关于 Kmeans 聚类算法，你需要知道这些东西 | 大白话

原创 数据小斑马 数据小斑马 5月27日

作者：数据小斑马 | 数据分析师

CSDN博客专家

Kmeans是我接触的第一个机器学习算法，原理简单，却很实用，只要一想到聚类，基本上没有Kmeans解决不了的问题。

本篇用大白话的方式，整理了Kmeans聚类原理，评判标准以及Sklearn实现过程，希望可以帮助你很好地入门

Kmeans聚类原理

用大白话来说，**Kmeans聚类算法分为三步**：

1、待分类的样本向量化，投射到坐标轴上，先定分几个类（假设3类），随机找3个点做为初始聚类中心，分别计算每个点到3个中心的距离，哪个最近，这个点就属于哪个类了

2、据第一步分好的类对其内部点求均值，重新做为聚类中心，再计算一遍所有点到这几个中心的距离，重新聚类，这时肯定会有一些点叛逃，没关系，就让他走！忠诚的还是会留下来的！

3、就这么一直迭代，直到再也没有点移动或者达到设定的标准了，就可以结束啦！

这里就有几个疑问，首先，怎么衡量距离？

回想一下我们学过的二维笛卡尔坐标系，如果2个点的横纵坐标差距很小，从空间上看两个点就会挨得特别近，这个其实就是聚类的核心思想，衡量2个点之间差距的就叫做欧氏距离：（(x1-x2)^

2+(y1-y2) ^ 2) ^1/2，扩大到多维，就是各维度坐标相减后求平方和再开方

其次，为什么聚类和分类都是以欧氏距离或者其它距离，而在推荐系统中余弦相似度却用的比较多？

经过深思熟虑之后，我认为主要还是业务需求不同造成的：

聚类是想将用户按特定标准分成几类，然后针对不同的类型采用不同的运营方式，因此每种类型的用户每个指标都要非常接近，比如每个月购买金额在10万以上，登录天数在20天以上，这很明显就是很有钱的忠诚用户，这个群体就是必须要小心翼翼地呵护的。

而推荐系统其实不是为了找到各方面都很接近的人，而是找到同样喜欢一些东西的人，那喜欢怎么衡量，余弦相似度会是更好的评判标准，它只看夹角，不看距离，不管一个月花10万，还是一个月花1千，只要两人买的最多的都是同一商品，那么就代表两人对这个商品的喜欢程度就是一样的。

Kmeans聚类标准

在使用Kmeans聚类前要考虑几个问题：

1、数据是否有聚类的趋势

如果是纯随机，那么不管怎么调参，聚类的效果均是不好的，聚类趋势怎么判定，用霍普金斯统计量，取任意N个点与最近向量的距离和，再取N个点与最近向量的距离和，前者除了两者的和，如果纯随机则在0.5附近，有聚类趋势的话会趋近于1或0。

实际应用中，一般会以业务经验判断，而不会真的去做这个统计量。

2、如何确定分几类？

一般有2种方法，1是经验法，分类数=样本数/2再开根号，当然这也没啥理论依据；另一种更科学的方法叫肘方法，先分1类，2类，3类，N类，分别计算不同分类下所有点到各自聚类中心的距离，可以想到肯定是分1类距离最大，分到N类距离为0（每个点都是一个类），当从1类变成2

类，距离会迅速减小，2类变成3类，3类变成4类，直到分到某个数时，发现其减少的量会变得很缓，达到一个拐点，那这个拐点就是最佳的分类数，如果画图就像一个手肘，于是美其名曰：轴方法

当然，实际应用一般会循环多个K值，根据聚类标准的评分，来决定K

3、如何评判分类质量?

不像其它监督学习可以用测试集直接进行质量评判，聚类没有样本输出，但可以根据簇内稠密度和簇外分散度来衡量，一般有2种，而这2种Sklearn中都有。

一个是轮廓系数，向量与簇内部各点距离求均值，衡量簇内部的紧凑程度，再与簇外部所有点的距离求均值，衡量簇外部的分散程度，后者减掉前者，再除了两者的最大值，结果在[-1,1]之间，如果趋近于1，那是分得相当好了，如果是负数，那啥也别说了，直接重来吧

在Sklearn中是用silhouette_score()计算所有点的平均轮廓系数，而silhouette_samples()返回每个点的轮廓系数

另一个是Calinski-Harabaz（CH），用的是簇间的协方差矩阵与簇内的协方差矩阵相除，如果前者越大，后者越小，那么分值越大，分类越好，在sklearn中et是是用metrics.calinski_harabaz_score

实际应用中，两个都可以，看你喜欢。

Sklearn实现

1、核心参数介绍

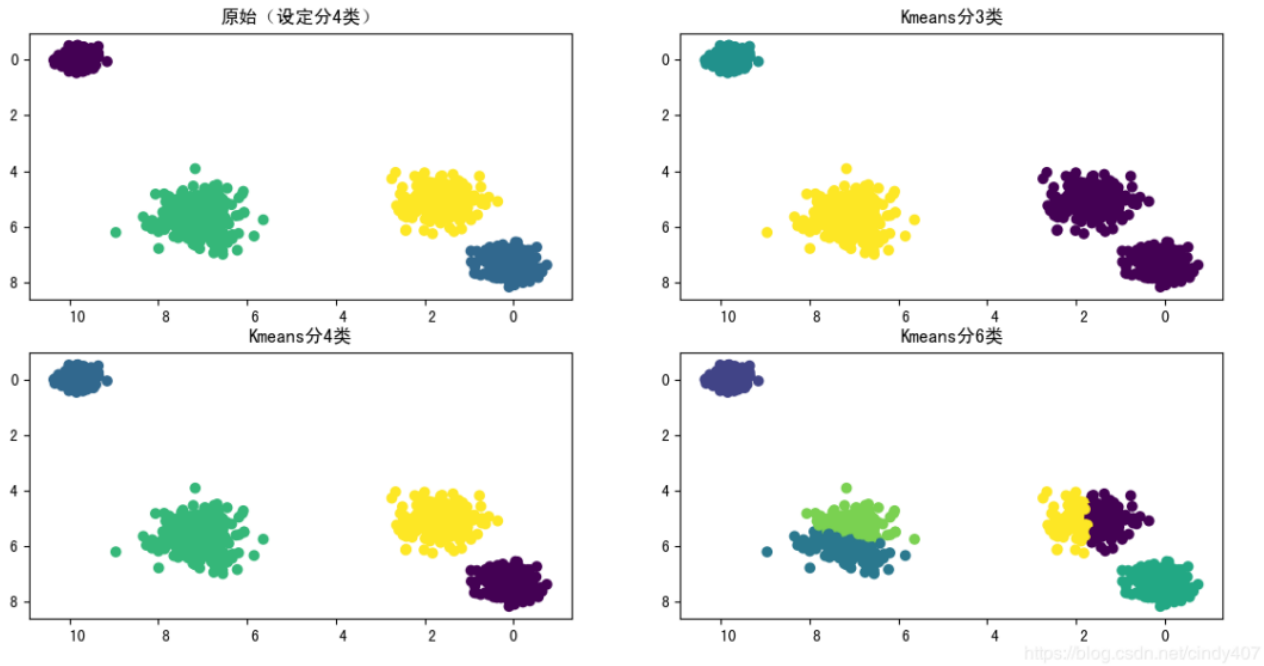
1） **n_clusters**: K值，这个值一般需要结合第3点的评判标准，找到最佳的K
2） **max_iter**： 最大的迭代次数，一般如果是凸数据集的话可以不管这个值，如果数据集不是凸的，可能很难收敛，此时可以指定最大的迭代次数让算法可以及时退出循环。
3） **n_init**：用不同的初始化质心运行算法的次数。由于K-Means是结果受初始值影响的局部最优的迭代算法，因此需要多跑几次以选择一个较好的聚类效果，默认是10，一般不需要改。如果你的k值较大，则可以适当增大这个值。

4） **init**： 初始值选择的方式，一般默认'k-means++'

2、使用数据生成器生成聚类数据，采用CH分数和散点图评判聚类结果

```
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn import datasets


x,y = make_blobs(n_samples=1000,n_features=2,centers=4,cluster_std=[0.2,0.3,0
print(x[:5])
score = []


fig = plt.figure(figsize=(20,20))
ax1 = fig.add_subplot(221)
plt.scatter(x[:,0],x[:,1],c=y)
plt.title('原始（设定分4类）')


ax2 = fig.add_subplot(222)
clf = KMeans(n_clusters=3,max_iter=1000)
pred = clf.fit_predict(x)
score.append(metrics.calinski_harabaz_score(x,pred))
plt.scatter(x[:,0],x[:,1],c=pred)
plt.title('Kmeans分3类')


ax3 = fig.add_subplot(223)
clf = KMeans(n_clusters=4,max_iter=1000)
pred = clf.fit_predict(x)
score.append(metrics.calinski_harabaz_score(x,pred))
plt.scatter(x[:,0],x[:,1],c=pred)
plt.title('Kmeans分4类')


ax4 = fig.add_subplot(224)
```

```
clf = KMeans(n_clusters=6,max_iter=1000)
pred = clf.fit_predict(x)
score.append(metrics.calinski_harabaz_score(x,pred))
plt.scatter(x[:,0],x[:,1],c=pred)
plt.title('Kmeans分6类')
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['font.serif'] = ['SimHei']  # 设置正常显示中文
plt.show()
print(score)
```




```
[8975.695887614422, 27056.054463295197, 21774.762463911247]
```

从CH分数看，分4类是最高的，这与预期设定的4类结果是一致的，从散点图上也能看出，分4类也是最好的。当特征超过2维后，我们肉眼已经无法直观判断时，CH分数可以做为很实用的替代方法

end

往 期 推 荐