K-means 实现图像聚类

一．实验基本思路

1. 获取数据集
2. 对数据集 kmeans 均值聚类
3. 计算聚类结果的 DBI 值并评估不同 k，L1 和 L2 范数的性能
4. 对数据集可视化输出

二．数据处理方式

1. CIFAR-10 数据集下载；http://www.cs.toronto.edu/~kriz/cifar.html ，下载 python 版本数据。

Download

If you're going to use this dataset, please cite the tech report at the bottom of this page.

Version	Size	md5sum
CIFAR-10 python version	163 MB	c58f30108f718f92721af3b95e74349a
CIFAR-10 Matlab version	175 MB	70270af85842c9e89bb428ec9976c926
CIFAR-10 binary version (suitable for C programs)	162 MB	c32a1d4ab5d03f1284b67883e8d87530

图表 1：可下载数据集

2. 下载后的数据集为矩阵集合形式，其中包含 5 个训练 batch 和一个测试 batch 的 mat 文件，每个 batch 包含 10000 幅图像，共有十种类别的图像。

使用 pickle 包输入二进制数据并返回一个字典变量：

```
def unpickle(file):
    import pickle
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict
```

图表 2：pickle 包的使用

3. 使用 for 循环并通过字典的映射把 1*3072 的图片和标签初始化到 10000*3072 和 10000*1 的两个 numpy 数组中。

三．算法

1. 概述

K-means 聚类算法也称 k 均值聚类算法，是集简单和经典于一身的基于距离的聚类算法。它采用距离作为相似性的评价指标，即认为两个对象的距离越近，其相似度就越大。该算法认为类簇是由距离靠近的对象组成的，因此把得到*紧凑且独立的簇作为最终目标。*

2. 算法核心思想

K-means 聚类算法是一种迭代求解的聚类分析算法，其步骤是随机选取 K 个对象作为初始的聚类中心，然后计算每个对象与各个种子聚类中心之间的距离，把每个对象分配给距离它最近的聚类中心。聚类中心以及分配给它们的对象就代表一个聚类。每分配一个样本，聚类的聚类中心会根据聚类中现有的对象被重新计算。这个过程将不断重复直到满足某个终止条件。终止条件可以是没有（或最小数目）对象被重新分配给不同的聚类，没有（或最小数目）聚类中心再发生变化，误差平方和局部最小。

3. 算法实现步骤

1、首先确定一个 k 值，即我们希望将数据集经过聚类得到 k 个集合。
2、从数据集中随机选择 k 个数据点作为质心。
3、对数据集中每一个点，计算其与每一个质心的距离（如欧式距离），离哪个质心近，就划分到那个质心所属的集合。
4、把所有数据归好集合后，一共有 k 个集合。然后重新计算每个集合的质心。
5、如果新计算出来的质心和原来的质心之间的距离小于某一个设置的阈值（表示重新计算的质心的位置变化不大，趋于稳定，或者说收敛），我们可以认为聚类已经达到期望的结果，算法终止。
6、如果新质心和原质心距离变化很大，需要迭代 3~5 步骤。

4.K-means 术语：

簇：所有数据的点集合，簇中的对象是相似的。

质心：簇中所有点的中心（计算所有点的中心而来）

5. K-means 算法优缺点

优点：

1、原理比较简单，实现也是很容易，收敛速度快。

2、当结果簇是密集的，而簇与簇之间区别明显时，它的效果较好。

3、主要需要调参的参数仅仅是簇数 k。

缺点：

1、K 值需要预先给定，很多情况下 K 值的估计是非常困难的。

2、K-Means 算法对初始选取的质心点是敏感的，不同的随机种子点得到的聚类结果完全不同 ，对结果影响很大。

3、对噪音和异常点比较的敏感。用来检测异常值。

4、采用迭代方法，可能只能得到局部的最优解，而无法得到全局的最优解。

算法思想：

输入： 样本集 $D=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_m\}$;
聚类簇数 k.

过程：

1: 从 D 中随机选择 k 个样本作为初始均值向量 $\{\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,\ldots,\boldsymbol{\mu}_k\}$
2: **repeat**
3: 令 $C_i=\varnothing\ (1\leqslant i\leqslant k)$
4: **for** $j=1,2,\ldots,m$ **do**
5: 计算样本 $\boldsymbol{x}_j$ 与各均值向量 $\boldsymbol{\mu}_i\ (1\leqslant i\leqslant k)$ 的距离: $d_{ji}=||\boldsymbol{x}_j-\boldsymbol{\mu}_i||_2$;
6: 根据距离最近的均值向量确定 $\boldsymbol{x}_j$ 的簇标记: $\lambda_j=\arg\min_{i\in\{1,2,\ldots,k\}} d_{ji}$;
7: 将样本 $\boldsymbol{x}_j$ 划入相应的簇: $C_{\lambda_j}=C_{\lambda_j}\bigcup\{\boldsymbol{x}_j\}$;
8: **end for**
9: **for** $i=1,2,\ldots,k$ **do**
10: 计算新均值向量: $\boldsymbol{\mu}_i'=\frac{1}{|C_i|}\sum_{\boldsymbol{x}\in C_i}\boldsymbol{x}$;
11: **if** $\boldsymbol{\mu}_i'\neq\boldsymbol{\mu}_i$ **then**
12: 将当前均值向量 $\boldsymbol{\mu}_i$ 更新为 $\boldsymbol{\mu}_i'$
13: **else**
14: 保持当前均值向量不变
15: **end if**
16: **end for**
17: **until** 当前均值向量均未更新

输出： 簇划分 $\mathcal{C}=\{C_1,C_2,\ldots,C_k\}$

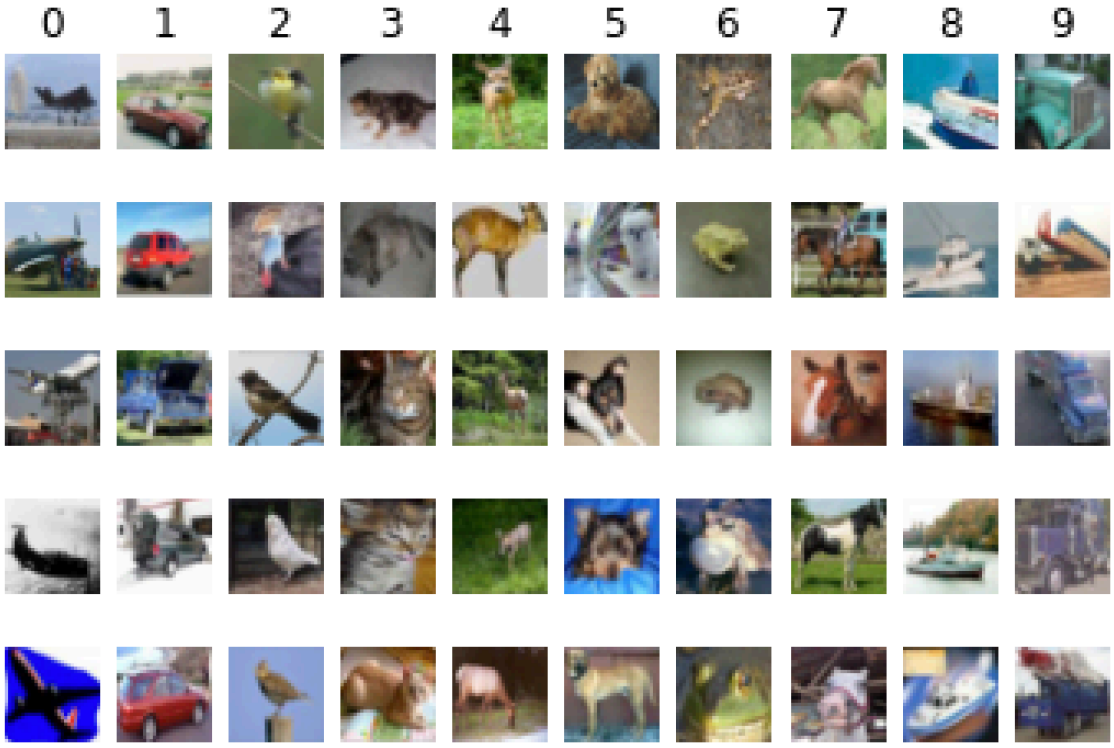
图表 3：k-means 算法

四．实现语言

使用 python 语言，调用 pickle 包（解压二进制数据），numpy（矩阵计算）matplotlib（图片的可视化）

五．实验效果

1.数据集的可视化（按标签分类）



图表 4：可视化数据（数据标签）

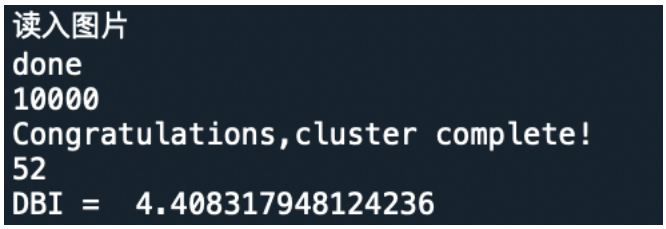
2.用 L1 和 L2 范数进行聚类并比较

（1）k=10:

```
读入图片
done
10000
Congratulations,cluster complete!
36
DBI =  2.3563734447922675
```

图表 5 L1 范数效果

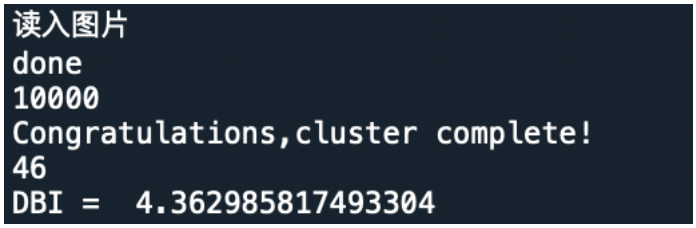
出现有些簇并不能分到图片的情况导致无法计算 DBI 和图像的显示，把未分到图片的簇的内部距离定义为 0，之后得到 DBI 约为 2.36。



图表 6 L2 范数效果

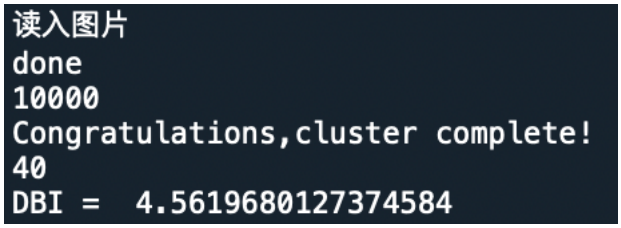
可以发现 L2 范数的性能是要优于 L1 范数的，因此之后选用 L2 范数进行 k 值的选择。

（2）k=9



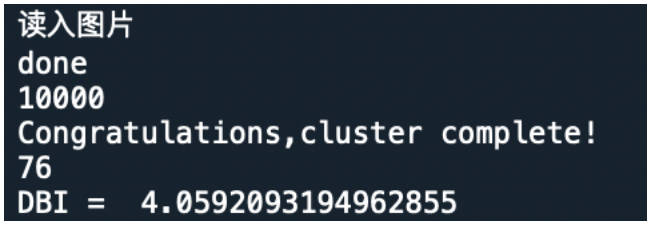
图表 7：k=9 效果

（3）k=8:



图表 8：k=8 效果

（4）k=7:



图表 9：k=7 效果

3.聚类结果的可视化（k=7，L2 范数）

图表 10：最优聚类可视化

六．性能评价

1. 因为 L1 范数会出现有些簇分不到图片的情况，因此本次都选择了 L2 范数。

2. 可以看出即使选择最优的模型，聚类的分类效果依旧不理想。这与一开始中心点的随机选择和图片的处理都是有关系的，我应该不能直接拿 RGB 图片的像素点进行聚类，需要对图片进行预处理突出图片的特征降低噪声的干扰。