K-Means 代码及其分析

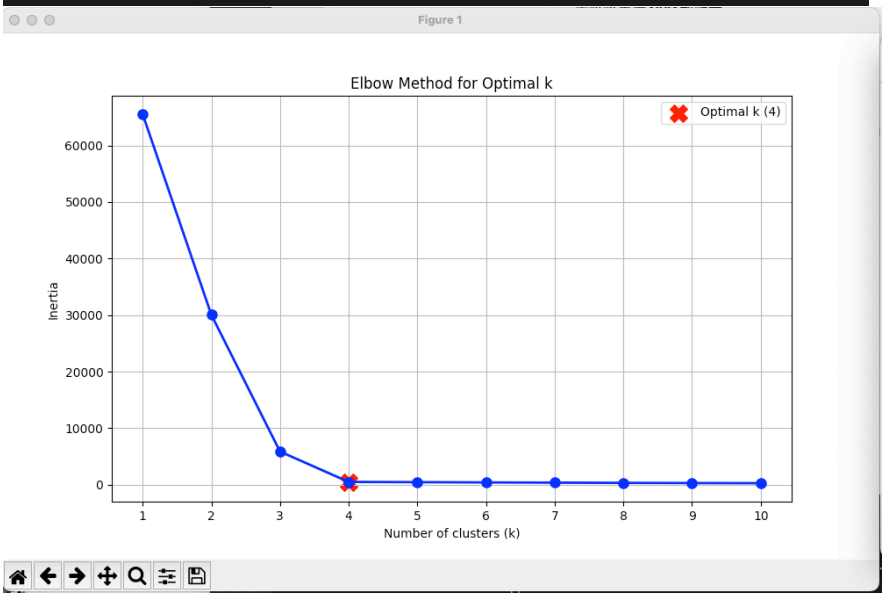
作者：DeepSeek

生成日期：2025/03/05

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

# 1. 生成示例数据
# 参数说明:
# n_samples: 样本数量
# centers: 真实聚类中心数
# cluster_std: 聚类标准差（控制分散程度）
# random_state: 随机种子（确保可重复性）
X, y = make_blobs(n_samples=1000,
                  centers=4,
                  cluster_std=0.5,
                  random_state=42)
```

```python
# 2. 计算不同k值的惯性(Inertia)
inertia = []
k_range = range(1, 11)  # 测试k=1到10

for k in k_range:
    kmeans = KMeans(n_clusters=k,
                    init='k-means++',
                    max_iter=300,
                    random_state=42)
    kmeans.fit(X)
    inertia.append(kmeans.inertia_)  # 获取当前k值的惯性值
```

```python
# 3. 绘制手肘图
plt.figure(figsize=(10, 6))
plt.plot(k_range, inertia, 'bo-', linewidth=2, markersize=8)
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
plt.xticks(k_range)
plt.grid(True)

# 4. 标记可能的最佳k值（示例取k=4）
optimal_k = 4
plt.scatter(optimal_k, inertia[optimal_k-1],
            s=200, c='red', marker='X',
            label=f'Optimal k ({optimal_k})')
plt.legend()
plt.show()
```



1 代码分析

1.1 1. 生成示例数据

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

```
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

X, y = make_blobs(n_samples=1000,
                  centers=4,
                  cluster_std=0.5,
                  random_state=42)
```

- 导入必要的库。
- 使用 `make_blobs` 函数生成一个包含 1000 个样本，4 个簇的随机数据集。
- `random_state` 确保数据集生成的可重复性。

1.2 2. 计算不同 k 值的惯性

```
inertia = []
k_range = range(1, 11)

for k in k_range:
    kmeans = KMeans(n_clusters=k,
                    init='k-means++',
                    max_iter=300,
                    random_state=42)
    kmeans.fit(X)
    inertia.append(kmeans.inertia_)
```

- 循环遍历 k 值（1 到 10），计算每个 k 值对应的惯性。
- 使用 `KMeans` 创建模型，并使用 k-means++ 初始化。
- `kmeans.inertia_` 获取惯性值。

1.3 3. 绘制手肘图

```
plt.figure(figsize=(10, 6))
plt.plot(k_range, inertia, 'bo-', linewidth=2, markersize=8)
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
plt.xticks(k_range)
plt.grid(True)
```

- 绘制 k 值与惯性值的关系图（手肘图）。
- 设置图表的标题、坐标轴标签、刻度和网格线。

1.4 4. 标记最佳 k 值

```
optimal_k = 4
plt.scatter(optimal_k, inertia[optimal_k-1],
            s=200, c='red', marker='X',
            label=f'Optimal k ({optimal_k})')
plt.legend()
plt.show()
```

- 标记手肘图中的最佳 k 值（示例中为 4）。
- 添加图例并显示图形。

2 惯性（Inertia）计算公式

在 K-means 聚类算法中，惯性（Inertia）也被称为簇内平方和（Within-Cluster Sum of Squares, WCSS）。它衡量的是每个样本点到其所属簇的质心的距离的平方和。

具体来说，惯性的计算公式如下：

$$Inertia = \sum_{i=1}^{n} ||x_i - c_{k_i}||^2$$

其中：

- n 是样本点的数量。
- x_i 是第 i 个样本点。
- c_{k_i} 是第 i 个样本点所属簇的质心。
- $||\cdot||$ 表示欧几里得距离。

简单来说，惯性就是所有样本点到其所属簇质心的距离的平方和。因此，惯性越小，表示簇内的样本点越紧密。

3 代码修订选项

3.1 数据标准化

在处理真实数据时，特征的尺度可能差异很大，这会影响 K-means 聚类的结果。因此，数据标准化是一个重要的预处理步骤。

```
from sklearn.preprocessing import StandardScaler

# 数据标准化
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 使用标准化后的数据进行K-means聚类
kmeans = KMeans(n_clusters=k, ...)
kmeans.fit(X_scaled)
```

3.2 处理真实数据

处理真实数据时，需要注意以下几点：

- **数据清洗：**处理缺失值、异常值等。
- **特征选择：**选择与聚类目标相关的特征。
- **数据标准化：**如上所述，标准化特征尺度。
- **评估指标：**除了手肘法，还可以结合其他评估指标。

3.3 拐点不明显时的应对

当手肘图的拐点不明显时，可以尝试以下方法：

- **调整 k 值范围：**扩大或缩小 k 值的搜索范围。
- **使用其他评估指标：**结合轮廓系数、Calinski-Harabasz 指数等。
- **可视化聚类结果：**将聚类结果可视化，直观判断聚类效果。

3.4 结合轮廓系数（Silhouette Score）

轮廓系数用于评估聚类效果，其取值范围为 [-1, 1]。

- **1：**样本与其所属簇的样本相似度很高，与其他簇的样本相似度很低。
- **0：**样本处于两个簇的边界上。
- **-1：**样本被错误地分配到簇中。

轮廓系数的计算公式如下：

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

其中：

- $a(i)$：样本 i 与其所属簇中其他样本的平均距离。
- $b(i)$：样本 i 与其他簇中样本的最小平均距离。

```python
from sklearn.metrics import silhouette_score

silhouette_avg = silhouette_score(X, kmeans.labels_)
```

4 编后语

请用辩证的观点，评价上述 LLM 材料对 K-Means 学习的帮助。