北京大学

机器学习基础
课程上机试验报告

试验内容： 聚类方法

姓　　名：王宇哲
学　　号：1800011828

1 题目 1

1.1 题目内容

比较 K-means 聚类方法和学习向量量化方法的主要思想的异同。

1.2 题目解答

1.2.1 相同点

1) 学习向量量化 (Learning Vector Quantization, LVQ) 方法与 K-means 聚类方法类似，均试图找到一组原型向量来刻画聚类结构。

2) LVQ 与 K-means 均通过对原型向量不断迭代优化，从而对原型向量以及样本空间的簇划分进行更新直至收敛，从而得到最终的聚类结果。

1.2.2 不同点

1) LVQ 假设数据样本带有类别标记，学习过程利用样本的这些监督信息来辅助聚类，而 K-means 的数据样本不带有标记信息，是无监督的。

2) LVQ 更新原型向量的方法与 K-means 不同。K-means 在每个簇 C_i 中根据

$$\mu_i' = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

对原型向量进行更新，更新后的原型向量为当前簇中样本的均值。而 LVQ 考虑任一样本 x_j，若最近的原型向量 p_{i^*} 与 x_j 的类别标记相同，则令 p_{i^*} 向 x_j 的方向靠拢，即

$$p' = p_{i^*} + \eta \cdot (x_j - p_{i^*})$$

其中 $\eta \in (0, 1)$ 为学习率。若 p_{i^*} 与 x_j 的类别标记不同，则

$$p' = p_{i^*} - \eta \cdot (x_j - p_{i^*})$$

2 题目 2

2.1 实验题目

对于需要预先设定簇数 k 的聚类算法，讨论 k 的确定策略，并自选一个聚类算法进行实验。

2.2 实验数据

题目 2 使用的数据集为 UCI 上的 wine 数据集 (https://archive.ics.uci.edu/ml/datasets/Wine)，在 scikit-learn 库中也集成了该数据集。Wine 数据集包含了种植在意大利相同地区的、被分为 class_0、class_1、class_2 三类的 178 个红酒样本及其 13 维化学分析数据，适用于一般的聚类问题。

2.3 实验工具

实验在个人 LEVENO YOGA 710 笔记本电脑上进行，主要硬件条件为：处理器 Intel i5-7200U CPU，内存大小 8.0 GB，显卡 NVIDIA Geforce 940MX；操作系统为 Windows 10 64 位系统。

实验使用语言为 python，具体版本为 python 3.7.0，通过 Anaconda 安装了 Matplotlib、NumPy、pandas、scikit-learn 等常用库。出于本次实验需要，通过 Anaconda 安装了 Gap Statistic 库 (https://github.com/milesgranger/gap_statistic)。实验代码编写和运行均在 Jupyter Notebook 上进行，具体版本为 Jupyter Notebook 6.3.0。

2.4 实验方法

对于需要预先设定簇数 k 的聚类算法，下面讨论一般的簇数 k 的确定策略，并选择 $K-$means++ 算法，在 wine 数据集上进行实验。$K-$means++ 算法在 $K-$means 算法的基础上，选取的初始聚类中心相互距离尽可能远，从而一定程度上减小了最终聚类结果的误差。

2.4.1 根据实际需要确定 k 值

在设定聚类簇数 k 时，若 k 在问题中有明确的意义，则根据问题的实际需要确定 k 值。例如在对 wine 数据集进行聚类时，若预先知道红酒样本被分为 3 类，则可以设定 $k = 3$ 进行聚类。该方法有很大程度的局限性，考虑到多数情况下 k 值无法事先加以确定。

2.4.2 手肘法 (Elbow Method)

对于聚类过程，定义簇内平方和 (Inertia, or within-cluster sum-of-squares)

$$W_k = \sum_{k=1}^{K} \sum_{x_i \in C_k} ||x_i - \mu_k||^2$$

W_k 即为聚类算法通过迭代优化进行最小化的目标函数。随着 k 值从 1 逐渐增大，当 k 值小于最优 k 值时，W_k 随 k 值增大而大幅减小；而当 k 值大于最优 k 值时，随 k 值增大，W_k 的变化较为平缓，W_k 随 k 的变化曲线形成手肘 (elbow) 形状。因此，可以选取拐点处 k 值作为最优 k 值。

实验使用 sklearn.KMeans 库对 wine 数据集进行聚类，计算不同 k 值下的 W_k，使用 Elbow Method 确定最优簇数 k，并作出 $W_k \sim k$ 曲线，代码实现如下。

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

def calculate_Inertia(data, n_clusters):
kmeans = KMeans(init="k-means++", n_clusters=n_clusters, n_init=4,
    random_state=0)
estimator = make_pipeline(StandardScaler(), kmeans).fit(data)
result = estimator[-1].inertia_
return result

n_list, Inertia = [], []
for n_clusters in range(1, 11):
n_list.append(n_clusters)
Inertia.append(calculate_Inertia(data, n_clusters))

n_list = np.array(n_list)
Inertia = np.array(Inertia)

%config InlineBackend.figure_format = 'svg'
f, ax = plt.subplots(1)
plt.xlabel(r'Cluster Count')
plt.ylabel(r'Inertia')
plt.rcParams['xtick.direction'] = 'in'
plt.rcParams['ytick.direction'] = 'in'
draw_plot_and_scatter(n_list, Inertia)
f.set_size_inches(5,4)
plt.savefig('elbow_method.jpg',dpi=1000, bbox_inches='tight')
plt.show(f)
```

手肘法操作简便，但本质上仍需要通过人工观察判断拐点位置，最优 k 值的选取有一定的随意性，难以实现自动化批量处理。

2.4.3 轮廓系数 (Silhouette Coefficient) 方法

轮廓系数 (Silhouette Coefficient) 由 Rousseeuw et al. 于 1987 年提出[1]。对于聚类过程中的任一样本，定义轮廓系数 (Silhouette Coefficient)

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

其中 a 为该样本与同簇内所有其他样本点的平均距离，b 为该样本与最近邻簇内所有样本点的平均距离。轮廓系数 s 能够一定程度上衡量聚类的合理性与有效性，所有样本 s 的平均值越高，则聚类结果越合理。因此，可以作出 $s \sim k$ 曲线，选取使得 s 取最大值的簇数 k 作为最优簇数 k。

实验对 wine 数据集进行聚类，计算不同 k 值下的 Silhouette Coefficient，确定最优簇数 k，并作出 $s \sim k$ 曲线，代码实现如下。

```
from sklearn import metrics

def calculate_silhouette(data, n_clusters):
kmeans = KMeans(init="k-means++", n_clusters=n_clusters, n_init=4,
    random_state=0)
estimator = make_pipeline(StandardScaler(), kmeans).fit(data)
result = metrics.silhouette_score(data, estimator[-1].labels_,
    metric="euclidean", sample_size=300)
return result

n_list, Silhouette = [], []
for n_clusters in range(2, 11):
n_list.append(n_clusters)
Silhouette.append(calculate_silhouette(data, n_clusters))

n_list = np.array(n_list)
Silhouette = np.array(Silhouette)

%config InlineBackend.figure_format = 'svg'
```

```
  f, ax = plt.subplots(1)
  plt.xlabel(r'Cluster Count')
  plt.ylabel(r'Silhouette Coefficient')
  plt.rcParams['xtick.direction'] = 'in'
  plt.rcParams['ytick.direction'] = 'in'
  draw_plot_and_scatter(n_list, Silhouette)
  f.set_size_inches(5,4)
  plt.savefig('Silhouette.jpg',dpi=1000, bbox_inches='tight')
  plt.show(f)
```

轮廓系数方法的缺陷是其计算复杂度为 $O(n^2)$，需要计算样本的距离矩阵，样本量较大时计算开销很大。

2.4.4 间隔统计量 (Gap Statistic) 方法

间隔统计量 (Gap Statistic) 方法由 Tibshirni et al. 于 2001 年提出[2]。对于聚类过程，定义

$$\mathrm{Gap}(k) = E(\log W_k) - \log W_k$$

其中 $E(\log W_k)$ 为 $\log W_k$ 的期望，通常通过 Monte Carlo 模拟产生，在样本所在的区域按照均匀分布随机产生和原始样本数一样多的随机样本，并对随机样本进行聚类得到 W_k，重复多次求平均值得到 $E(\log W_k)$ 的近似值。作出 $\mathrm{Gap}(k) \sim k$ 曲线，选取使得 $\mathrm{Gap}(k)$ 取最大值的簇数 k 即为最优簇数 k。

实验对 wine 数据集进行聚类，使用 Gap Statistic 库计算不同 k 值下的 $\mathrm{Gap}(k)$，确定最优簇数 k，并作出 $\mathrm{Gap}(k) \sim k$ 曲线，代码实现如下，由于结果波动较大，对 100 次计算结果取平均值得到最终结果。

```
  from gap_statistic import OptimalK

  optimalK = OptimalK(n_jobs=4, parallel_backend='joblib')

  def gap_stat(data, value='gap_value'):
  n_clusters = optimalK(data, cluster_array=np.arange(1, 11))
  optimalK.gap_df.head()
  return optimalK.gap_df.n_clusters, optimalK.gap_df.eval(value)

  n_array, gap_array = gap_stat(data)
```

```
  for k in range(99):
  gap_array = [i + j for i, j in zip(gap_array, gap_stat(data)[1])]
  gap_array = [0.01*i for i in gap_array]


  %config InlineBackend.figure_format = 'svg'
  f, ax = plt.subplots(1)
  plt.xlabel(r'Cluster Count')
  plt.ylabel(r'Gap Value')
  plt.rcParams['xtick.direction'] = 'in'
  plt.rcParams['ytick.direction'] = 'in'
  draw_plot_and_scatter(n_array, gap_array)
  f.set_size_inches(5,4)
  plt.savefig('gap_statistic.jpg',dpi=1000, bbox_inches='tight')
  plt.show(f)
```

在某些情况下，Gap$(k) \sim k$ 曲线可能单调下降或单调上升，从而导致无法获得最优簇数 k。此时，记 Monte Carlo 模拟次数为 B，B 次 Monte Carlo 模拟计算得到 $\log W_k$ 的标准差为 sd(k)，定义

$$s_k = \sqrt{1 + \frac{1}{B}}\ \mathrm{sd}(k)$$

此时只需选取最小的 k，满足

$$\mathrm{Gap}(k) \geq \mathrm{Gap}(k+1) - s_{k+1}$$

定义

$$\mathrm{Diff}(k) = \mathrm{Gap}(k) \geq \mathrm{Gap}(k+1) - s_{k+1}$$

作出 Diff$(k) \sim k$ 曲线，选取使得 Diff(k) ≥ 0 的最小的 k 即为最优簇数 k。

实验对 wine 数据集进行聚类，使用 Gap Statistic 库计算不同 k 值下的 Diff(k)，确定最优簇数 k，并作出 Diff$(k) \sim k$ 曲线，代码实现如下，由于结果波动较大，对 100 次计算结果取平均值得到最终结果。

```
  n_array, diff_array = gap_stat(data, value='diff')

  for k in range(99):
```

```
    diff_array = [i + j for i, j in zip(diff_array, gap_stat(data,
        value='diff')[1])]
    diff_array = [0.01*i for i in diff_array]

    %config InlineBackend.figure_format = 'svg'
    f, ax = plt.subplots(1)
    plt.xlabel(r'Cluster Count')
    plt.ylabel(r'Diff Value')
    plt.rcParams['xtick.direction'] = 'in'
    plt.rcParams['ytick.direction'] = 'in'
    draw_plot_and_scatter(n_array, diff_array)
    f.set_size_inches(5,4)
    plt.savefig('diff_statistic.jpg',dpi=1000, bbox_inches='tight')
    plt.show(f)
```

2.5 实验结果

2.5.1 手肘法 (Elbow Method)

实验代码运行时长为 1.25 s。作出 $W_k \sim k$ 曲线，如图 **1** 所示。

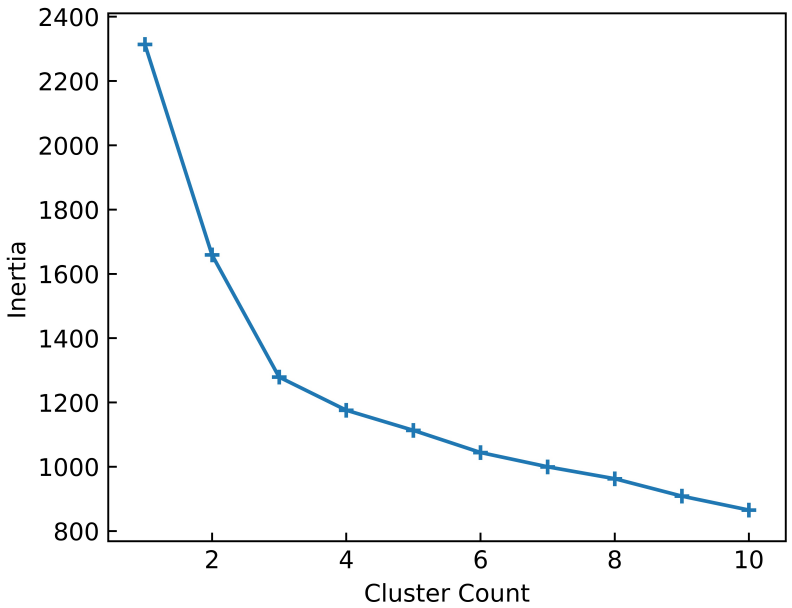

图 **1** $W_k \sim k$ 曲线

Fig. 1 $W_k \sim k$ **Curve**

根据图 **1**，可见 $k = 3$ 为曲线从迅速下降到平缓下降的拐点，即“手肘”的位置，故选取 $k = 3$ 作为最优簇数 k。这一结果与 wine 数据集的样本类数是一致的。

2.5.2 轮廓系数 (Silhouette Coefficient) 方法

实验代码运行时长为 1.26 s。作出 $s \sim k$ 曲线，如图 **2** 所示。

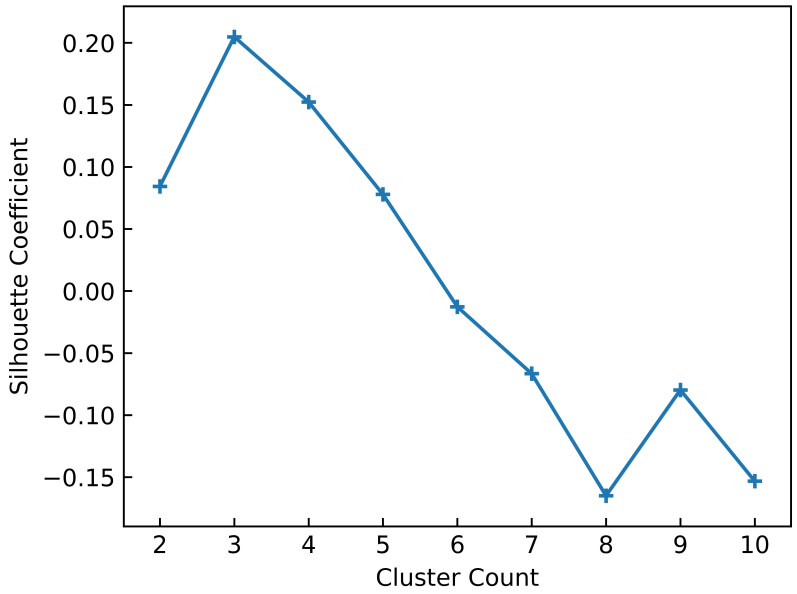

图 **2** $s \sim k$ 曲线

Fig. 2 $s \sim k$ **Curve**

根据图 **2**，可见 $k = 3$ 时 s 取最大值，故选取 $k = 3$ 作为最优簇数 k。这一结果与 wine 数据集的样本类数是一致的。

2.5.3 间隔统计量 (Gap Statistic) 方法

首先计算不同 k 值下的 $\mathrm{Gap}(k)$，实验代码运行时长为 14.2 s。作出 $\mathrm{Gap}(k) \sim k$ 曲线，如图 **3** 所示。

根据图 **3**，可见 $k = 1$ 时 $\mathrm{Gap}(k)$ 取最大值，根据间隔统计量方法的原理，应当选取 $k = 1$ 作为最优簇数 k。这一结果与 wine 数据集的样本类数不一致。

进一步地，计算不同 k 值下的 $\mathrm{Diff}(k)$，实验代码运行时长为 8.78 s。作出 $\mathrm{Diff}(k) \sim k$ 曲线，如图 **4** 所示。

根据图 **4**，可见恒有 $\mathrm{Diff}(k) \geq 0$，根据间隔统计量方法的原理，应当选取 $k = 1$ 作为最优簇数 k。这一结果与 wine 数据集的样本类数不一致。

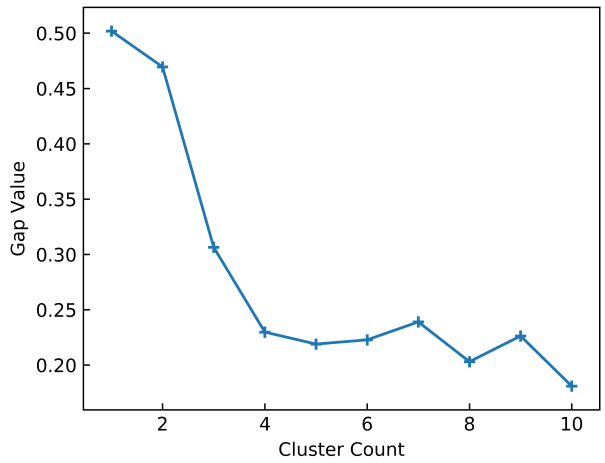

图 **3** Gap(k) $\sim k$ 曲线
Fig. 3 Gap(k) $\sim k$ **Curve**

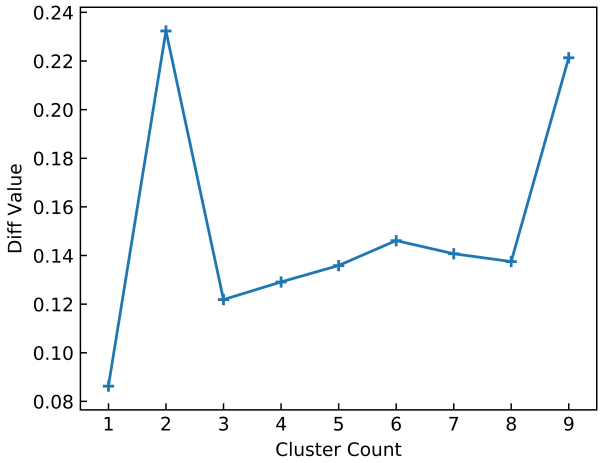

图 **4** Diff(k) $\sim k$ 曲线
Fig. 4 Diff(k) $\sim k$ **Curve**

2.6 结果分析

根据 2.5 的实验结果，手肘法 (Elbow Method) 和轮廓系数 (Silhouette Coefficient) 方法确定最优簇数 $k = 3$，与 wine 数据集的样本类数一致。而间隔统计量 (Gap Statistic) 方法确定最优簇数 $k = 1$，与 wine 数据集的样本类数不一致，其原因可能是 wine 数据集的样本数量较少，间隔统计量方法计算结果波动较大，从而导致对最优簇数 k 的判断出现偏差。

对于需要预先设定簇数 k 的聚类算法，若无法事先根据实际需要确定最优簇数 k，则在确定最优簇数 k 时可以综合考虑手肘法 (Elbow Method)、轮廓系数 (Silhouette Coefficient) 方法、间隔统计量 (Gap Statistic) 等方法所给出的结果，结合实际需要作出综合判断。

参考文献

[1] P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[2] R. Tibshirani, G. Walther, and T. Hastie, “Estimating the number of clusters in a data set via the gap statistic,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.