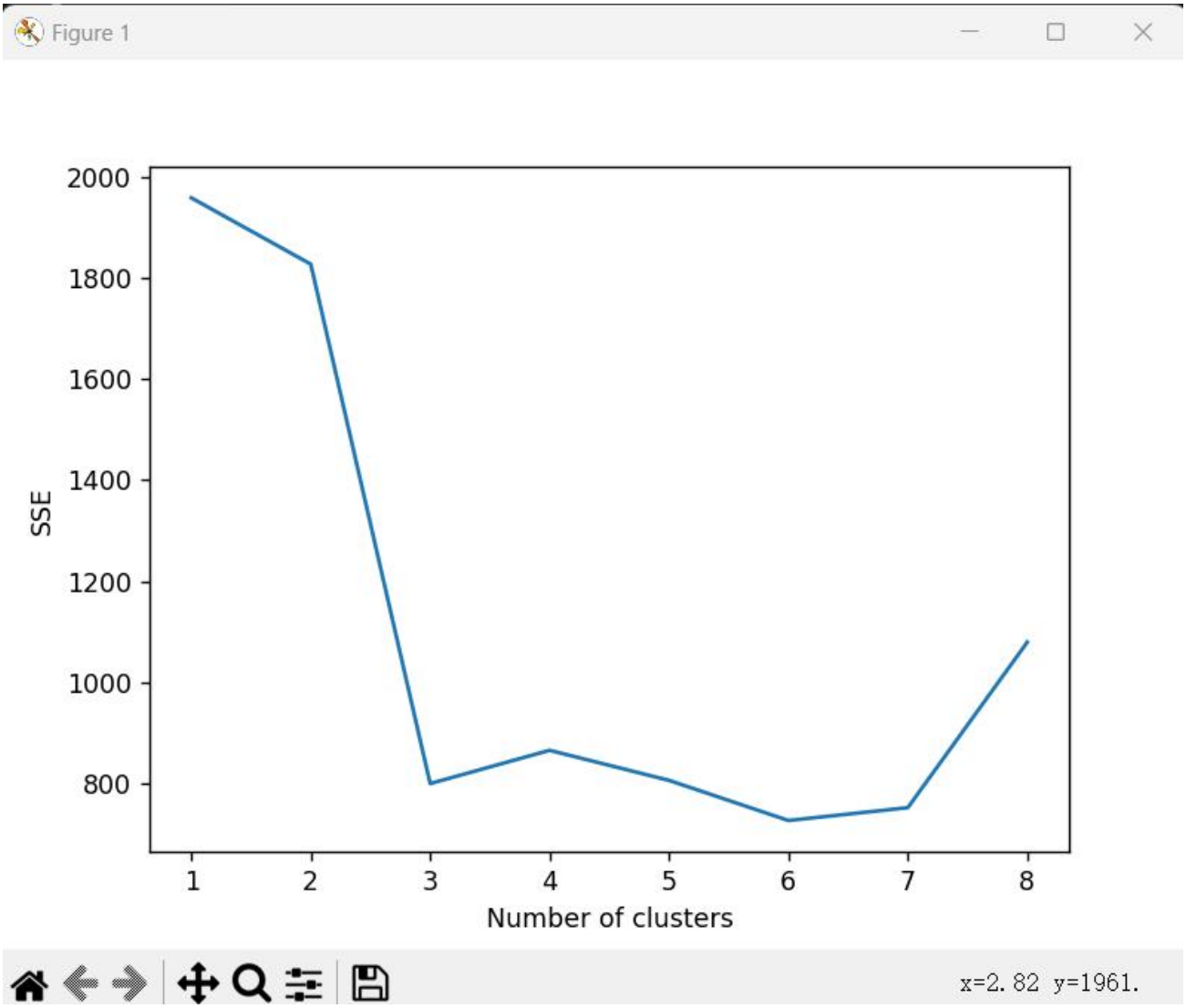
```
    centers = []
    centers.append(random.sample(data.tolist(),1)[0])   #随机选取一个点作为初始样本中心
    for cnt in range(1,k):  #选取剩下的样本中心
        dist = []
        for i in range(len(data)):
            dis = 0
            for j in range(len(center)):
                dis += distance(data[i],center[j])  #求出每个样本点与现有的样本中心的距离之和
            dist.append(dis)
        new_item = None
        while True:
            new_item = random.choices(data.tolist(),dist,k=1)[0]    #根绝dist数组设置权重，生成权重数组，选择新的中心
            if(new_item not in centers):break   #检查新生成的样本中心是否再原来已有的样本中心中，如果在则继续
        centers.append(new_item)
    return centers

def k_means(center,data,k,mistake=0.00001, max_iters=100):  #设置误差值和最大循环次数
    """k_means核心算法,求解聚类"""
    for i in range(max_iters):  #设置最大循环次数，防止过度循环而浪费时间和资源
        cluster,new_center = classify(data, center, k)
        distance = np.linalg.norm(np.array(new_center) - np.array(center))
        if distance < mistake:  #如果在误差值以内则退出，成功分类
            break
        center = new_center
    return cluster, new_center  #返回聚类和当前的聚类中心

def calculate_sse(cluster,center,k):
    """计算在不同的k值下的SSE值"""
    sum = 0.0
    for i in range(k):
        for j in range(len(cluster)):
            for k in range(len(cluster[j])):
                sum += distance(cluster[j][k],center[j])**2
    return sum

data = read_file('kmeans_data.csv')
```

```
sse = []
for k in range(1,9):    #循环测试多个k值
    init_cent = init_center(data, k)
    cluster,center = k_means(init_cent,data, k)
    sse.append(calculate_sse(cluster, center, k))

plt.plot(range(1, 9), sse)
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
plt.show()  #将SSE值随k的变化的折线图绘制出来

init_cent = init_center(data, 3)    #根据折线图可知 k = 3 处为最佳
cluster,center = k_means(init_cent,data, 3)
colors = ['r','b','g','k','c','m','y','w']  #设置颜色列表，标记成不同颜色
for i in range(len(cluster)):
    plt.scatter([x[0] for x in cluster[i]], [x[1] for x in cluster[i]],
c=colors[i%len(colors)],cmap = 'viridis')
    #为每个聚类赋予不同的颜色标记
plt.show()  #将最终的分类结果展示出来
```

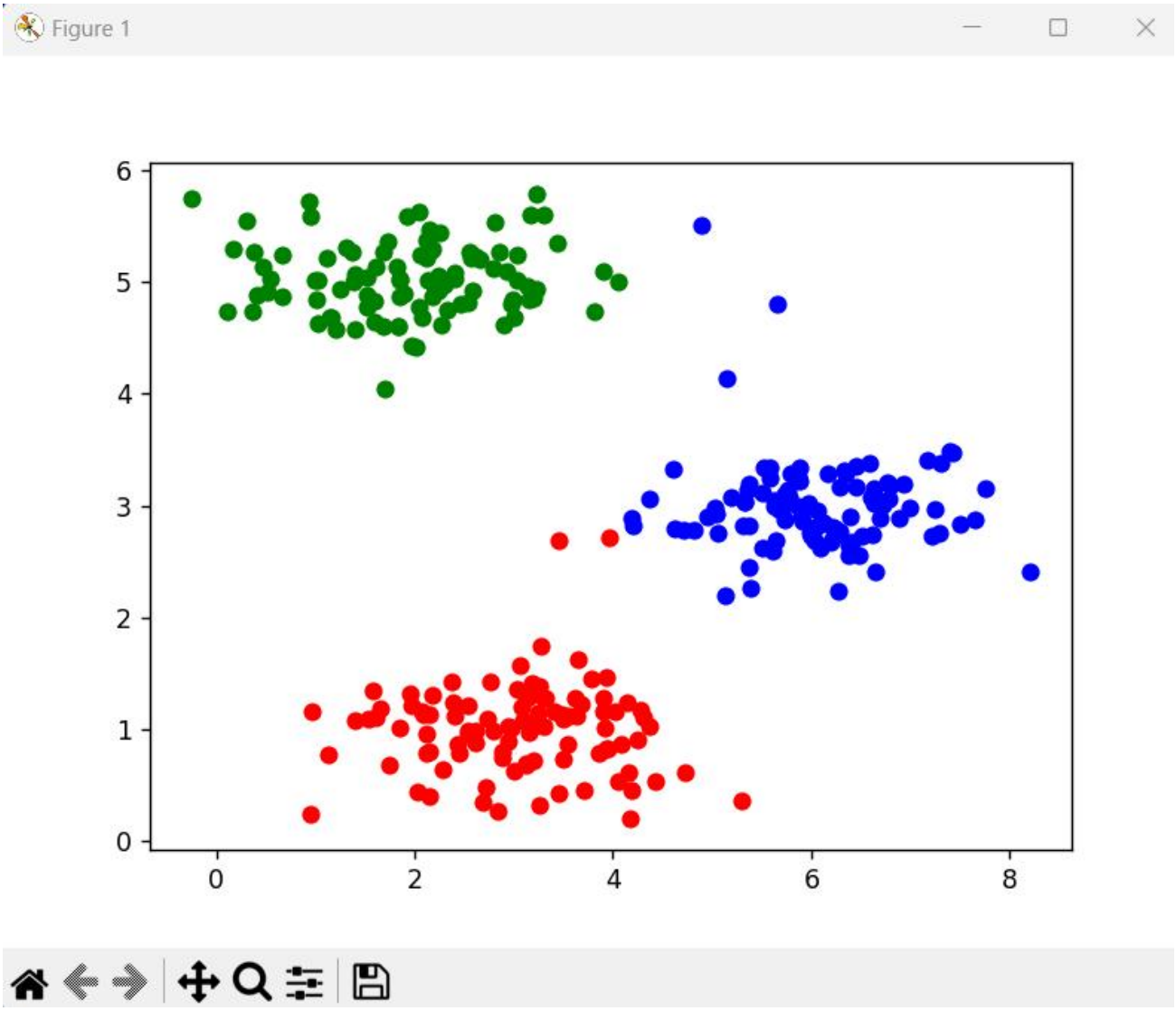
三、实验结果分析

运行截图

- **k-means手肘图**



- **k-means聚类图**



- **k-means++手肘图**

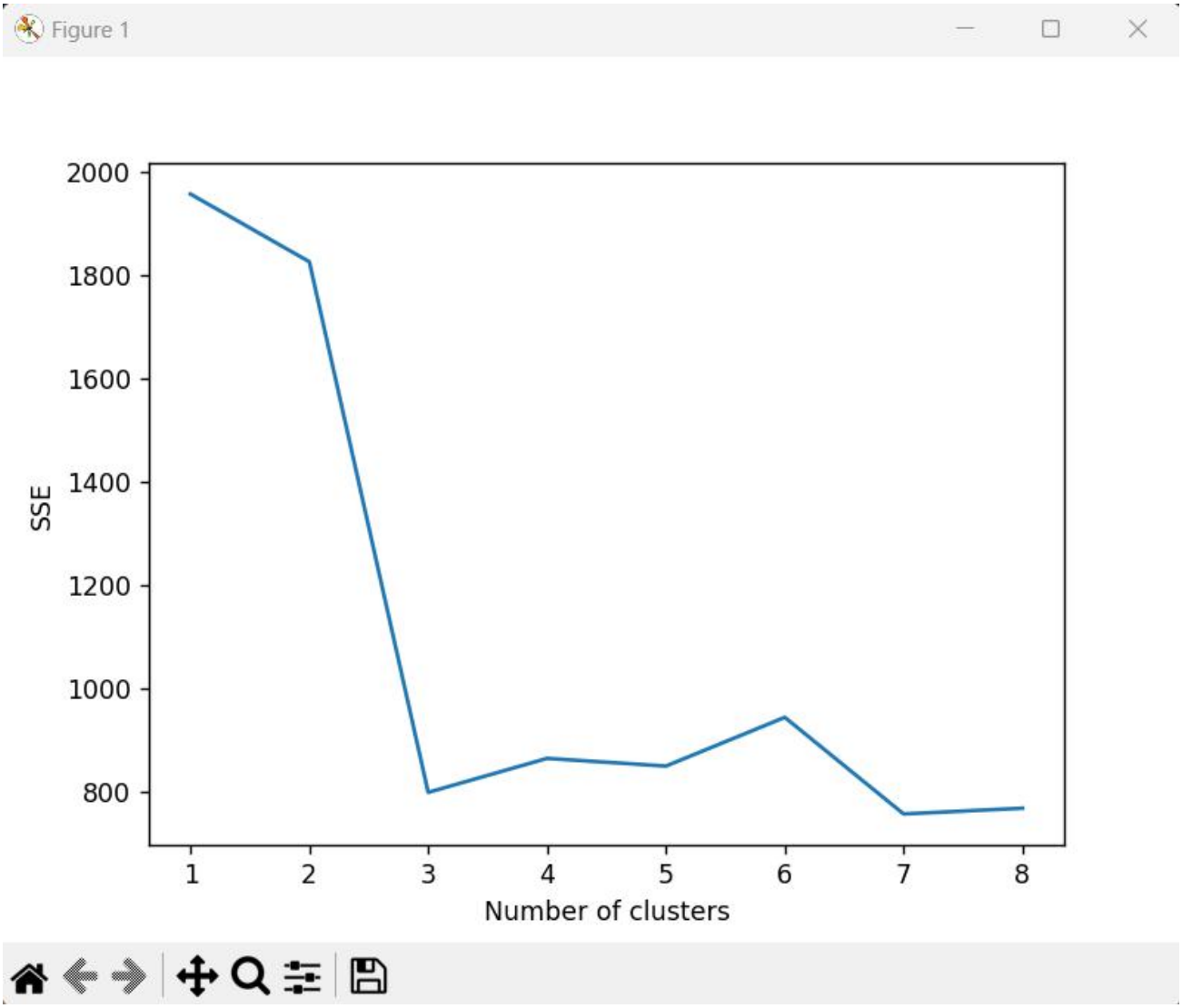
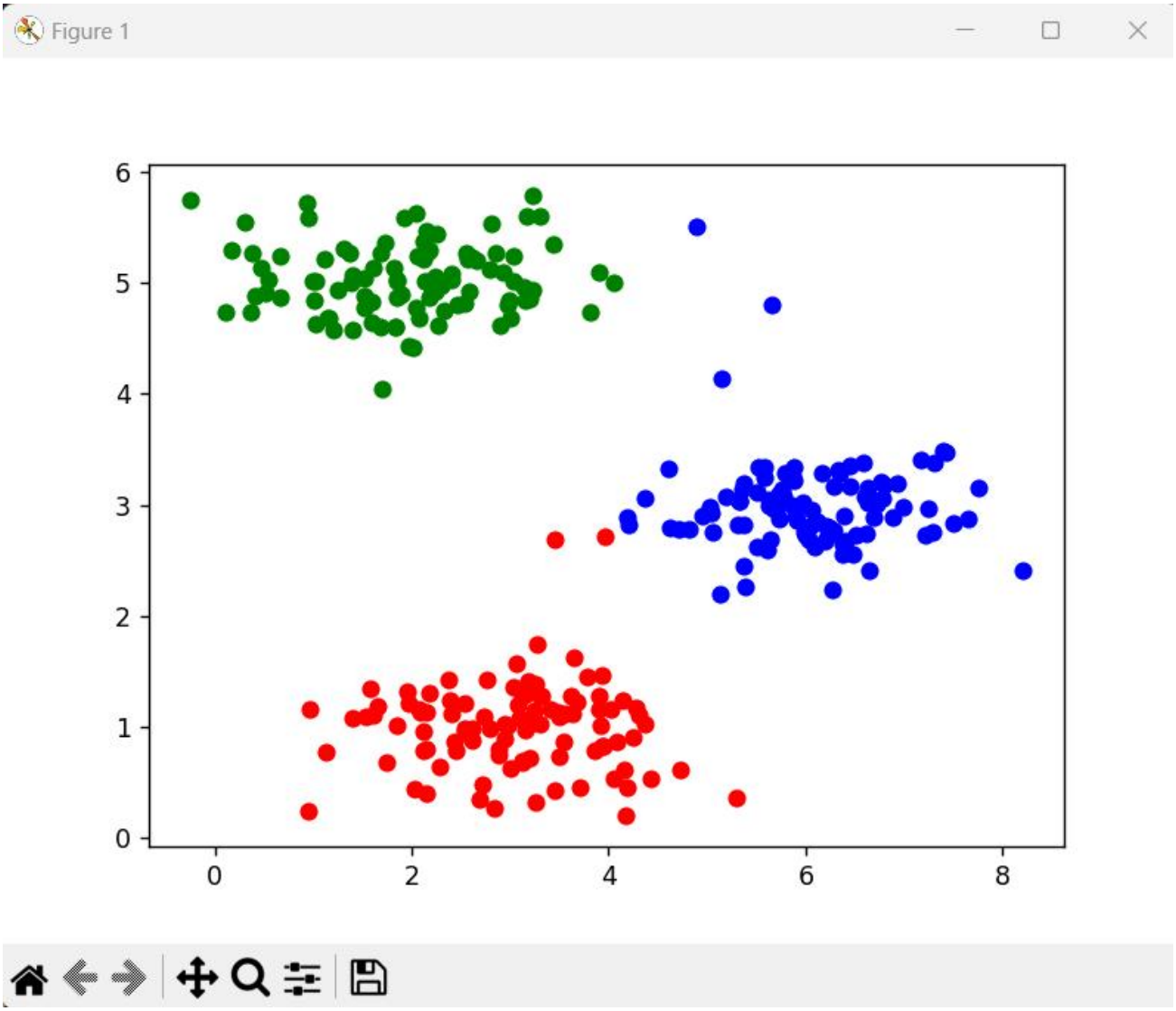


Figure 1
2000
1800
1600
1400
1200
1000
800
SSE
1
2
3
4
5
6
7
8
Number of clusters

- **k-means++聚类图**



结果分析

- 手肘图分析
  由SSE计算公式可得$\sum_i^k \sum_x |x - m_i|^2$，画出随k值变化的SSE值，可以看到，在k = 3时趋于平缓，即所谓的肘部，因此，聚类的真实分类为3

- k = 3
  结合肘部分析法，可以选取k=3时作图，实验结果如上所示，明显的区分出了三个聚类，具有良好的效果