西安交通大學
XI'AN JIAOTONG UNIVERSITY

机器学习

实验报告

2023-2024 学年　第二学期

学　　院：　数学与统计学院

班　　级：　数学强基 2102

学　　号：　2213210010

姓　　名：　周冠程

指导教师：　孟德宇老师

实验地点：　数学实验中心

1 实验内容

1.1 问题描述

假设你有一个二维数据集，其中包含 200 个样本，每个样本有 2 个特征。你想使用 K-means 算法将这些样本分成四个簇。

1.2 实验步骤

1. 随机生成 200 个二维样本作为数据集。
2. 将数据集分成四个簇。在算法中，设置初始簇中心为随机选择的四个样本。
    - 初始化聚类中心：随机选择 K 个数据点作为初始的聚类中心。
    - 分配数据点到最近的聚类中心：对于每个数据点，计算它与各个聚类中心的距离，并将其分配到距离最近的聚类中心所在的簇。
    - 更新聚类中心：对于每个簇，计算该簇内所有数据点的平均值，将其作为新的聚类中心。
    - 重复步骤 2 和 3：重复执行步骤 2 和 3，直到聚类中心不再发生变化或者达到预定的迭代次数。
3. 可视化聚类结果。使用不同颜色表示不同的簇，并将簇中心用大圆点标记出来。

1.3 实验要求

使用 Matlab 编程语言来完成实验；尽量不调用 Matlab 中的 kmeans 函数实现 K-means 算法；可以使用 Matlab 中的 rand 函数来随机生成样本。最终需要将聚类结果可视化展示，并标记出簇中心点。

1.4 提示

- 可以使用 Python 编程语言和相应的库来完成实验；
- 可以使用鸢尾花数据集。

2 实验描述

2.1 算法介绍

K-means 算法是一种常见的聚类算法，用于将数据集中的样本划分为多个不同的类别，使得同一类别内的样本之间的相似度较高，而不同类别之间的相似度较低。该算法由 James MacQueen 于 1967 年提出，是一种基于迭代的无监督学习算法。

K-means 算法的基本思想和步骤如下：

1. **初始化：**随机选择 K 个样本作为初始的聚类中心。
2. **聚类分配：**将数据集中的每个样本分配给距离其最近的聚类中心所在的类别，形成 K 个簇。
3. **更新聚类中心：**对于每个簇，计算该簇内所有样本的平均值，将该平均值作为新的聚类中心。
4. **迭代：**重复步骤 2 和步骤 3，直到满足终止条件，通常是聚类中心不再发生变化或达到最大迭代次数。

K-means 算法的优点包括：

- 简单易实现，计算效率高。
- 在处理大规模数据集时具有较好的可扩展性。
- 适用于发现球状簇的数据集。

K-means 算法的缺点包括：

- 需要提前指定簇的数量 K，且对初始聚类中心的选择敏感。
- 对异常值敏感，可能导致聚类中心偏移。
- 对非球形簇的数据效果较差。

总的来说，K-means 算法是一种简单而有效的聚类算法，在各种领域中都有广泛的应用，如市场分割、图像分割、文本聚类等。

2.2 理论推导

2.2.1 模型建模

通常意义下，一个机器学习算法模型需要包含：**训练数据集**、**决策函数集**、**表现度量**、**优化算法**，本部分将根据 K-means 算法的步骤构建机器学习算法模型，下一部分将说明这原始 K-means 的流程和该机器学习框架下的优化过程是一致的，进而在机器学习框架下理解 K-means 算法。

这是一个无监督的聚类问题，原始训练集记为 $D = \{x_i : i \in \Lambda, x_i \in \mathbb{R}^C\}$, 其中 Λ 是指标集, C 为数据特征维度。聚类结果记为 $C_i\Lambda$, 且 $\{C_i\}_{i=1}^k$ 构成 Λ 的一个分割，其中 k 为聚类数目，并定义类特征，定义 C_i 类内的数据特征为 $\mu_i \in \mathbb{R}^C$。并且在数据空间的度量通过范数 $\| \bullet \|$: $\mathbb{R}^C \to \mathbb{R}*$ 来定义，并通过该度量来评价数据差异性。通常情况下，该范数可以选取 Lp 度量族中的成员，也可以直接选取相关系数、余弦距离来直接定义距离。基于以上设定，可以容易写出模型的决策函数集

$$f(x;\mu_i) = \arg\min_j \| x - \mu_j \|^2 \tag{1}$$

一个好的聚类需要满足两个特征：**类内差异小**和**类间差异大**。根据 K-means 算法流程可以发现其仅关注类内差异，基于此，我们刻画类内差异的表现度量。C_i 类内差异刻画为

$$\sum_j \| x_j - \mu_i \|^2 \tag{2}$$

若模型参数 (C_i 和 μ_i) 已定，则整个模型的总类内差异可以表示为

$$\sum_{i=1}^k \sum_{j \in C_i} \| x_j - \mu_i \|^2 \tag{3}$$

该式用于衡量整个模型在观测数据上的总类内差异，于是表现度量可以表示为

$$\arg\min_{C_i,\mu_i} \sum_{i=1}^k \sum_{j \in C_i} \| x_j - \mu_i \|^2 \tag{4}$$

由于 C_i 的选取是一个组合问题，则该优化问题的求解是一个 NP 难的问题，难以构建多项式时间复杂度算法来得到全局最优解，可以采用迭代算法来求取近似极优解。基于此，引入**均值迭代算法**。具体而言对于优化问题 $\min_{\theta_1,\theta_2} g(\theta_1, \theta_2)$，采用以下步骤进行迭代求解：

1. 初始化 θ_1^0, θ_2^0，t=1，精度 ϵ;

2. $\theta_1^t = \arg\min_{\theta_1} f(\theta_1, \theta_2^{t-1})$;

3. $\theta_2^t = \arg\min_{\theta_2} f(\theta_1^t, \theta_2)$;

4. 若迭代次数达到上限，或者 $\| \theta_1^t - \theta_1^{t-1} \| < \epsilon$ 且 $\| \theta_2^t - \theta_2^{t-1} \|^2 < \epsilon$，则结束迭代，结果为 θ_1^t, θ_2^t。否则 $t = t + 1$，返回步骤 2。

值得注意的是，迭代解法的结果依赖于 θ_1^0, θ_2^0 的选取，也就是说不同的 θ_1^0, θ_2^0 选取会导致不同的结果，同时该算法并不保证解的最优性，所得解可能是某个局部极优解，但并非全局最优解。

综上所述，该机器学习框架为

组成部分	**定义**
训练数据集	$D = \{x_i : i \in \Lambda, x_i \in \mathbb{R}^C\}$
决策函数集	$\left\{ f(x; \mu_i) = \arg\min_j \| x - \mu_j \|^2 : \mu_i \in \mathbb{R}^C \right\}$
表现度量	$\min_{C_i, \mu_i} \sum_{i=1}^{k} \sum_{j \in C_i} \| x_j - \mu_i \|^2$
优化算法	均值迭代算法

表 1: 机器学习框架

2.2.2 相容性与收敛性

本部分将证明上一部分所定义机器学习框架的算法与传统的 K-means 是等价的，并在机器学习的框架下证明 K-means 算法是弱收敛的。

首先在机器学习框架具体化优化算法的步骤，我们选择 $\theta_2 := \{\mu_i\}$，此时均值迭代算法的具体步骤为

1. 初始化: 随机选取 μ_i^0，t=1，精度 ϵ;

2. 计算

$$C_i^t = \arg\min_{C_i} \sum_{i=1}^{k} \sum_{j \in C_i} \| x_j - \mu_i^{t-1} \|^2 \tag{5}$$

3. 计算

$$\mu_i^t = \arg\min_{\mu_i} \sum_{i=1}^{k} \sum_{j \in C_i^t} \| x_j - \mu_i \|^2 \tag{6}$$

4. 若迭代次数达到上限，或者 $\| \mu_i^t - \mu_i^{t-1} \| < \epsilon$，则结束迭代，结果为 μ_i^t。否则 $t = t + 1$，返回步骤 2。

该过程将原始需要优化的参数分为了两个部分，进而将原始的问题转换为了两个简易的子问题（式 [5][6]），这两个简易的子问题可以容易求解，接下来分析子问题的显式解。

对于子问题式 [5]

$$\min_{C_i} \sum_{i=1}^{k} \sum_{j \in C_i} \| x_j - \mu_i^{t-1} \|^2 \tag{7}$$

该问题可以转化为等价形式

$$\min_{C_i} \sum_{j \in \Lambda} \sum_{i=1}^{k} \mathbb{I}\{j \in C_i\} \| x_j - \mu_i^{t-1} \|^2 \tag{8}$$

具有上界关系

$$\sum_{j \in \Lambda} \sum_{i=1}^{k} \mathbb{I}\{j \in C_i\} \| x_j - \mu_i^{t-1} \|^2 \geq \sum_{j \in \Lambda} \min_{i=1,2,\cdots,k} \| x_j - \mu_i^{t-1} \|^2 \tag{9}$$

等号当且仅当分割 C_i 满足

$$C_i^t = \{j : \| x_j - \mu_i^{t-1} \|^2 = \min_{s=1,2,\cdots,k} \| x_j - \mu_s^{t-1} \|^2\} \tag{10}$$

值得注意的是，该问题的最优解并不唯一，因为可能出现 $i_1 \neq i_2, \| x_j - \mu_{i_1}^{t-1} \|^2 = \| x_j - \mu_{i_2}^{t-1} \|^2 = \min_{s=1,2,\cdots,k} \| x_j - \mu_s^{t-1} \|^2\}$ 的情况。

接下来分析子问题式 [6]

$$\min_{\mu_i} \sum_{i=1}^{k} \sum_{j \in C_i^t} \| x_j - \mu_i \|^2 \tag{11}$$

由于 μ_i 间独立，且 C_i 为分割，则原问题可以转化为

$$\min_{\mu_i} \sum_{i=1}^{k} \sum_{j \in C_i^t} \| x_j - \mu_i \|^2 = \sum_{i=1}^{k} \min_{\mu_i} \sum_{j \in C_i^t} \| x_j - \mu_i \|^2 \tag{12}$$

基于此，对于每一类考虑分解问题

$$\min_{\mu_i} \sum_{j \in C_i^t} \| x_j - \mu_i \|^2 \tag{13}$$

为了方便考虑，假定 $\| \bullet \|$ 是 L2 范数，原问题为一个经典的凸优化问题，对于目标函数求导有

$$\frac{d}{d\mu_i} \sum_{j \in C_i^t} \| x_j - \mu_i \|_2^2 = \sum_{j \in C_i^t} 2 * (\mu_i - x_j) \tag{14}$$

令式 [14] 为 0 得到极值点为

$$\mu_i^t = \frac{1}{\| C_i^t \|} \sum_{j \in C_i^t} x_j = \underset{j \in C_i^t}{mean}\{x_j\} \tag{15}$$

其他度量情况类似。

基于式 [10][15] 的结果，机器学习框架下的优化流程转变为了

1. 初始化: 随机选取 μ_i^0，t=1，精度 ϵ;

2. 计算
$$C_i^t = \{j : \| x_j - \mu_i^{t-1} \|^2 = \min_{s=1,2,\cdots,k} \| x_j - \mu_s^{t-1} \|^2\} \tag{16}$$

3. 计算
$$\mu_i^t = \frac{1}{\| C_i^t \|} \sum_{j \in C_i^t} x_j = \underset{j \in C_i^t}{mean}\{x_j\} \tag{17}$$

4. 若迭代次数达到上限，或者 $\| \mu_i^t - \mu_i^{t-1} \| < \epsilon$，则结束迭代，结果为 μ_i^t。否则 $t = t + 1$，返回步骤 2。

该过程与传统的 K-means 算法一致，于是可知该机器学习框架与 K-means 算法的等价性和相容性。

一个良好的算法需要具有可穷性，证明 K-means 算法的可穷性等价于证明迭代优化算法的收敛性。下面讲证明迭代优化算法是弱收敛的。

[命题] 2.1 关于上方求得的均值迭代算法，随机选取初始参数 μ_i^0，满足 $\lim_{t \to +\infty} \sum_{i=1}^{k} \sum_{j \in C_i^t} \| x_j - \mu_i^t \|^2$ 收敛。

[证明] 2.1 记

$$g(C_i, \mu_i) = \sum_{i=1}^{k} \sum_{j \in C_i} \| x_j - \mu_i \|^2 \tag{18}$$

对于有限维度数据，有

$$g(C_i, \mu_i) \geq 0 \tag{19}$$

由优化问题式 *[5][6]* 有关系

$$\begin{cases} g(C_i^t, \mu_i^{t-1}) \leq g(C_i^{t-1}, \mu_i^{t-1}) \\ g(C_i^t, \mu_i^t) \leq g(C_i^t, \mu_i^{t-1}) \end{cases} \tag{20}$$

则有

$$g(C_i^t, \mu_i^t) \leq g(C_i^t, \mu_i^{t-1}) \leq g(C_i^{t-1}, \mu_i^{t-1}) \tag{21}$$

则有数列 $\{g(C_i^t, \mu_i^t)\}$ 单调下降，又由于 $g(C_i^t, \mu_i^t) \geq 0$，根据单调有界收敛原理，$\lim\limits_{t \to +\infty} g(C_i^t, \mu_i^t)$ 收敛，原命题成立。

值得注意的是，这并没有说明 μ_i 的收敛性，但是可以通过度量 $\sum\limits_{i=1}^{k} \sum\limits_{j \in C_i} \| x_j - \mu_i \|^2$ 的数值变化来终止算法。以上命题和分析说明了均值迭代算法的收敛性，进而证明了 K-means 算法的可穷性。

2.2.3 不同度量分析

不同度量对于算法的影响仅在于子问题式 [6] 的求解之上，接下来我们将具体分析几种常见度量的迭代形式。

对于 L1 度量表示为

$$\sum_{j \in C_i} \| x_j - \mu_i \|_1 = \sum_{j \in C_i} \sum_{s=1}^{C} |x_{js} - \mu_{is}| \tag{22}$$

求导有

$$\frac{\partial}{\partial \mu_{is}} \sum_{j \in C_i} \sum_{s=1}^{C} |x_{js} - \mu_{is}| = \sum_{j \in C_i} sig(\mu_{is} - x_{js}) \tag{23}$$

此时有

$$\mu_{is} = median_{j \in C_i} \{x_{js}\} \tag{24}$$

对于 L1 度量，构成 K-means 的变种 K-medians，该算法相较于 K-means 算法对于病态数据更加不敏感

对于 L_∞ 度量

$$\sum_{j\in C_i} \parallel x_j - \mu_i \parallel_\infty = \sum_{j\in C_i} \max_{s=1,2,\cdots,C} |x_{js} - \mu_{is}| \tag{25}$$

求导为

$$\frac{\partial}{\partial \mu_{is}} \sum_{j\in C_i} \max_{s=1,2,\cdots,C} |x_{js}-\mu_{is}| = \sum_{j\in C_i} \sum_{s=1}^{C} \mathbb{I}\{\mu_{is}-x_{js} = \max_{s=1,2,\cdots,C} |x_{js}-\mu_{is}|\} sig(\mu_{is}-x_{js}) \tag{26}$$

对于 Cosine 度量

$$\sum_{j\in C_i} \parallel x_j - \mu_i \parallel_\infty = \sum_{j\in C_i} \frac{x_j \bullet \mu_i}{\parallel x_j \parallel \parallel \mu_i \parallel} \tag{27}$$

求导有

$$\frac{\partial}{\partial \mu_{is_0}} \sum_{j\in C_i} \frac{x_j \bullet \mu_i}{\parallel x_j \parallel \parallel \mu_i \parallel} = \sum_{j\in C_i} \frac{1}{\parallel x_j \parallel} \left[\frac{x_{js_0}}{\parallel \mu_i \parallel} - \sum_{s=1}^{C} \frac{x_{js}\mu_{is}\mu_{is_0}}{\parallel \mu_i \parallel^3} \right] \tag{28}$$

该问题较难求出显式解，在此不多加分析。

通常情况下，复杂的度量意味着困难的求解，所以在实际应用中通常只使用 L2 和 L1 度量下的结果，也就是 K-means 和 K-medians 算法。

3 程序框图

训练过程伪代码为

Algorithm 1: Train a K-means model

Data: $X \in \mathbb{R}^{n*C}$: Data set;k: The number of clusters; l: The maximum steps

Result: $Cr \in \mathbb{R}^{k*C}$: The center of each cluster

Random select Cr from X;
step $\leftarrow 0$;
while *step*<=l **do**
    Update $C_i = \{j : \| x_j - Cr_i \| = \max_{s=1,2,\cdots,k} \| x_j - Cr_s \|\}$;
    Update $\hat{Cr}_i = mean_{j \in C_i} x_j$;
    if $\hat{Cr}_i$=Cr_i **then**
        return Cr;
    $Cr \leftarrow \hat{Cr}$;
    $step \leftarrow step + 1$;
return Cr;

预测过程伪代码为

Algorithm 2: Predict through a K-means model

Data: $X \in \mathbb{R}^{n*C}$: Data set;$Cr \in \mathbb{R}^{k*C}$: The center of each cluster

Result: $Y \in \{1, 2, \cdots, k\}^n$: The predict cluster of each data

Calculate the distance $d \in \mathbb{R}^{*n*k}$ between X and Cr;
Calculate $Y_j = \arg\max_{i=1,2,\cdots,k} d_{j,i}$;
return Y;

4 实验代码

代码使用 Python 实现，构建对象类 kmeans 实现算法，K-means 类代码为

Code Listing 1: K-means 类代码

```
class kmeans:
    def __init__(self, k=3, distance_type='l2'):
        self.k = k
```

```
        self.centers = None
        self.distance_func = None
        self.update_func = None
        if distance_type=='l2':
            self.distance_func = l2_distance
            self.update_func = l2_update
        elif distance_type=='l1':
            self.distance_func = l1_distance
            self.update_func = l1_update

    def train(self, a, lim_iter=100):
        n =  len(a)

        # init
        center_ids = np.random.choice(n, self.k)
        centers = np.array([a[i] for i in center_ids])
        # print(centers)
        run_flag = True
         iter = 0

        while run_flag and ( iter < lim_iter):
             iter += 1
            dis = self.distance_func(a, centers)
            _class = np.argmin(dis, axis=1)
            new_centers = self.update_func(a, _class, self.k)
            if (new_centers == centers). all():
                run_flag = False
            else:
                run_flag = True
            centers = np.array(new_centers)
        self.centers = centers
        return centers

    def predict_kmeans(self, X):
        assert self.centers is not None, 'Please␣train␣k-means'
        dis = self.distance_func(X, self.centers)
        y = np.argmin(dis, axis=1)
        return y
```

L2 度量下距离计算和均值更新代码为

Code Listing 2: L2 度量下距离计算和均值更新代码

```
def l2_distance(a, centers):
    """
```

```
    Calculate L2 Distance
    :param a: N*C
    :param centers: K*C
    :return: N*K
    """
    k =  len(centers)
    delta = (np.repeat(np.expand_dims(a, axis=1), k, axis=1) - centers) # N*k*C
    # dis = np.einsum('ijk,ijk->ij', delta, delta)
    dis = np. sum(np. abs(delta)**2, axis=2)
    return dis

def l2_update(a, _class, k):
    new_centers = []
    for i in  range(k):
        new_centers.append(np.mean(a[_class == i], axis=0))
    return np.array(new_centers)
```

L1 度量下距离计算和均值更新代码为

Code Listing 3: L1 度量下距离计算和均值更新代码

```
def l1_distance(a, centers):
    """
    Calculate L1 Distance
    :param a: N*C
    :param centers: K*C
    :return: N*K
    """
    k =  len(centers)
    delta = (np.repeat(np.expand_dims(a, axis=1), k, axis=1) - centers)  # N*k*C
    dis = np. sum(np. abs(delta), axis=2)
    return dis

def l1_update(a, _class, k):
    new_centers = []
    for i in  range(k):
        new_centers.append(np.median(a[_class == i], axis=0))
    return np.array(new_centers)
```

实验指标计算代码为（具体含义见下一部分介绍）

```
def evaluate_external(Y_pred, Y_gt, k=None):
    RI = rand_score(Y_gt, Y_pred)
    ARI = adjusted_rand_score(Y_gt, Y_pred)
    h, c, v = homogeneity_score(Y_gt, Y_pred), completeness_score(Y_gt, Y_pred),
        v_measure_score(Y_gt, Y_pred)
```

```
print('Evaluation')
print('␣RI', RI, 'ARI', ARI)
print('␣H', h, 'C', c, 'V', v)


hashs =  list( range(k))
acc = 0
for perm in itertools.permutations(hashs, k):
    _Y = np.zeros_like(Y_pred)
    for i in  range(k):
        _Y[Y_pred==i] = perm[i]
    # print(_Y)
    acc =  max(np. sum(_Y==Y_gt), acc)
print('␣Acc', acc/ len(Y_pred)*100, '%')
```

5 实验结果与分析

5.1 实验指标

本实验采用现代的无监督聚类外部指标来进行评价，实验中采用的数据的标签均有观测，也就是说数据集为 $D = \{(X, Y)\}$，其中 X 为数据特征，Y 为标签，但是在训练时仅使用 $\{X\}$ 进行训练，评价时与 Y 进行比较。为了测试模型的泛化性能，训练集和测试集的分配比例为 8:2。

具体而言，实验中采用 Adjusted Rand Index(ARI)、Homogeneity 度量 (H)、Completeness 度量 (C)、v 测度 (V) 和准确率 (Acc) 进行模型效果评价。

ARI 指标由 Rand Index(RI,[1]) 改进而来，通过两两比较来衡量聚类分配与真实类标签之间的相似性，表示为

$$RI = \frac{A + B}{\binom{n}{2}} \tag{29}$$

其中 A 是具有相同类标签且属于同一聚类的点对的数目，B 是具有不同类标签且属于不同聚类的点对的个数，n 是总点数。RI 虽然可以刻画聚类效果，但是也有一定的缺陷，即使对于随机的簇分配，它也可以得到很高的值，特别是当簇数量很大时。这是因为当聚类数量增加时，随机将不同标签的点分配给不同聚类的概率增加。因此特定的 RI 值可能是模糊

的，因为不清楚分数中有多少是偶然的，多少是实际一致的。原式改进为 ARI 指标

$$ARI = \frac{RI - \mathbb{E}RI}{\max(RI) - \mathbb{E}RI} \tag{30}$$

其中 $\mathbb{E}RI$ 为随机聚类分配下 Rand 指数的期望值，$\max(RI)$ 表示最大可能的配对数。ARI 值的范围从-1 到 1，其中 1 表示簇分配和类标签之间完全一致，0 表示随机一致，负值表示一致性低于偶然预期。

同质性 Homogeneity 度量每个簇是否只包含单个类的成员，表示为

$$\begin{aligned} H &= 1 - \frac{H(C|K)}{H(C)} \\ H(C|K) &= \sum_{k=1}^{|K|} P(K=k) H(C|K=k) \\ &= -\sum_{k=1}^{|K|}\sum_{c=1}^{|C|} \frac{n_{c,k}}{n} log \frac{n_{c,k}}{n_k} \\ H(C) &= -\sum_{c=1}^{|C|} \frac{n_c}{n} log \frac{n_c}{n} \end{aligned} \tag{31}$$

其中 C 代表真值类标签，K 表示算法分配的聚类标签，$n_{c,k}$ 表示分配给 k 簇的 c 类样本数，n_k 为 k 簇的样本数，n_c 为 c 类的样本数。同质性评分范围为 0　1，其中 1 表示完全同质性，即每个簇只包含单个类的成员。

完整性 Completeness 度量给定类的所有成员是否被分配到同一个簇，表示为

$$\begin{aligned} C &= 1 - \frac{H(K|C)}{H(K)} \\ H(K|C) &= -\sum_{k=1}^{|K|}\sum_{c=1}^{|C|} \frac{n_{c,k}}{n} log \frac{n_{c,k}}{n_c} \\ H(K) &= -\sum_{c=1}^{|K|} \frac{n_k}{n} log \frac{n_k}{n} \end{aligned} \tag{32}$$

完整性的范围从 0 到 1，其中 1 表示完全完整，即每个类成员被分配到单个簇。

V-measure 是同质性和完备性的调和平均值，它可以提供一个单一的分数来评估聚类性能，表示为

$$V = 2\frac{HC}{H + C} \tag{33}$$

通过使用调和均值，V-measure 惩罚同质性和完整性之间的不平衡，鼓励更均匀的聚类性能。

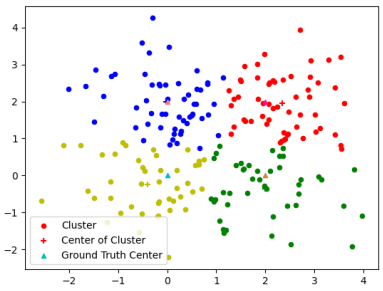
Acc 将聚类问题视为一个分类问题，并通过分类问题中的准确率指标来评价，表示为

$$Acc = \max_{p\ is\ a\ permutation} \frac{1}{\parallel \Lambda \parallel} \sum_{i \in \Lambda} \mathbb{I}\{pred_i = p_{gt_i}\} \tag{34}$$

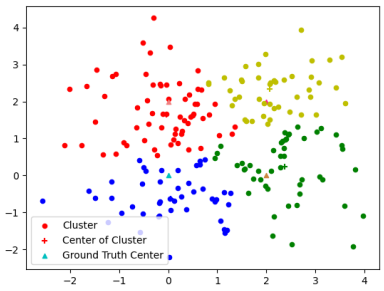
其中 p 是一个容量为 $\parallel \Lambda \parallel$ 的排列，$pred_i$ 表示 x_i 的预测聚类序号，gt_i 表示 x_i 的真实分类标签。Acc 范围从 0 到 1，越接近 1 表示聚类的结果与真实情况越接近。

5.2 随机二维数据分类及可视化

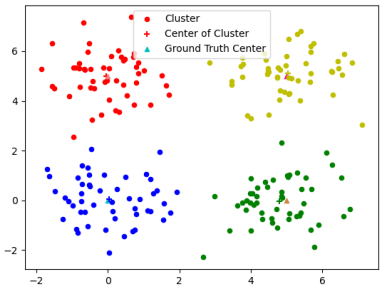
数据空间为 $\mathbb{R}^2$，共有 4 类数据，分别采样自 $N([0,0]^T, \mathbf{I})$,$N([s,s]^T, \mathbf{I})$, $N([0,s]^T, \mathbf{I})$ 和 $N([s,0]^T, \mathbf{I})$，其中 s 为尺度系数。为了可视化聚类结果，每类采样 50 个数据，尺度系数分别设置为 2 和 5，结果见图1。通过结果可以发现，无论是何种超参数设置，K-means 都能较好地完成聚类。但是对于尺度系数较大的数据，因为本身数据间分布区别较大，因此聚类较为容易，进而无论是 L1 度量还是 L2 度量，聚类结果均相同。但是对于小尺度的数据，不同类别间的数据混杂严重，L2 和 L1 之间的聚类结果在类别边界处出现较大的差异。



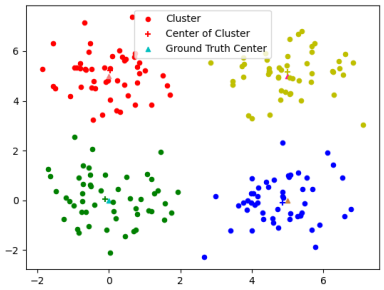
(a) L2 度量，s=2



(b) L1 度量，s=2



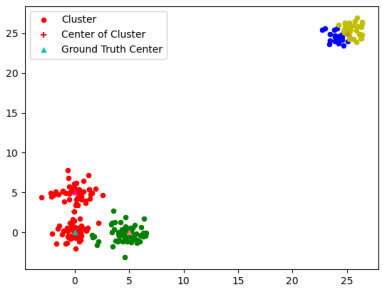
(c) L2 度量，s=5



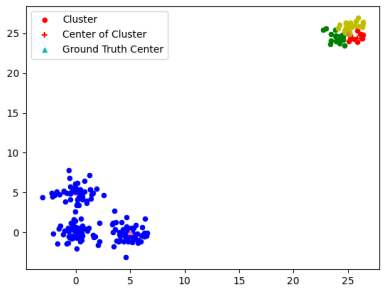
(d) L1 度量，s=5

图 1: K-means 结果

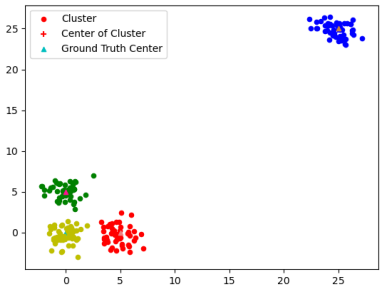
K-means 算法对于初始值的选取较为敏感，不同的初值会导致不同的结果。在特殊构造的数据上进行 K-means 算法，可能会出现如图2的情况，良好的聚类结果应当如图2(c)，但是如果初值的选取不当，会导致出现图2(a) 和图2(b) 的情况。可以预见的是，这一问题在类别边界清晰和类别数较多的数据上更为明显。基于此，在使用 K-means 算法的时候建议运行多次后采用最优结果，或者使用其他算法获得一个近似结果再通过 K-means 进行优化。



(a) 聚类结果



(b) 聚类结果



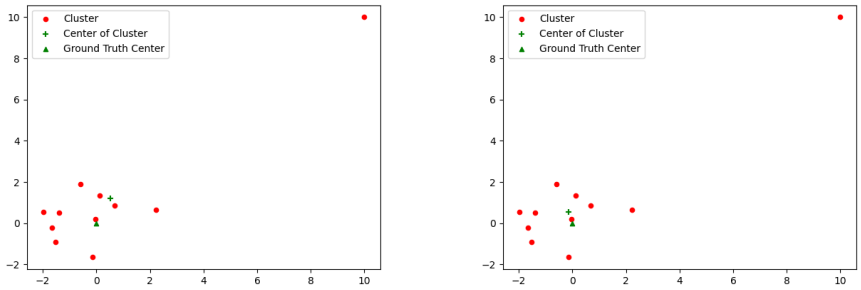
(c) 聚类结果

图 2: K-means 的初值影响

K-means 和 K-medians 在机器学习框架下研究时区别仅在于数据空间的度量设置, K-means 采用的是 L2 度量而 K-medians 度量。如表2所示, 对于一个无离群点 (小概率采样概率) 的数据而言, L2 度量的 K-means 算法的聚类效果是更优的。当数据出现离群点时, K-means 算法预测的聚类中心可能与实际情况会大相径庭, 而 K-medians 算法在牺牲了聚类精度的同时提高了预测结果的鲁棒性, 图3表明当出现离群点的时候, K-means 算法预测的聚类特征与理想特征之间的差距要远大于 K-medians 算法预测的聚类特征。因而在实际应用中需要根据实际情况选取合适的聚类算法。

度量	ARI	H	C	V	Acc
L1	**0.6025**	**0.6034**	**0.6006**	**0.6020**	**83.0000%**
L2	0.5363	0.5423	0.5384	0.5404	80.0000%

表 2: 实验结果，数据采样自 $N([0,0]^T, \mathbf{I})$、$N([2.5,0]^T, \mathbf{I})$、$N([0,2.5]^T, \mathbf{I})$ 和 $N([2.5,2.5]^T, \mathbf{I})$，训练集每类采样 100 个数据，测试集每类采样 25 个数据，保证数据无离群点，对于数据每种参数设置进行 10 次实验，取最优结果



(a) L2 度量　　(b) L1 度量

图 3: 离群点情况

5.3 鸢尾花数据集结果

鸢尾花数据集 [2] 是由英国统计学家、优生学家和生物学家 Ronald Fisher 引入的一个多变量数据集，因为 Edgar Anderson 收集数据是为了量化三个相关物种鸢尾花的形态变化，它有时被称为安德森的鸢尾花数据集。数据集中三种鸢尾花中有两种是在加斯皮纳半岛采集的，且保证了”都来自同一牧场，同一天采摘，同一时间由同一个人用同一仪器测量”。

数据集中共包含 150 个有效数据，每条数据收集了萼片长度、萼片宽度、花瓣长度和花瓣宽度四个特征，均使用实数表示，并标注了每个数据所属类别，类别为 setosa、versicolor、virginica 三者之一。

实验中将数据集按照 8：2 的比例分为了训练集和测试集，K-means 模型在训练集上进行训练，并在测试集上进行指标的计算。结果见表3。结果表明，效果最好的模型采用 L2 度量，也就是传统的 K-means 算法，在测试集上的准确率可以达到 94%，但是也有可能出现上一章节中提及的

问题，出现极大的效果衰减，L2 度量的第 2 组效果远低于其他组别的结果，预测是出现了初值选取的问题。

度量	组别	ARI	H	C	V	Acc
	1	**0.8462**	**0.8606**	**0.8567**	**0.8586**	**94.3662%**
L2	2	0.3989	0.5075	0.6248	0.5600	56.3380%
	3	0.8148	0.7820	0.8077	0.7947	92.0635%
	1	0.8160	0.8234	0.8503	0.8366	92.9577%
L1	2	0.7077	0.7616	0.7995	0.7801	88.7324%
	3	0.8148	0.7820	0.8077	0.7947	92.0635%

表 3: 鸢尾花数据集实验结果

6 遇到的问题及其解决措施

K-means 算法的实现较为简单，整个实验基本未出现什么大的问题，主要的阻碍就是算法本身依赖初始聚类中心的缺陷，这意味着同一个实验的结果是不确定的，为了真正验证算法的性能，需要同一组数据进行多次数据才能真正确定。

参考文献

[1] W. M. Rand (1971). Objective criteria for the evaluation of clustering methods” . Journal of the American Statistical Association. 66 (336): 846–850.

[2] Fisher, R. A. . (1936). The use of multiple measurements.