第120天：机器学习算法之 K 均值聚类

原创 轩辕御龙 Python技术 2月5日

本文我们来学习一下另一种经常听到的机器学习算法—— K 均值聚类。

这个名字确实跟“K 近邻”有些相像，但是要明确的是，“K 近邻”中的“K”，指的是“与输入数据最接近的 K 个数据点”；而“K 均值聚类”中的 K，指的则是“**将一堆无标记数据划分为 K 个类别**”，其中这个“类别”通常被称为“**簇**”（cluster），即一簇花两簇花的簇。

而“均值”则更加直白：均值就是指的平均值。也就是每一簇数据的平均值，这个平均值就可以作为这一簇数据的中心点，用来判断其他数据与该簇数据的差异程度。

K 均值聚类的原理其实很好理解，“聚类”就是“将若干数据按类别聚在一起”，透露出的是“物以类聚，人以群分”的朴素哲学。算法的关键在于：我们应当以什么作为标准来判定一个数据与其他数据属于一个类别，也就是一个簇？

在前面我们介绍过“距离度量”的概念，若有必要可以参考。在 K 均值聚类的过程中，我们用来判别数据之间差异程度的方式就是考察二者之间的“距离”。

1. 算法实现

1.1 初始化

这一次我们用一个类来实现 K 均值聚类算法。

使用 K 均值聚类，首先需要对模型进行初始化，也就是按照要划分的簇数 K，在给定数据中随机选定 K 个数据，作为初始的各簇中心。

另外需要提到的是，在通常实现机器学习算法的时候，我们还需要对训练数据进行“预处理”，其中包括一个称为“**数据归一化**”的操作。这一步的目的是将各个数据统一到一个相同的数值范围内，使得不同数据之间具有可比性；否则，对于两个长度、重量分别为（20 m, 10000 g）、（1 m, 10500 g）的数据而言，长度的差异很容易就被重量的差异掩盖了。不过好在我们使用的 `iris` 数据集中用到的数据大体都在一个数量级上，可能会有一些缺陷，但不影响我们演示算法的实现，因此跳过了这个步骤。

```
import numpy as np
import pandas as pd
import random


class Clusters():
    def __init__(self, train_data, K):
        '''
        :params train_data: ndarray，训练数据
```

```
        :params train_data: ndarray. 训练数据.
        :params K: int. 要划分的簇的数量.
        '''
        super().__init__()

        self.train_data = train_data
        self.K = K
        # 标记聚类是否完成。具体的真假，取决于是否还存在需要从一个簇移动到另一个簇的数据
        self.finished = False

        # 随机选取 K 个数据作为各个簇的中心点
        index = random.sample(range(len(self.train_data)), self.K)
        self.centroid = train_data[index, 1:5]

        # 将训练数据均匀分配到各个簇，以便以同一的形式适用于数据的分配
        self.clusters = []
        offset = len(train_data) // self.K
        for i in range(self.K):
            start = offset * i
            if i < self.K-1:
                self.clusters.append(train_data[start:start+offset,:])
            else:
                # 最后一个簇包含剩下的所有数据
                self.clusters.append(train_data[start:,:])

    # 加载所要用到的数据集
    @staticmethod
    def getData():
        '''
        获取数据，返回值类型为 ndarray
        '''
        train_data = pd.read_csv('iris.csv').to_numpy()

        return train_data
```

这里的初始化方法需要从外部接受两个参数：train_data 和 K。train_data 是需要进行聚类的数据集，数据类型是 ndarray；而 K 则接受需要划分的簇的数目，数据类型为 int。

第 19 ~ 21行我们利用 `random` 模块中的随机取样功能，从 0 ~149 共计 150 个整数中随机抽取 K 个整数作为索引，对应的数据就是我们随机初始化的各簇中心。

第 23 ~ 32 行按 K 的大小将训练数据均匀分配到各个簇中去，这一步实际上是为了使属性 `clusters` 的格式与之后的“分配”环节相适应。

第 34 ~ 42 行定义了一个静态方法，用于获取要用到的训练数据。

1.2 分配

这一步我们需要分别计算各个数据与当前的各簇中心之间的距离，然后选取与各数据距离最小的簇作为目标簇，将数据移动到目标簇中，同时在原簇中删去相应数据。

也就是说这一步的要求是：对应于各簇的中心，所有数据应当各归其位，分属于最恰当的那个簇。

下面是方法的实现代码：

```
1    # 将各数据分配到每个簇中去
2    def assign(self):
3        self.finished = True
4        # data_index_list 和 target_index_list 分别记录“需要移动的数据在当前簇中的索引”以及“要移动到的目标簇
5        target_index_list = []
6        data_index_list = []
7        for i in range(self.K):
8            target_index_list.append([])
9            data_index_list.append([])
10
11       for cluster_index in range(len(self.clusters)):
12           for data_index in range(len(self.clusters[cluster_index])):
13               diff = self.clusters[cluster_index][data_index, 1:5] - self.centroid
14               distance_square = np.sum(diff * diff, axis=1)
15               target_index = np.argmin(distance_square)
16
17               if cluster_index != target_index:
18                   self.finished = False
19                   target_index_list[cluster_index].append(target_index)
20                   data_index_list[cluster_index].append(data_index)
21
22       for cluster_index in range(self.K):
23           for index in range(len(target_index_list[cluster_index])):
24               target_index = target_index_list[cluster_index][index]
25               data_index = data_index_list[cluster_index][index]
26
27               self.clusters[target_index] = np.append(self.clusters[target_index],
28                                                       self.clusters[cluster_index][data_index, :])
29
30       for cluster_index in range(self.K):
31           data_index = data_index_list[cluster_index]
32           self.clusters[cluster_index] = np.delete(self.clusters[cluster_index], data_index, axis=
```

为了体现出 K 均值聚类的底层逻辑，因此代码实现中尽量不使用现成代码，而是一步步顺序实现。由于水平有限，因此上述代码可能存在实现不够精简的问题，大家如果有更好的实现，欢迎反馈。先行谢过。

在代码的 5、6两行，我们创建了两个空列表，用于记录“需要移动的数据在原簇中的索引”（data_index_list），以及“需要移动的数据应当移至的目标簇”（target_index_list）。

在第 7 ~ 9 行的初始化中，我们按 K 值将这两个列表初始化为具有 K 个列表元素的嵌套列表，这样可以节省一个用于指

示“需要移动的数据原先所在簇”的数据，只需由这两个列表的一级索引即可判断应该从哪个簇移动 / 删除数据。

第 11 ~ 15 行是计算数据点和各簇中心间距离的代码，无需赘言。需要提的是，由于对数值求平方根并不影响数值的相对大小，因此我们用于比较的实际上是“距离的平方”，而非真正的“距离”。这样在数据量大的时候也能够稍微减少一些计算量。

第 17 ~ 20 行即是判断数据是否在它应该在的簇中，如果不在就表示应当移动这个数据到另一个簇。由于对全部数据的循环遍历不一定结束，因此在这里我们不能够直接对原本的 `clusters` 进行数据的增删，而仅仅是记录下需要移动的目标簇和数据索引，在遍历结束之后再修改原始数据。

容易判断，我们需要先增，再删，否则很可能导致索引的数据张冠李戴，发生错误。第 22 ~ 32 行干的就是这个活儿，先把数据添加到需要移动到的目标簇中，然后再从原簇中删去相应的簇。并且这个删除要么应该从后往前删除；要么应该一次全选，同时删除。我们用的是后一种方法，即用一个列表作为索引，选中全部需要删除的数据。

1.3 更新

这一步也很好理解，实现起来也很简单。

简单讲就是需要根据新得到的簇的内容，重新计算各个簇相应的中心点。我们使用 `Numpy` 提供的求均值功能来实现。代码如下：

```
    # 更新各个簇的质心
    def update(self):
        for cluster_index in range(len(self.clusters)):
            self.centroid[cluster_index] = np.mean(self.clusters[cluster_index][:,1:5], axis=0)
```

到此，K 均值聚类算法的完整步骤都已经实现了。

2. 增加可用性

2.1 训练

为了方便，我们还为 `Clusters` 类定义了一个 `train` 方法，用于直接完成算法的训练过程。

该方法实际上就是封装了 `assign` 和 `update` 两个方法，没有其他的功能上的增加。

```
    def train(self):
        '''
        进行聚类训练
        '''
        while not self.finished:
            self.assign()
            self.update()
        print('训练完成！！！')
```

值得一提的是，我们用一个名为 `finished` 的布尔变量来标识何时结束训练。

在一开始， `finished` 的值是被初始化为 `False` 的。进入分配（assign）步骤时，首先就将其置为 `True` ，也就是假定训练过程已经完成。一旦在分配过程中发生了有数据需要从当前所在簇移动到另一个簇的情况，就立即将 `finished` 置为 `False` ，也就是说明训练过程还在继续，没有结束。

2.2 打印结果

```
    def printResult(self):
        '''
        打印聚类结果
        '''
        print('-'*80)
        print('*'*80)
        print('-'*80)
        print('*'*30, '聚类结果', '*'*30)
        print('-'*30,'各簇中心','-'*30)
        for i in range(self.K):
            print('第', str(i), '簇中心：', self.centroid[i])
        print('-'*80)
        print('-'*30,'各簇结果','-'*30)
        for i in range(self.K):
            print('-'*20, '第', str(i), '簇结果', '-'*20,)
            for d in self.clusters[i]:
                print(d[5])

        print('-'*80)
        print('*'*80)
        print('-'*80)
```

这个方法就纯粹是汇总一下最终结果，并且格式化打印为可读的文本。

2.3 调用 Clusters

```
print('-'*80)
K = int(input('请输入要划分的簇数（应为正整数）：'))
data = Clusters.getData()
clusters = Clusters(data, K)
clusters.train()
clusters.printResult()
```

3. 运行结果

```
--------------------------------------------------------------------------------
请输入要划分的簇数（应为正整数）：3
训练完成！！！
--------------------------------------------------------------------------------
********************************************************************************
--------------------------------------------------------------------------------
****************************** 聚类结果 ******************************
------------------------------ 各簇中心 ------------------------------
第 0 簇中心：[5.005999999999999 3.4180000000000006 1.464 0.2439999999999999]
第 1 簇中心：[6.853846153846153 3.0769230769230766 5.715384615384615 2.053846153846153]
第 2 簇中心：[5.883606557377049 2.740983606557377 4.388524590163935 1.4344262295081964]
--------------------------------------------------------------------------------
------------------------------ 各簇结果 ------------------------------
-------------------- 第 0 簇结果 --------------------
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
```

```
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
-------------------- 第 1 簇结果 --------------------
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
```

```
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
-------------------- 第 2 簇结果 --------------------
Iris-virginica
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-virginica
Iris-virginica
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
```

```
Iris-virginica
Iris-versicolor
Iris-versicolor
Iris-virginica
Iris-virginica
Iris-versicolor
Iris-virginica
Iris-virginica
Iris-virginica
Iris-versicolor
Iris-virginica
Iris-virginica
Iris-versicolor
Iris-virginica
Iris-virginica
Iris-virginica
--------------------------------------------------------------------------------
**********************************************************************************
--------------------------------------------------------------------------------

```

4. 总结

为了避免堆砌公式，因此本文以可用代码为例，逐步讲解了典型机器学习算法 K 均值聚类的实现过程。文中代码为个人实现，因此难免存在问题，还请大家不吝指出。

感谢大家一起成长。

> 示例代码：https://github.com/JustDoPython/python-100-day/tree/master/

参考资料

https://book.douban.com/subject/26708119/ https://book.douban.com/subject/33437381/
https://www.kaggle.com/uciml/iris https://blog.csdn.net/pipisorry/article/details/51822775

系列文章

PS： 公号内回复：Python，即可进入Python 新手学习交流群，一起**100天计划！**

-END-

Python 技术

关于 **Python** 都在这里