关于分类算法，我的讲解已经告一段落，从这一小节开始，我们进入聚类算法的学习。不知道你是否对前面讲解的 “什么是聚类问题” 还有印象，我在这里再简单介绍一下。

聚类算法属于无监督学习，与分类算法这种有监督学习不同的是，聚类算法事先并不需要知道数据的类别标签，而只是根据数据特征去学习，找到相似数据的特征，然后把已知的数据集划分成几个不同的类别。

比如说我们有一堆树叶，对于分类问题来说，我们已经知道了过去的每一片树叶的类别。比如这个是枫树叶，那个是橡树叶，经过学习之后拿来一片新的叶子，你看了一眼，然后说这是枫树叶。而对于聚类问题，这里一堆树叶的具体类别你是不知道的，所以你只能学习，这个叶子是圆的，那个是五角星形的；这个边缘光滑，那个边缘有锯齿…… 这样你根据自己的判定，把一箱子树叶分成了几个小堆，但是这一堆到底是什么树叶你还是不知道的。

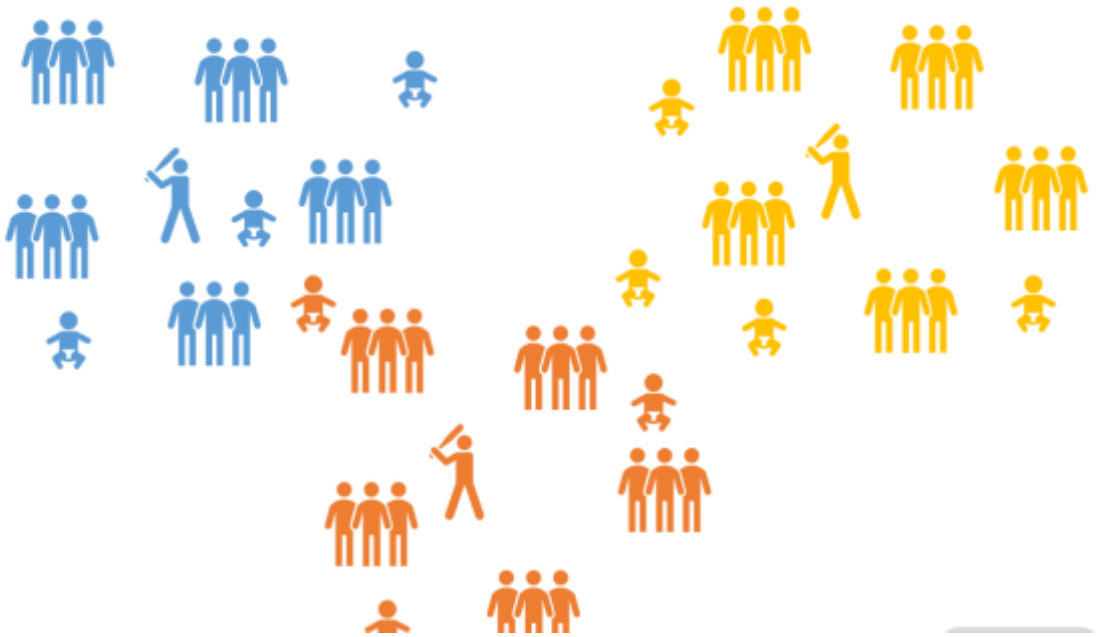
一个例子

今天我要介绍的聚类算法称为 K-means 算法。首先我还是讲一个小例子来介绍一下这个方法的思路。

假设我们在罪恶都市里，有三个区域，每个区域有一个帮派进行管理。每个帮派都有一个大佬，每个大佬都管理着一群小弟，小弟们也有不同的等级。大佬给高级小弟安排任务，高级小弟再给低级小弟安排任务，而低级小弟们负责具体实施。有些小弟可能就在自己的区域活动，管理本区域内的店铺、保障本区域的治安；有些小弟可能会负责跟其他两个帮派联络、洽谈地盘、交易等业务。

这个时候来了个国民警卫队要整治这个区域，所以他们希望能够把帮派的关系理清楚，但是有那么多人该从哪里入手呢？最好的方案当然是先把大佬抓到，然后看一下他都联系谁就一目了然了。但是国民警卫队也不知道谁是大佬，那要怎么办？

假设国民警卫队已经知道这里面有三个帮派，那么国民警卫队派人在每个区设一个点，先随便抓一个人，最开始可能抓到的只是一个边缘小弟，甚至有一些可能抓到的两个是同一个帮派。但是没关系，先假设他是大佬，看跟他联系密切的都是哪些人，然后再从这些人里找一个跟其他人联系更密切的人。就这样反复寻找，最后终于找到每个帮派的大佬，而大佬联系的那些人自然就是这个帮派的小弟了。



算法原理

上面的小故事可以看到一些 K-means 的思想。接着我们来具体介绍一下算法的原理。假设我们的数据总共有 m 条，我们计划分为 3 个类别。如果我们的数据有两个特征维度，那我们的数据就分布在一个二维平面上，如果有十个维度，就分布在一个十维的空间中。

第一轮，先随机在这个空间中选取三个点，我们称之为**中心点**，当然选取的三个点不一定是实际的数据点。接着计算所有的点到这三个点的距离，这里的距离计算仍然使用的是**欧氏距离**，每个点都选择距离最近的那个作为自己的中心点。这个时候我们就已经把数据划分成了三个组。使用每个组的数据计算出这些数据的一个均值，使用这个均值作为下一轮迭代的中心点。

后面若干轮重复上面的过程进行迭代，当达到一些条件，比如说规定的轮次或者中心点的变动很小等，就可以停止运行了。

K-means 的算法原理就已经解释完了，也是非常简洁、易于理解，但是这里面有一些问题需要解决。

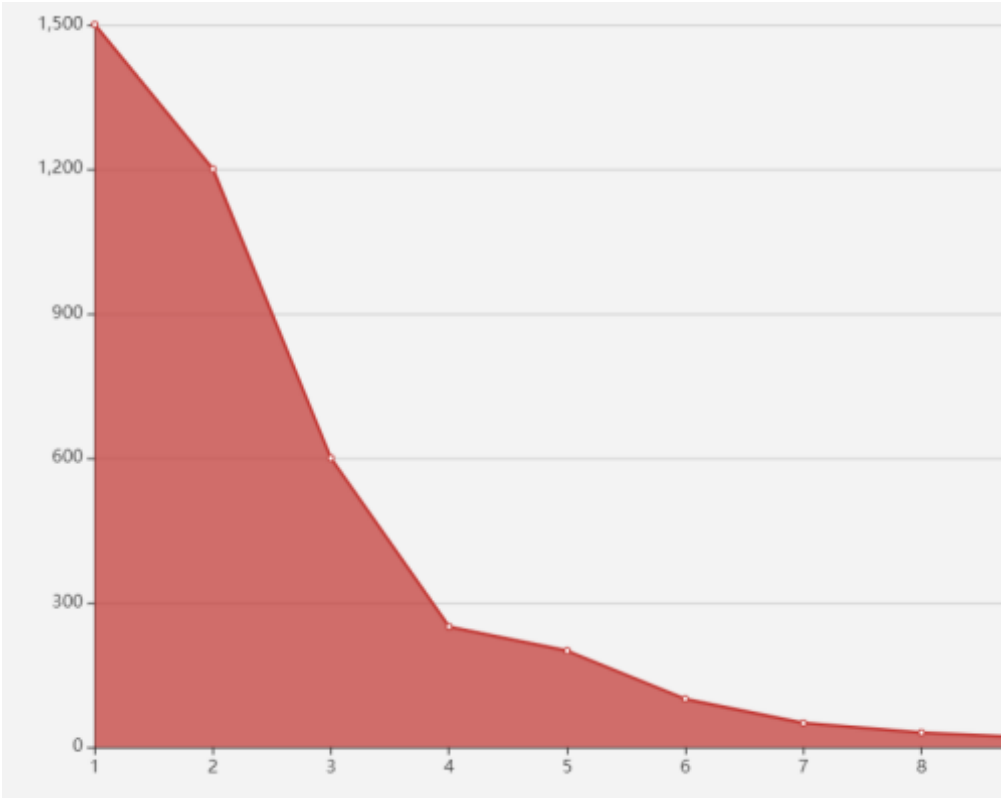
如何确定 k 值

在算法实现的过程中，我们面临的问题就是如何确定 K 值。因为在日常的情况下，我们也不知道这些数据到底会有多少个类别，或者分为多少个类别会比较好，所以在选择 K 值的时候比较困难，只能根据经验先拍一个数值。

有一个比较常用的方法，叫作**手肘法**。就是去循环尝试 K 值，计算在不同的 K 值情况下，所有数据的损失，即用每一个数据点到中心点的距离之和计算平均距离。可以想到，当 K=1 的时候，这个距离和肯定是最大的；当 K=m 的时候，每个点也是自己的中心点，这个时候全局的距离和是 0，平均距离也是 0，当然我们不可能设置成 K=m。

而在逐渐加大 K 的过程中，会有一个点，使这个平均距离发生急剧的变化，如果把这个距离与 K 的关系画出来，就可以看到一个拐点，也就是我们说的手肘。

如下图，我在这里虚拟了一份数据，可以看到在 K=4 的时候就是我们的肘点，在这个肘点前平均距离下降迅速，在 4 之后平均距离下降变得缓慢。但是这个方法只能适用 K 值不那么大的情况，如果 K 值较大，如几千几万，那迭代的次数就太多了，当然你也可以选择一个比较大的学习率来加以改进。不过总体而言，需要消耗一定的时间。



要确定 K 值确实是一项比较费时费力的事情，但是也是必须要做的事情。下面我们来看看这个算法的优缺点。

算法优缺点

优点

- **简洁明了，计算复杂度低。** K-means 的原理非常容易理解，整个计算过程与数学推理也不是很困难。
- **收敛速度较快。** 通常经过几个轮次的迭代之后就可以获得还不错的效果。

缺点

- **结果不稳定。** 由于初始值随机设定，以及数据的分布情况，每次学习的结果往往会有一些差异。
- **无法解决样本不均衡的问题。** 对于类别数据量差距较大的情况无法进行判断。
- **容易收敛到局部最优解。** 在局部最优解的时候，迭代无法引起中心点的变化，迭代将结束。
- **受噪声影响较大。** 如果存在一些噪声数据，会影响均值的计算，进而引起聚类的效果偏差。

尝试动手

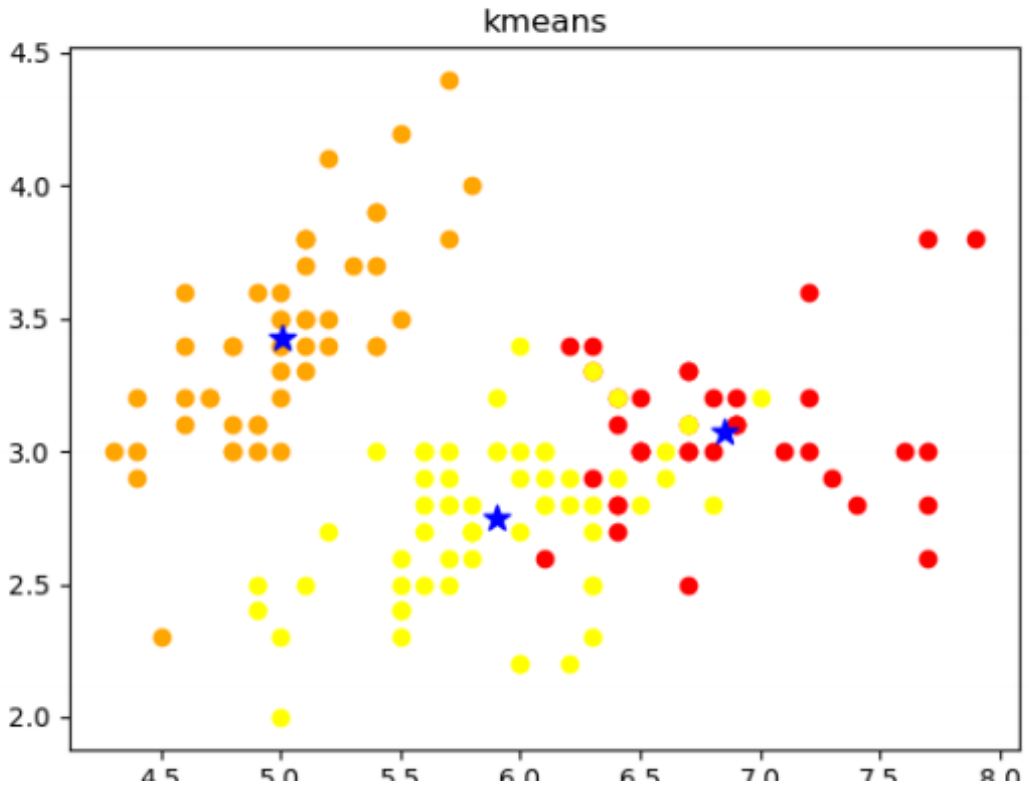
和前面一样，在对 K-means 算法有了一定了解之后，我们来动手尝试通过代码来实际感受 K-means 算法的效果。这次我们使用的仍然是鸢尾花数据集，当然，由于是聚类，我们不需要使用标签数据，只需要使用特征数据就可以了。

```
from sklearn import datasets


import numpy as np


import matplotlib.pyplot as plt


from sklearn.cluster import KMeans


""" 画出聚类后的图像


labels: 聚类后的label, 从0开始的数字


cents: 质心坐标


n_cluster: 聚类后簇的数量


color: 每一簇的颜色


"""


def draw_result(train_x, labels, cents, title):


    n_clusters = np.unique(labels).shape[0]
```

```
    color = ["red", "orange", "yellow"]

    plt.figure()

    plt.title(title)

for i in range(n_clusters):

        current_data = train_x[labels == i]

        plt.scatter(current_data[:, 0], current_data[:,1], c=color[i])

        plt.scatter(cents[i, 0], cents[i, 1], c="blue", marker="*", s=100)

return plt

if __name__ == '__main__':

    iris = datasets.load_iris()

    iris_x = iris.data

    clf = KMeans(n_clusters=3, max_iter=10,n_init=10, init="k-means++",
algorithm="full", tol=1e-4,n_jobs= -1,random_state=1)

    clf.fit(iris_x)

    print("SSE = {0}".format(clf.inertia_))

    draw_result(iris_x, clf.labels_, clf.cluster_centers_, "kmeans").show()

SSE = 78.851441426146
```

注：SSE是误差平方和，这个值越接近0说明效果越好

通过运行上面的代码，会输出下面的这幅图像，当然，我们的鸢尾花数据集的属性有四个维度，这里输出的图像我们只使用了两个维度，但是仍然可以看出通过 K-means 计算出的中心点与数据分布基本上是一致的，而且效果也还不错。



扩展内容

做完了实践，我们再来看一下 K-means 都有什么样的衍生方法。由于 K-means 也是一种非常不错的方法，所以有很多人为了改正它存在的一些问题进行了相应的研究。

K-means++

第一种是 K-means++，这种方法主要在初始选取中心点的时候进行了优化。原本第一轮是随机进行选取的，但是由于算法可能会陷入局部最优解，随机地选取可能引起结果的不稳定。K-means++ 则是从已有的数据中随机地进行多次选取 K 个中心点，每次都计算这一次选中的中心点的距离，然后取一组最大的作为初始化中心点。

mini batch K-means

第二种 mini batch 方法，主要是基于在数据量和数据维度都特别大的情况下，针对运算变得异常缓慢的问题进行的改进。我们前面提到，K-means 的收敛速度相对较快，所以前面几步的变动比较大，到了后面的步骤其实只有非常小的变动。mini batch 的方案就是在迭代时，不再使用所有的点，而是每个集合中选取一部分点进行计算，从而降低计算的复杂度。

总结

写到这里，本课时的主要内容已经告一段落。这节课我们进入了新的算法类型——聚类算法的学习。在开头我又简单介绍了一下什么是聚类算法，聚类与分类有什么样的区别，接着就讲到了本节课的主角——K-means 算法，它是一种非常简洁的基于划分的聚类算法。与前面一样，在介绍完算法的思想之后我加入了一段代码来实现快速上手，并且加入了一个画图的方法来展示聚类的效果。

在看完了这一课时的内容之后，你是否能在自己的工作中使用 K-means 来解决问题了呢？下一课时，我们将介绍另外一种聚类算法 “DBScan”，到时见。