# 本科实验报告

| | |
|---|---|
| 课程名称： | 人工智能 |
| 实验名称： | $k$-means 聚类 |
| 专业名称： | 保密管理 |
| 学生姓名： | 武自厚 |
| 学生学号： | 20336014 |
| 实验地点： | 东校园实验楼 D502 |
| 实验成绩： | |
| 报告时间： | 2022 年 5 月 23 日 |

## 一、 实验要求

利用 $k$-means，$k$-means++ 对文本特征进行聚类，并计算聚类准确率.

## 二、 实验过程

### 1. $k$-means 原理

$k$-means 是一种简单的无监督学习聚类算法. 其过程可以简要概括为:

(1) 输入样本, 并随机选择 $k$ 个样本作为中心.

(2) 根据单个样本距离 $k$ 个中心的距离决定这个样本所属聚类.

(3) 所有样本确定聚类后, 根据各个聚类中所有样本位置的平均值确定新的中心位置.

(4) 预先设置临界值 $\varepsilon$, 如果所有样本更新过程中移动的距离小于 $\varepsilon$, 则算法结束, 否则回到第 (2) 步继续.

其中样本的分布以及特征的选取会影响该算法的效率和准确度, 最开始中心选择的随机性也导致了算法的结果不稳定.

### 2. $k$-means++

$k$-means++ 是对 $k$-means 算法最开始中心选择随机性过高而做出的一个改进. 改进的内容为:

(1) 随机选择一个样本作为第一个中心.

(2) $\forall x \in$ 样本集$X$, 根据权重 $D^2(x)$ 随机选择一个新的样本加入中心集, 其中 $D^2(x)$ 是样本 $x$ 离所有中心中最近的距离.

这样, 距离现有中心更远的点更有可能成为新的中心, 改进了中心选择可能太近导致聚类效果不佳的缺点.

### 3. 关键代码

代码中采用 `sklearn` 库中的 `CountVectorizer` 来提取语料库. 具体预测代码如下:

```python
def dist(x: np.ndarray, y: np.ndarray) -> np.ndarray:
    return np.sqrt(np.sum(np.power(x - y, 2), axis=1))

def calc_min_d(centroids: np.ndarray, x: np.ndarray):
    dists = dist(centroids, x)
    return np.power(np.min(dists), 2)

def match_best_centroids(centroids: np.ndarray, x: np.ndarray):
    dists = dist(centroids, x)
    return np.argmin(dists)
```

```python
def renew_centroids(x: np.ndarray, clusters: np.ndarray, k: int):
    new_centroids = []
    for i in range(k):
        indices = np.argwhere(clusters == i).flatten()
        x_cluster = x[indices, :]
        new_centroid = np.sum(x_cluster, axis=0) / np.size(x_cluster, axis=0)
        new_centroids.append(new_centroid)
    return np.array(new_centroids)

def init_standard(x: np.ndarray, k: int):
    x_col, x_row = x.shape
    return np.random.choice(np.arange(x_col), size=k, replace=False)

def init_plusplus(x: np.ndarray, k: int):
    x_row, x_col = x.shape
    first = np.random.choice(np.arange(x_row))
    selected = np.array([first])

    for i in range(k-1):
        remained_indices = np.where(np.arange(x_row) != selected.any())[0]
        d_x = np.apply_along_axis(calc_min_d, 1, x[remained_indices, :], x[selected,
        ↪  :])
        new_centroid = np.random.choice(remained_indices, p=d_x/np.sum(d_x))
        selected = np.append(selected, new_centroid)

    return selected

def k_means(x: np.ndarray, k: int, plusplus: bool):
    if plusplus:
        centroids_indices = init_plusplus(x, k)
    else:
        centroids_indices = init_standard(x, k)

    centroids = x[centroids_indices, :]
    init_centroids = centroids
    while True:
        clusters = np.apply_along_axis(match_best_centroids, 1, x, centroids)
        new_centroids = renew_centroids(x, clusters, k)
        if np.all(dist(centroids, new_centroids)) < 0.00001:
            return init_centroids, centroids, clusters
        else:
            centroids = new_centroids
```

## 三、 实验结果

如前几次实验的一样, 这里将文本集聚类为 6 类展示. 聚类评估采用轮廓系数评价, 轮廓系数的分布区间为 $[-1, 1]$, 数值越大聚类效果越好, 轮廓系数的评估调用 `sklearn.metrics.silhouette_score`. 图示结果中不同颜色的点代表不同聚类, 黑色的 "×" 表示聚类中心. 数据展示采用 TSNE 降维.

$k$-means 算法得到的聚类结果如图所示:

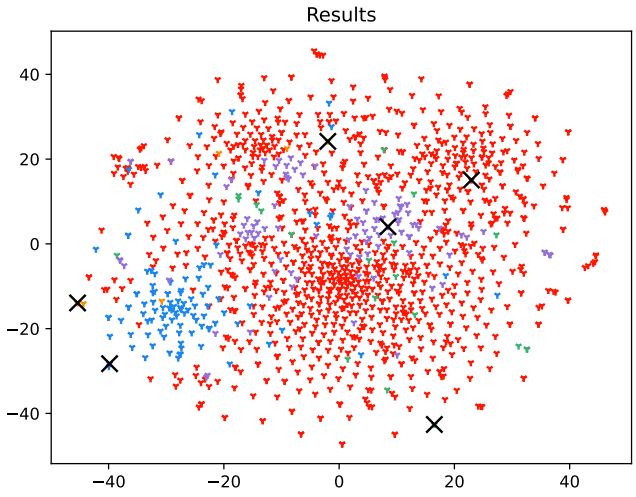


图 1: $k$-means 算法结果 (轮廓系数为 0.007981)

$k$-means++ 算法得到的聚类结果如图所示:

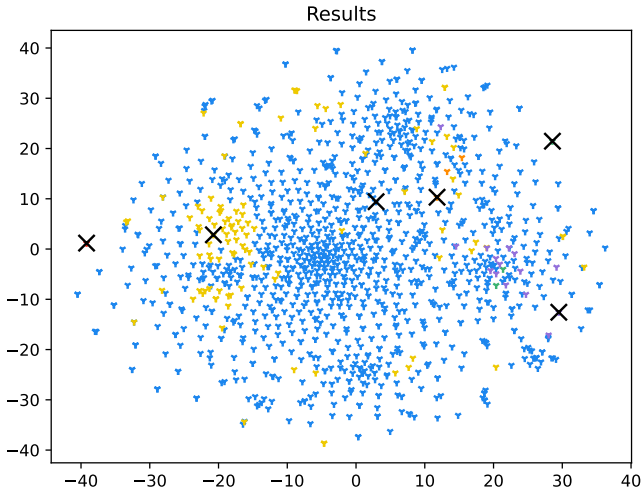


图 2: $k$-means++ 算法结果 (轮廓系数为 0.033370)

可以发现 $k$-means++ 算法做出的改进可以对于聚类效果进行显而易见的优化.