第一次大作业

赵丰

2018 年 12 月 4 日

1 概要

本文自己实现了 k-means 算法，并根据自己提出的标签差异个数以及目标函的相对差两个一致性指标对比了自己的实现和工业级实现在结果上的差异，使用 Silhouette 作图来评估聚类的好坏。通过实际实验发现 Affinity Propogation 算法各方面都比较差。最后我们将 k-means 聚类算法用于图像分割的情形，了解 k-means 聚类算法的实际应用。

2 k-means 标准算法描述

给定初始 k 个中心 $m_1^{(1)}, \ldots, m_k^{(1)}$，k-means 算法在下面两个步骤间迭代 [1]：

赋值： $S_i^t = \{x_p : \|x_p - m_i^{(t)}\| \leq \|x_p - m_j^{(t)}\| \forall j, 1 \leq j \leq k\}$

更新： $m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$

3 k-means 标准算法分析

3.1 正确性

考虑目标函数

$$f(x) = \sum_{i=1}^{k} \sum_{x \in S_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2 \tag{1a}$$

$$= \sum_{j=1}^{n} \sum_{i=1}^{k} \|\boldsymbol{x}_j - \boldsymbol{\mu}_i\| \mathbb{1}_{\boldsymbol{x}_j \in S_i} \tag{1b}$$

$$\dot{\bigcup_{i=1}^{k}} S_i = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$$

希望求解最优的 k-划分和 k 个点的坐标。该问题联合求解（全局最优解）是 NP-Hard 的（组合优化问题一般都有这个特征），但如果给定 k-划分求解 k 个点的坐标，根据(1a)式，由二次函数极值的性质易得其为每个划分中各点的质心；如果给定 k 个点的坐标，根据(1b)式，每个 $\boldsymbol{x}$ 应归入距离最近的中心所在的类。k-means 标准算法即利用了这样的关系迭代求解出**局部最优值**。

我们考察在 k-means 标准算法迭代过程中取值的变化。赋值步骤相当于 k 个点的坐标不动，对 S_i 进行重新划分，根据(1b)式从而使得目标函数减小；更新步骤相当于划分不变，重新求解 k 个点的坐标，根据(1a)式从而使得目标函数减小。根据单调数列收敛的结论，算法的收敛性得到保证。注意到这里使用 L_2 范数即欧氏距离才可以保证这一点，因为欧氏距离下各划分的质心才是给定 S_i 下(1a)式的最优解。

3.2 算法时间复杂度

假设其主要逻辑中两个步骤共重复迭代 t 次，赋值操作要对 k 个集合进行，对每个集合又要把所有的数据点（设为 n 个）遍历一遍，设数据为 d 维，则算法复杂度为 $O(tknd)$。

3.3 实际考虑

使用脚本编程语言实现 k-means 算法时，更新步骤可以通过调用内置矩阵运算加速，避免自己写循环。在这一点上，csdn 的某个范例 [2] 做得很

不好，可能会误导初学者。但即使在用户程序中使用矩阵加速，自己实现的也只能供学习、科研目的使用。

3.4 算法验证

将自己实现的 k-means 算法与工业级的实现进行对比有多种不同的方法，这里我们采用对比数据点的标签差异的个数以及目标函数（(1a)式）的相对差两个指标。由于聚类本身的标签是相对的，我们采用对 k 个标签进行全枚举的方法寻找两种实现间最佳的匹配。

用数学的语言表示，我们给出下面的定义：

定义 1 (k-标签数列)**.** 若有限长数列 ℓ 每个位置的元素取自 1 到 k 的正整数，则称该有限长数列为 k-标签数列, 所有 k-标签数列组成集合 S。

从排列的观点，我们知道 k-全排列函数 p 是定义域和值域都是 1 到 k 的整数的集合的双射。我们可以把 k-全排列函数（函数全体集合记为 P）拓展到 k-标签数列上。

定义 2 (作用在 k-标签数列上的 k-全排列函数)**.** 设 p 是 k-全排列函数，ℓ 是 k-标签数列，定义 $\tilde{p}: S \to S, p(\ell) = \ell'$, 其中 $\ell'(i) = p(\ell(i))$. $\tilde{p}$ 称为由 p 诱导出来的作用在 k-标签数列上的 k-全排列函数。

我们知道离散度量 $d: X \times X \to \{0,1\}$ 是空间 X 上的一种度量，定义为 $d(x,y) = \begin{cases} 0 & x = y \\ 1 & x \neq y \end{cases}$。$m$ 个数据可以看成 m 个 X 的笛卡尔乘积空间，其中 $X = \{1,2,\ldots,k\}$，从而 $\tilde{d}: X^d \times X^d \to \{0,1\}^d, \tilde{d}(\boldsymbol{x},\boldsymbol{y}) = (d(x_1,y_1),\ldots,d(x_n,y_n))$.

定义 3 (聚类算法标签差异度)**.** 给定两个不同实现细节的 k-means 的聚类算法，对 m 个数据进行 k 分类，标签分别为 ℓ_1, ℓ_2。标签差异度 e 定义为：

$$e = \frac{1}{m} \min_{p \in P} \|d(p(\ell_1), \ell_2)\|_1 \tag{2}$$

评论 1. 在定义3中 $d(p(\ell_1),\ell_2)$ 是 k 维向量，每个分量均为 0 或 1,$\|\cdot\|_1$ 是取 L_1 范数，即将各元素取绝对值相加。

下面的定理进一步说明了聚类算法标签差异度的合理性：

定理 1.

$$\min_{p\in P}\|d(p(\ell_1),\ell_2)\|_1 = \min_{q\in P}\|d(\ell_1,q(\ell_2))\|_1 = \min_{p,q\in P}\|d(p(\ell_1),q(\ell_2))\|_1 \tag{3}$$

为证明定理1，我们需要下面这个引理：

引理 1. $\forall p\in P, \exists q\in P,\ s.t.\ d(p(\ell_1),\ell_2) = d(\ell_1,p(\ell_2))$

证明. $d(p(\ell_1),\ell_2) = d(p(\ell_1),p(p^{-1}(\ell_2))) = d(\ell_1,p^{-1}(\ell_2))$, 因此只需取 $q = p^{-1}$。 □

下面我们给出两个正数的相对差的定义：

定义 4 (相对差)**.** 设 $a,b>0$, a 和 b 的相对差定义为：

$$r(a,b) = 2\frac{|a-b|}{a+b} \tag{4}$$

根据文献 [4] 中的相关讨论和证明，我们有

定理 2. 相对差是 $\mathbb{R}^+$ 上的度量。

评论 2. 我们用目标函数相对差来衡量不同实现的 k-means 算法结果是否很接近。

4 性能评价

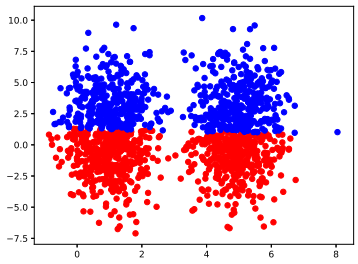
存在着多种评价聚类算法的指标，这里我们仅采用 Silhouette 系数, 一般而言，高的 Silhouette 系数对应着较好的聚类效果。

5 实验

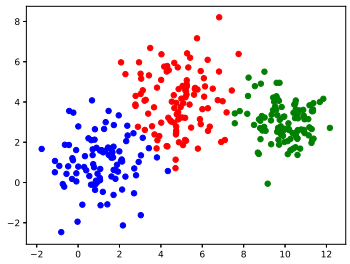
5.1 k-means 聚类分析

我们使用混合高斯模型产生二维数据点，针对方差矩阵非均匀的情况，受 [3] 的启发，我们做了图1a所示的数值实验，验证了对于非均匀的数据点使用 k-means 算法可能得到与直觉不一致的结果。

但对于通常的情形，比如图1b所示的数值实验，k-means 给出了比较合理的结果。



(a) 两类椭圆形数据点



(b) 三类各向同性数据点

	三类各向同性数据点	两类椭圆形数据点
标签差异度	0.1%	6.6%
目标函数相对差	1e-4	0.02
平均迭代次数	6.2	11.5

表 1: k-means 算法比较

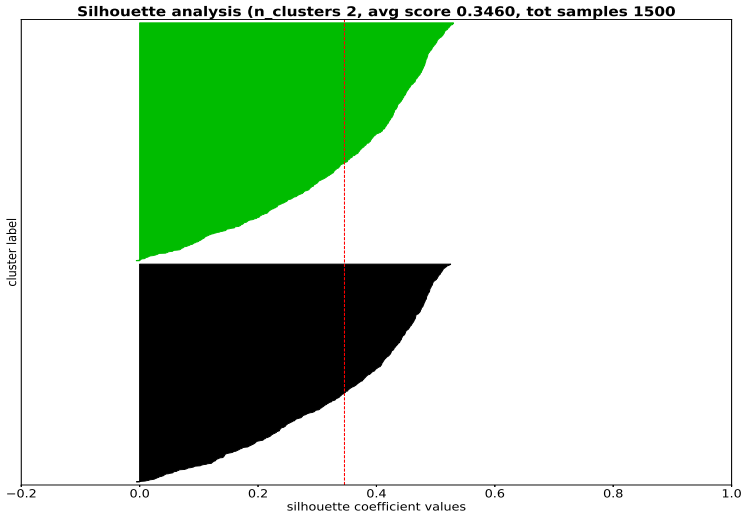
根据3.4小节提出的指标，我们比较了 k-means 标准算法的工业级实现和自己的实现，结果如表1所示。表1所示的结果为 30 次聚类结果取平均得到的。

三类各向同性数据点是比较适合于用 k-means 算法进行聚类的人工数据集，其标签差异度仅为 0.1% 左右，这意味着平均意义上 1000 个数据点有 1 个点两种算法聚类结果不一致，可见我们实现的算法与工业级算法在结果上的高度稳合性。目标函数相对差只有 1e-4 左右，说明目标函数有局部最优值，但在比较适合用 k-means 聚类的情形下不同实现（初值点不一样）得到的局部最优值非常接近。

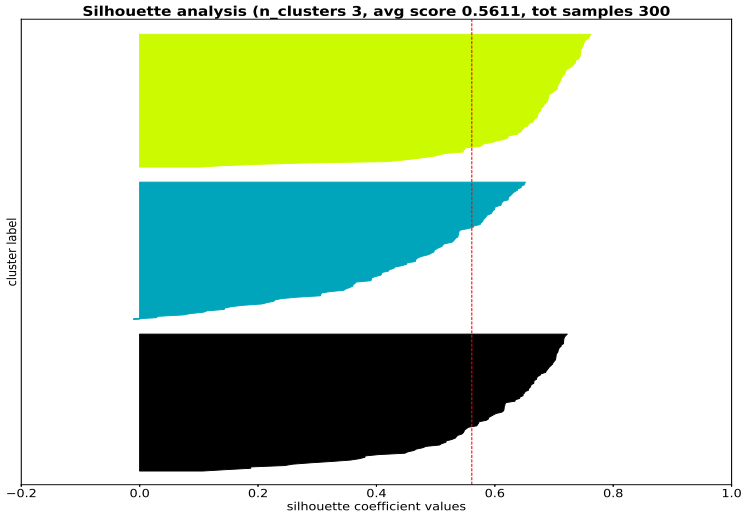
两类椭圆形数据点，如图1a所示，是不适合用 k-means 进行聚类的。如果使用的话，如表1所示，标签差异度和目标函数相对差均较大。这说明了问题对初值点比较敏感，目标函数各局部最优值相差较大，在图1a的 x 轴附近的数据点在不同的聚类实例中会随机地分到两个类中从而导致了总体标签差异度较大。

表1还展示了 k-means 算法在实验人工数据集上的收敛速度，在比较合适的数据集上仅用 6 次多就收敛，而在不合适的数据集上要 11.5 次才收敛。

我们对均一和非均一两组人工数据集的 k-means 聚类结果分别做出



(a) 两类椭圆形数据点



(b) 三类各向同性数据点

图 2: Silhouette 作图

silhouette 图，如2a所示。由图可见，对于两类椭圆形数据点 silhouette 值较低，平均只有 0.34，聚类效果不好；而对于三类各向同性数据点，silhouette 得分平均值为 0.56，相比较而言较好。

5.2 工业级 k-means 标准算法代码阅读

通过阅读工业级 k-means 标准算法的源码 [5]，我总结出以下算法独有的特点：

1. 换不同的初值点多算几次取目标函数最小的作为返回结果
2. 默认使用 k-means++ 选取初值点加快收敛速度
3. 对于稀疏矩阵和稠密矩阵区别处理

而我实现的算法初值是随便选取的，只算一次，不考虑矩阵的稀疏性。

工业级 k-means 标准算法的源码对于迭代过程中可能出现的 S_i 为空（求质心不能除以零）的情况采用了**经验式**的处理方法，即取那些距现有中心最远的点单独归为一类，该点即为此类的质心。而我在算法实现中采用不更新质心的经验式方法，即下一轮的迭代的质心就是上一轮的。

6 Affinity Propagation 算法

我们用 Affinity Propagation 算法对两类椭圆形数据点和三类各向同性数据点进行聚类，发现存在如下问题：

1. 理论时间复杂度是 $O(tdn^2)$，t 为迭代次数，d 为数据维数，而 n 为数据个数。实际运行结果的感受也是 Affinity Propagation 远慢于 k-means 。
2. Affinity Propagation 无须事先指定聚类数 k 的个数，但个人感觉在数据看上去不是分的很开的情况下（三类各向同性数据点）默认参数 $s(i,i)$ 产生的 k 太大，为使聚类数为 3, 需要用区间搜索法调参，对不同的问题又需要重新调参，增加了计算开销。
3. Affinity Propagation 即使在有阻尼系数的情况下收敛速度可能会很慢。在两类各向同性数据点的情形下，实际测试发现将默认的最大迭代次数调节到 1000 时仍不能保证收敛。

4. 从分类的结果来看，对于两类各向同性数据点，产生的分类结果和 k-means 一致而与预期的类不一致，说明 Affinity Propagation 也不能很好地处理非均匀数据分布。

基于我们的实际测试，我们得出的结论是 Affinity Propagation 不是一个很好的算法，效率低、聚类结果差、调参繁琐，对部分问题不适用。

7 图像分割

对于彩色图像，其三个通道的色彩值在一定程度上代表了图像的局部特征，我们利用这一点在 RGB 色彩空间对给定的图片用 k-means 算法进行聚类，输入图像如图 3a所示，实验结果如图 3b所示。

通过实际测试发现存在如下问题：

1. 由于图像每个像素点均为一个待聚类的数据点，规模很大，即使用 k-means 聚类单次迭代的速度也很慢。

2. 收敛速度很慢，我们设置最大迭代次数为 200 次仍不收敛, 聚类效果欠佳。

参考文献

[1] 维基百科 k-means

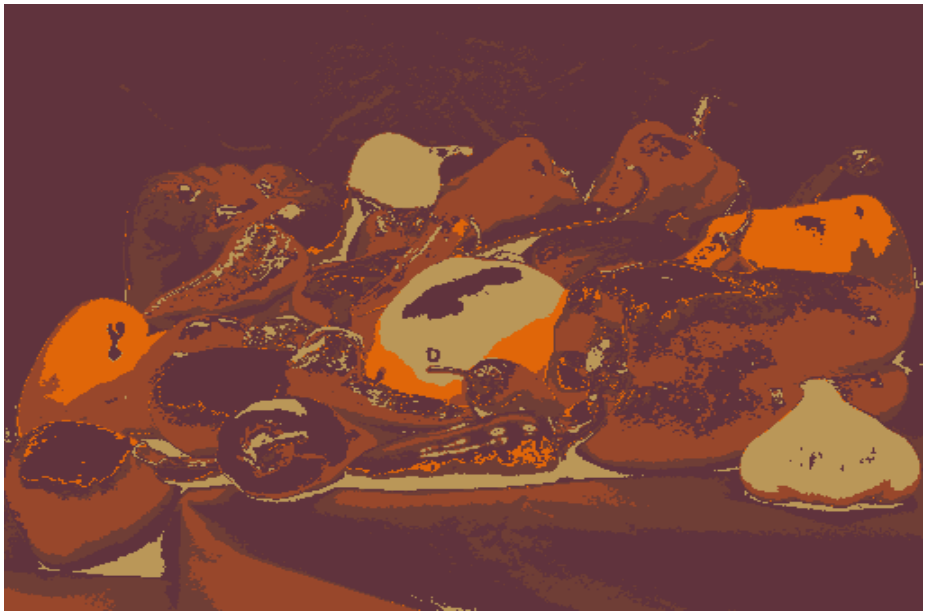
[2] 机器学习算法与 Python 实践之（五）k 均值聚类（k-means）

[3] Demonstration of k-means assumptions

[4] A Multiplicative Metric

[5] sklearn k-means implementation

(a) 输入图像



(b) $k = 5$ 聚类效果图

图 3: k-means 图像分割