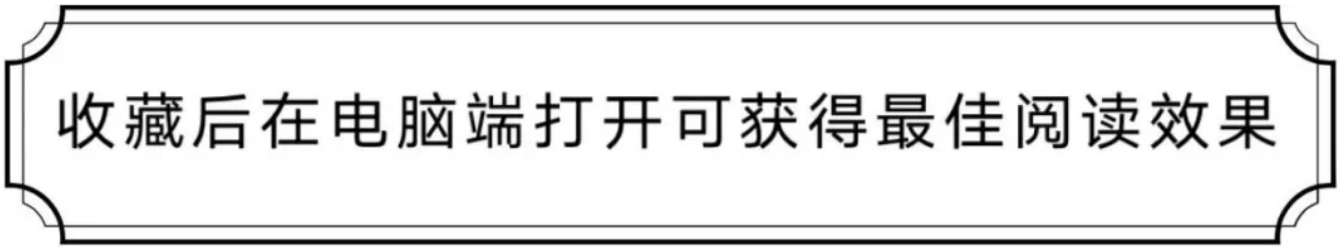
Kmeans聚类求解与实现

原创 空字符 月来客栈 6月24日

收录于话题
#《跟我一起机器学习》

51个



在上一篇文章中，笔者介绍了$Kmeans$聚类算法的主要思想与原理，并且还得到了其对应的目标函数。在接下来的这篇文章中笔者就开始介绍$Kmeans$聚类算法的求解过程，以及其对应的代码实现。

1 目标函数求解

由上一篇文章的内容可知，$Kmeans$聚类算法的目标函数如下所示：

$$P(U,Z)=\sum_{p=1}^{k}\sum_{i=1}^{n}u_{ip}\sum_{j=1}^{m}(x_{ij}-z_{pj})^2 \tag{1}$$

服从于约束条件：

$$\sum_{p=1}^{k}u_{ip}=1 \tag{2}$$

同SVM一样，对于目标函数(1)的求解我们依旧是借助拉格朗日乘数法进行，点击拉格朗日乘数法即可回顾相应内容。由目标函数(1)可知，我们一共需要求解的未知参数包括两个：簇中心矩阵Z和簇分配矩阵U。

1.1 求解簇中心矩阵

针对于目标函数(1)，关于变量z_{pj}求导可得：

$$\frac{\partial P(U,Z)}{\partial z_{pj}}=-2\sum_{i=1}^{n}u_{ip}(x_{ij}-z_{pj}) \tag{3}$$

进一步，令式子(3)为0有：

$$\begin{aligned} &\sum_{i=1}^{n} u_{ip}(x_{ij}-z_{pj})=0 \\ \Longrightarrow\ &\sum_{i=1}^{n} u_{ip}x_{ij}=\sum_{i=1}^{n} u_{ip}z_{pj} \\ \Longrightarrow\ &z_{pj}=\frac{\sum\limits_{i=1}^{n} u_{ip}x_{ij}}{\sum\limits_{i=1}^{n} u_{ip}} \end{aligned} \tag{4}$$

由此，我们便得到了簇中心的计算公式(4)。这个公式什么含义呢？其实就是每个簇中样本点对应维度的平均值。例如某个簇中有三个样本点$[1,2],[2,3],[4,6]$，则其簇中心为$\frac{1}{3}[1+2+4,2+3+6]$。

1.2 求解簇分配矩阵

在求解得到簇中心矩阵Z后，我们该怎么求解分配矩阵呢？其实根本不用求，比较即可。我们在前面介绍$Kmeans$聚类的思想时说过，聚类的本质可以看成是不同样本间相似度比较的一个过程，把相似度较高的样本放到一个簇，而把相似度较低的样本点放到不同的簇中。因此，对于每个样本点来说，我们只需要分别计算其与K个簇中心的距离（相似度），然后将其划分到与之相似度最高（距离最近）的簇中即可。也就是说求解分配矩阵其实就是一个比较的过程，通过公式(5)即可完成：

$$u_{ip}=\begin{cases} 1, & \sum\limits_{j=1}^{m}(x_{ij}-z_{pj})^2 \le \sum\limits_{j=1}^{m}(x_{ij}-z_{tj})^2, \text{for } 1\le t\le k \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

公式(5)的意思就是，计算每个样本点到所有簇中心的距离，然后将其划分到离它最近的簇中。例如某个样本点到三个簇中心的距离分别是$5,2,8$，则簇分配矩阵对应行为$[0,1,0]$。

2 聚类算法实现

经过上面的介绍，我们已经知道了$Kmeans$聚类算法两个关键未知变量的计算公式，那么接下来需要完成的应该就是对其进行编码实现。在上一篇文章中我们介绍到，聚类算法的步骤主要分为如下五个步骤：

①首先随机选择K个样本点作为K个簇的初始簇中心；

②然后计算每个样本点与这个K个簇中心的相似度大小，并将该样本点划分到与之相似度最大的簇中心所对应的簇中；

③根据现有的簇中样本，重新计算每个簇的簇中心；

④循环迭代步骤②③，直到目标函数收敛，即簇中心不再发生变化。

其中步骤④为循环过程，而关键在于前三步。接下来，我们就开始分别对其进行实现。

2.1 随机初始化簇中心

$Kmeans$聚类算法的簇中心是同时随机初始化k个簇中心，因此我们可以借助python中的 `random.sample` 来实现。

```
def InitCentroids(X, K):
    n = np.size(X, 0)
    rands_index = np.array(random.sample(range(1, n), K))
    centriod = X[rands_index, :]
    return centriod
```

其中X, K分别表示聚类数据集和簇中心的个数。

2.2 簇分配矩阵的实现

对于簇分配矩阵的实现，根据公式(5)可知，只需要遍历每个样本点然后计算其到每个簇中心的聚类，选择较近的即可：

```
def findClostestCentroids(X, centroid):
    idx = np.zeros((np.size(X, 0)), dtype=int)
    n = X.shape[0]  # n 表示样本个数
    for i in range(n):# 遍历每一个样本点
        subs = centroid - X[i, :]
        dimension2 = np.power(subs, 2)
        dimension_s = np.sum(dimension2, axis=1)# 得到每个点到k个簇的距离
        dimension_s = np.nan_to_num(dimension_s)
        idx[i] = np.where(dimension_s == dimension_s.min())[0][0]
        # 选择最小距离所对应的簇编号
    return idx
```

需要注意的是，我们在实际的编码过程中其实并不需要返回这么一个形状为$n \times k$的分配矩阵U。只需要将每个簇进行一个类别编号，然后对每个样本点赋予一个对应的编号即可。因此，上述代码中返回的 `idx` 就是每个样本点距离其最近簇的簇编号。例如 `idx=[0,1,2]` 就表示这四个样本点分别属于第 `0` 个簇、第 `1` 个簇和第 `2` 个簇。

2.3 簇中心矩阵的实现

对于簇中心矩阵的计算，根据公式(4)可知，只需要遍历k个簇，然后分别计算每个簇中所有样本点的平均中心即可：

```
def computeCentroids(X, idx, K):
    n, m = X.shape
    centriod = np.zeros((K, m), dtype=float)
    for k in range(K):
        index = np.where(idx == k)[0]  # 一个簇一个簇的分开来计算
        temp = X[index, :]  # ? by m # 每次先取出一个簇中的所有样本
        s = np.sum(temp, axis=0)
        centriod[k, :] = s / np.size(index)
    return centriod
```

2.4 聚类实现

在分别完成上述三个步骤的编码后，我们就可以将其结合在一起完成整个聚类的过程：

```
def kmeans(X, K, max_iter=200):
    centroids = InitCentroids(X, K)
    idx = None
    for i in range(max_iter):
        idx = findClostestCentroids(X, centroids)
        centroids = computeCentroids(X, idx, K)
    return idx


if __name__ == '__main__':
    x, y = load_data()
    K = len(np.unique(y))
    y_pred = kmeans(x, K)
    nmi = normalized_mutual_info_score(y, y_pred)
    print("NMI by ours: ", nmi)

    model = KMeans(n_clusters=K)
    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:  0.7581756800057784
NMI by sklearn:  0.7581756800057784
```

其中 `nmi` 为一种聚类评价指标，我们在后面的文章再进行介绍。同时，为了方便使用，笔者自己也根据sklearn的接口风格将上诉代码进行了重写，具体可以参见示例代码。

3 总结

在这篇文章中，笔者首先介绍了聚类算法未知参数的求解过程，分别得到了其各自的迭代计算公式；接着介绍了如何动手自己实现$Kmeans$聚类算法。到此，关于$Kmeans$聚类算法的所有内容就基本结束了。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] 示例代码 ： https://github.com/moon-hotel/MachineLearningWithMe

近期文章

[1]Kmeans聚类算法

[2]原来这就是支持向量机

[3]朴素贝叶斯算法

[4]K最近邻算法