Kmeans++原理与实现

原创 空字符 月来客栈 6月29日

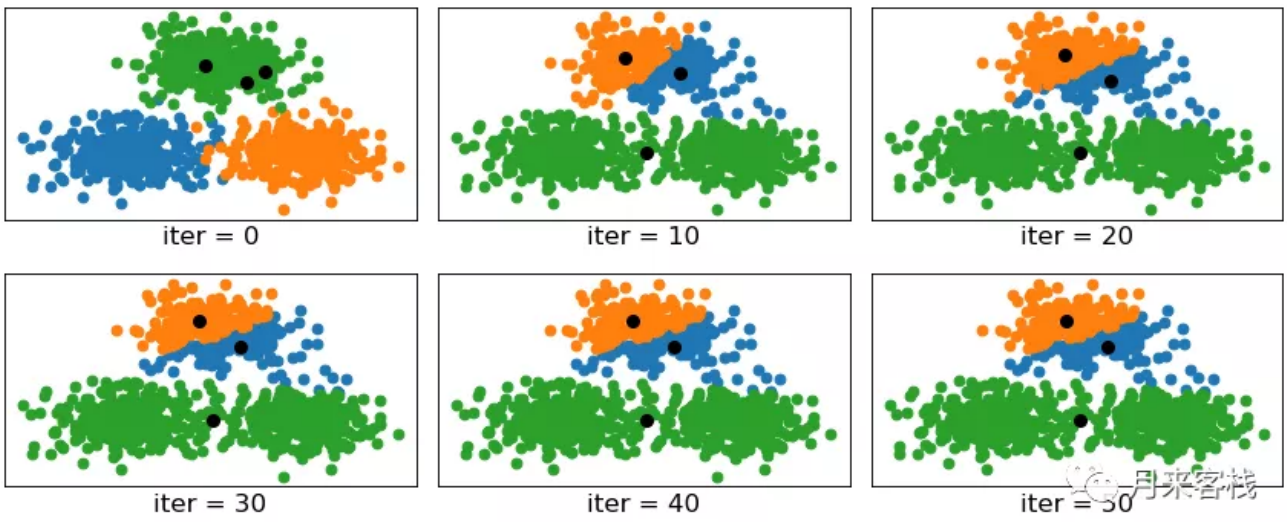
收录于话题
#《跟我一起机器学习》

51个

收藏后在电脑端打开可获得最佳阅读效果

1 引例

在上一篇文章中，笔者介绍了什么是聚类算法，并且同时还介绍了聚类算法中应用最为广泛的$Kmeans$聚类算法。从$Kmeans$聚类算法的原理可知，$Kmeans$在正式聚类之前首先需要完成的就是初始化k个簇中心。同时，也正是因为这个原因，使得$Kmeans$聚类算法存在着一个巨大的缺陷——收敛情况严重依赖于簇中心的初始化状况。试想一下，如果在初始化过程中很不巧的将k个（或大多数）簇中心都初始化了到同一个簇中，那么在这种情况下$Kmeans$聚类算法很大程度上都不会收敛到全局最小值。也就是说，当簇中心初始化的位置不得当时，聚类结果将会出现严重的错误。



如图所示的数据集仍旧是在上一篇文章中我们所用到的人造数据集，不同的是在进行聚类时我们人为的将三个簇中心初始化了到一个簇中。其中 `iter=0` 时的情况为未开始聚类前根据每个样本的真实簇标签所展示的情况，其中蓝色、绿色和橙色分别表示三个簇的分布情况，三个黑色圆点为初始化的簇中心。从上图中可以发现，$Kmeans$在进行完第10次迭代后，将最上面的一个簇分为了两个簇，将下面的两个簇划分成了一个簇。同时，当进行完第30次迭代后，可

以发现算法已经开始收敛，后续簇中心并没有发生任何变化。最终的结果仍旧是将上面一个簇分为两个，下面两个簇划分为一个。

通过上面这个例子我们知道，**初始簇中心的位置会严重影响到$Kmeans$聚类算法的最终结果**，那么我们该怎么最大可能的避免这种情况呢？此时就开始轮到$Kmeans++$算法登场了。

2 Kmeans++聚类算法

$Kmeans++$[1]，仅从名字也可以看出它就是$Kmeans$聚类算法的改进版，那它又在哪些地方对$Kmeans$进行了改进呢？一言以蔽之，$Kmeans++$算法仅仅只是在初始化簇中心的方式上做了改进，其它地方同$Kmeans$聚类算法一样。

2.1 Kmeans++原理

$Kmeans++$在初始化簇中心时的方法总结成一句话就是：**逐个选取k个簇中心，且离其它簇中心越远的样本点越有可能被选为下一个簇中心。** 其具体做法如下（其中引用英文部分论文原文）：

①从数据集$\mathcal{X}$中随机（均匀分布）选取一个样本点作为第一个初始聚类中心c_i;

> 1a. Take one center c_1, chosen uniformly at random from $\mathcal{X}$.

②接着计算每个样本与当前已有聚类中心之间的最短距离，用$D(x)$表示；然后计算每个样本点被选为下一个聚类中心的概率$P(x)$，最后选择最大概率值所对应的样本点作为下一个簇中心；

> 1b. Take a new center c_i, choosing $x \in \mathcal{X}$ with probability $P(x)$.

$$P(x) = \frac{D(x)^2}{\sum_{x \in \mathcal{X}} D(x)^2} \tag{1}$$

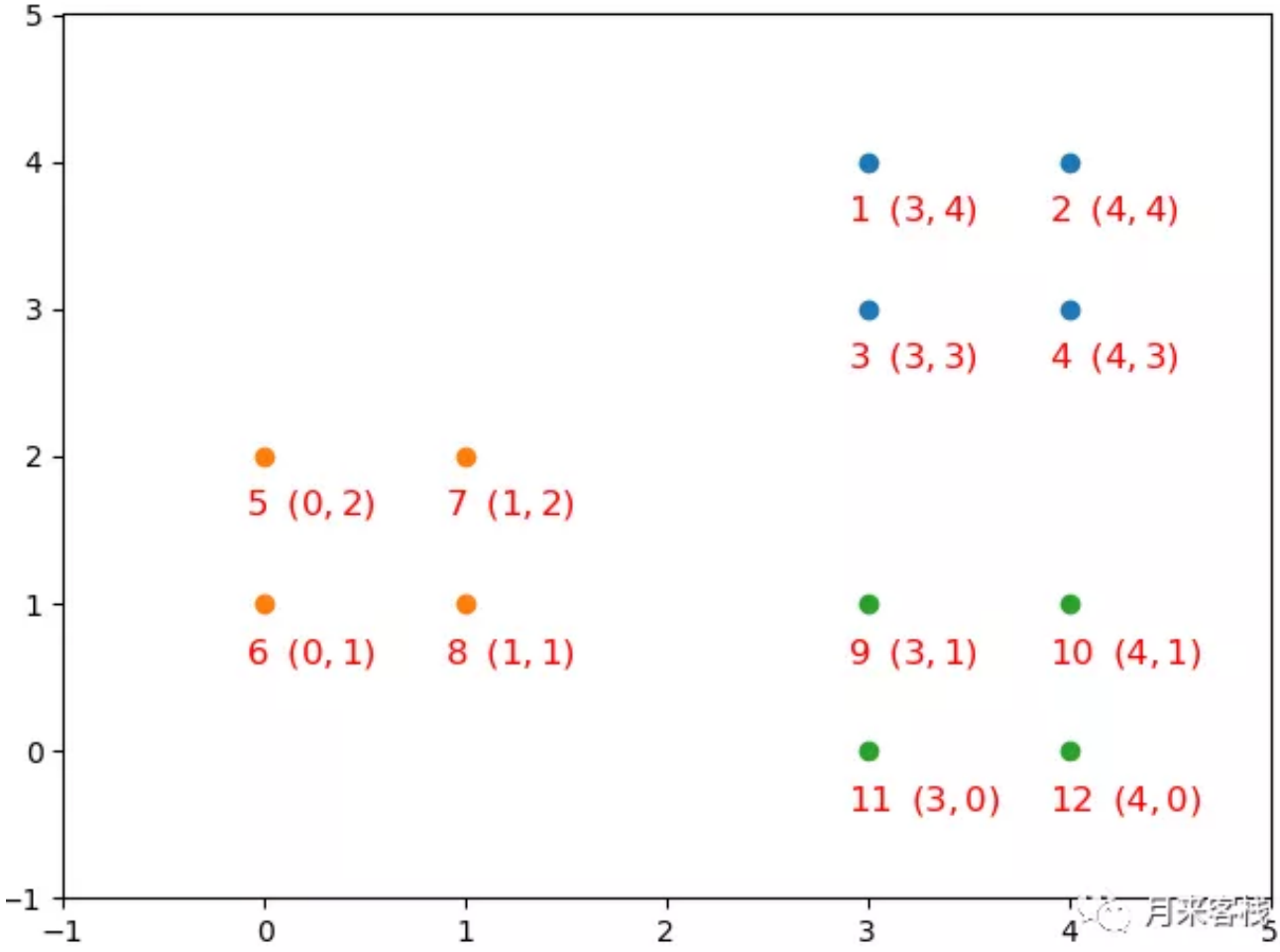
③重复第②步，直到选择出k个聚类中心；

> 1c. Repeat Step 1b. until we have taken k centers altogether.

从公式(1)也可以看成，距离现有簇中心越远的样本点，越可能被选为下一个簇中心。

2.2 计算示例

在上面的内容中，我们已经介绍了$Kmeans++$聚类算法在初始化簇中心时的具体步骤，不过仅仅只是列出公式显然不是本系列文章的风格。下面，我们就通过一个例子来实际计算一下簇中心的选择过程。



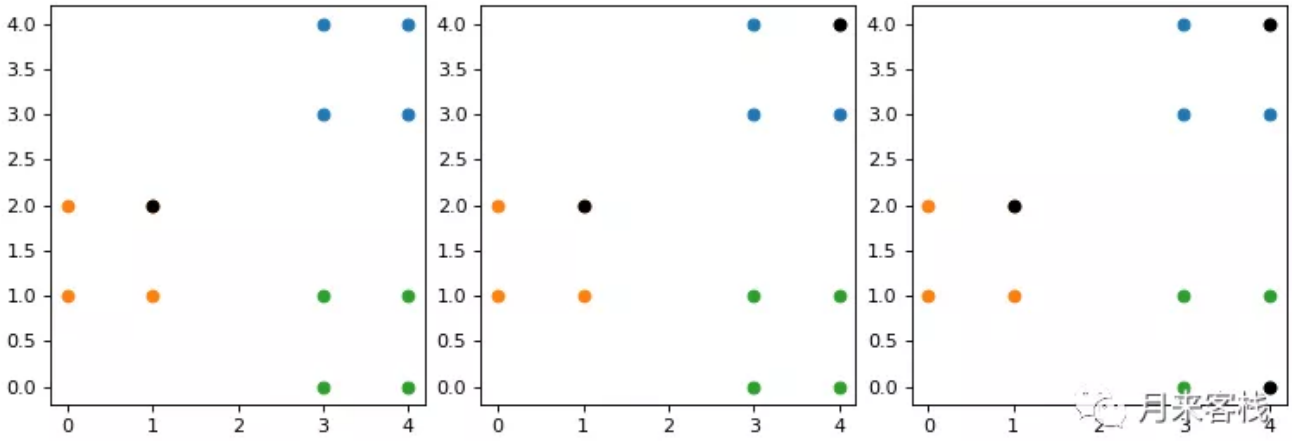
如图所示为所有的样本点，且很明显的就能看成一共包含有3个簇，这也就意味着我们需要找到3个簇中心。我们假设第一步选择的是将7号样本点$(1,2)$作为第一个初始聚类中心，那么在进行第二个簇中心的查找时，我们就需要计算所有样本点到7号样本点的距离，然后进行一个归一化。由此我们就能得到如下表格：

编号	1	2	3	4	5	6	7	8	9	10	11	12
$D(x)^2$	8	13	5	10	1	2	0	1	5	10	5	13

从表中可以看出，离7号样本点最远的是2号和12号样本点（其实从图中也可以看出），因此$Kmeans++$就会选择2号样本点为下一个聚类中心。接着，我们再次重复步骤二又能得到下面这个表格：

编号	1	2	3	4	5	6	7	8	9	10	11	12
$D(x)^2$	1	0	2	1	1	2	0	1	5	9	8	13

从表中可以看出，离2号和7号样本点最远的是12号样本点，所以下一个簇中心就会是12号样本点。进一步，我们可以得到如下可视化结果：



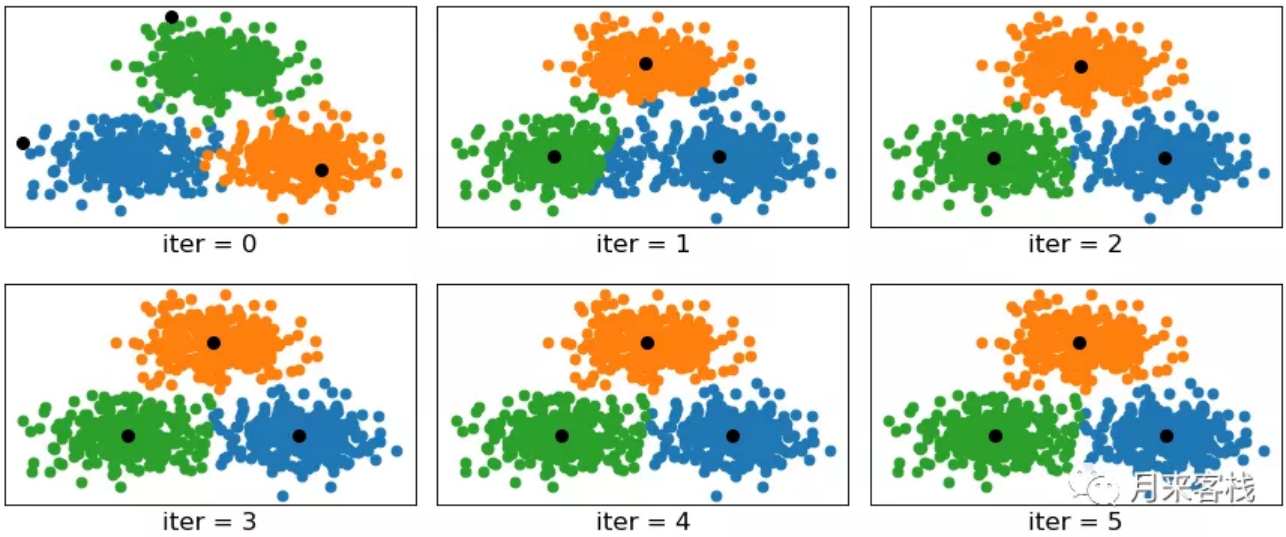
2.3 编程实现

由于$Kmeans++$算法仅仅只是在簇中心初始化方法上做了改变，因此只需要重写该函数即可。

```
def InitialCentroid(x, K):
    c0_idx = int(np.random.uniform(0, len(x)))
    centroid = x[c0_idx].reshape(1, -1)  # 选择第一个簇中心
    k = 1
    n = x.shape[0]
    while k < K:
        d2 = []
        for i in range(n):
            subs = centroid - x[i, :]
            dimension2 = np.power(subs, 2)
            dimension_s = np.sum(dimension2, axis=1)  # sum of each row
            d2.append(np.min(dimension_s))
        new_c_idx = np.argmax(d2)
        centroid = np.vstack([centroid, x[new_c_idx]])
        k += 1
    return centroid

if __name__ == '__main__':
    x, y = make_data()
    K = len(np.unique(y))
    # y_pred = kmeans(x, y)
```

```
    y_pred = kmeanspp_visual(x,y,K)
    nmi = normalized_mutual_info_score(y, y_pred)
    print("NMI by ours: ", nmi)


    model = KMeans(n_clusters=K, init='k-means++')
    model.fit(x)
    y_pred = model.predict(x)
    nmi = normalized_mutual_info_score(y, y_pred)
    print("NMI by sklearn: ", nmi)

#结果:
NMI by ours:  0.9456935014894648
NMI by sklearn:  0.9456935014894648
```

最终，我们再次将先前的人造模拟数据进行聚类便可以得到如下可视化结果：



从图示中可以看到，初始化的簇中心彼此相距都十分的远，从而不可能再发生初始簇中心在同一个簇中的情况。同时需要说明的是，sklearn中$Kmeans$聚类算法的默认中心选择方式就是通过$Kmeans++$的原理来实现的，通过参数`init=k-means++`来进行控制。

到此为止，我们就介绍完了$Kmeans++$聚类算法的的主要思想与具体实现。

3 总结

在本篇文章中，笔者首先介绍了$Kmeans$聚类算法在初始化簇中心上的弊端；然后介绍了$Kmeans++$这种聚类算法对初始化簇中心的改进，当然改进方法也不止一种；最后还介绍了如何通过python来编码实现初始化方法，同时还对聚类结果进行了可视化。本次内容就到此结束，感谢阅读！

若有任何疑问与见解，请发邮件至moon-hotel@hotmail.com并附上文章链接，青山不改，绿水长流，月来客栈见！

引用

[1]Kmeans++ http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf

[2]示例代码：https://github.com/moon-hotel/MachineLearningWithMe

近期文章

[1]Kmeans聚类算法

[2]Kmeans聚类算法求解与实现

收录于话题 #《跟我一起机器学习》 51个

上一篇 下一篇