【华子】机器学习022-使用均值漂移聚类算法构建模型 -

(本文所使用的Python库和版本号: Python 3.5, Numpy 1.14, scikit-learn 0.19, matplotlib 2.2)

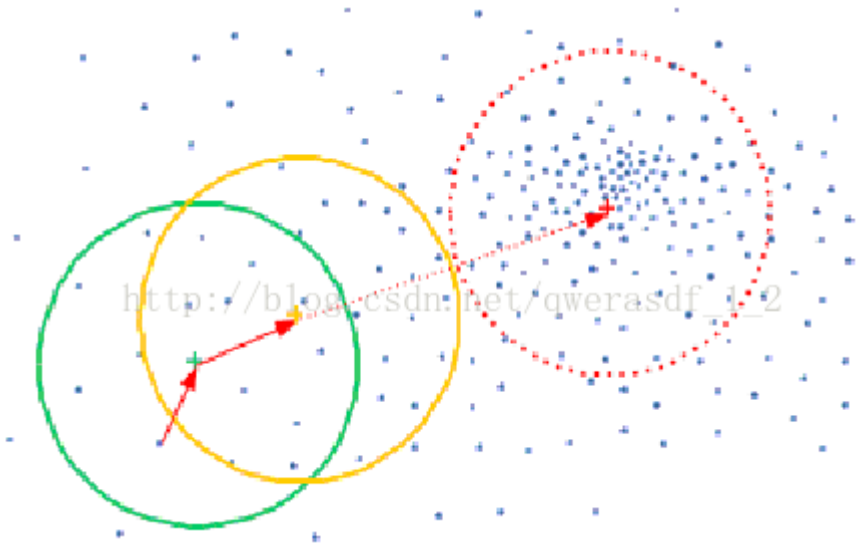
无监督学习算法有很多种，前面已经讲解过了K-means聚类算法，并用该算法对图片进行**矢量量化压缩**。下面我们来学习第二种无监督学习算法----**均值漂移算法**。

1. 均值漂移算法简介

均值漂移算法是一种基于密度梯度上升的非参数方法，它经常被应用在图像识别中的目标跟踪，数据聚类，分类等场景。

其核心思想是：首先随便选择一个中心点，然后计算该中心点一定范围之内所有点到中心点的距离向量的平均值，计算该平均值得到一个偏移均值，然后将中心点移动到偏移均值位置，通过这种不断重复的移动，可以使中心点逐步逼近到最佳位置。这种思想类似于梯度下降方法，通过不断的往梯度下降的方向移动，可以到达梯度上的局部最优解或全局最优解。

如下是漂移均值算法的思想呈现，首先随机选择一个中心点（绿色点），然后计算该点一定范围内所有点到这个点的距离均值，然后将该中心点移动距离均值，到黄色点处，同理，再计算该黄色点一定范围内的所有点到黄点的距离均值，经过多次计算均值--移动中心点等方式，可以使得中心点逐步逼近最佳中心点位置，即图中红色点处。



1.1 均值漂移算法的基础公式

从上面核心思想可以看出，均值漂移的过程就是不断的重复计算距离均值，移动中心点的过程，故而计算偏移均值和移动距离便是非常关键的两个步骤，如下为计算偏移均值的基础公式。

$$\mathrm{M}(\mathrm{x}) = \frac{1}{k} \sum_{x_i \in S_h} (x - x_i)$$

其中Sh:以x为中心点，半径为h的高维球区域； k:包含在Sh范围内点的个数； xi:包含在Sh范围内的点

第二个步骤是计算移动一定距离之后的中心点位置，其计算公式为：

$$\mathrm{x}^{t+1} = M^{\mathrm{t}} + x^t$$

其中，Mt为t状态下求得的偏移均值; xt为t状态下的中心

很显然，移动之后的中心点位置是移动前位置加上偏移均值。

1.3 均值漂移算法的运算步骤

均值漂移算法的应用非常广泛，比如在聚类，图像分割，目标跟踪等领域，其运算步骤往往包含有如下几个步骤：

1，在数据点中随机选择一个点作为初始中心点。

2，找出离该中心点距离在带宽之内的所有点，记做集合M，认为这些点属于簇C.

3，计算从中心点开始到集合M中每个元素的向量，将这些向量相加，得到偏移向量。

4，将该中心点沿着偏移的方向移动，移动距离就是该偏移向量的模。

5，重复上述步骤2,3,4，直到偏移向量的大小满足设定的阈值要求，记住此时的中心点。

6，重复上述1,2,3，4，5直到所有的点都被归类。

7，分类：根据每个类，对每个点的访问频率，取访问频率最大的那个类，作为当前点集的所属类。

1.4 均值漂移算法的优势

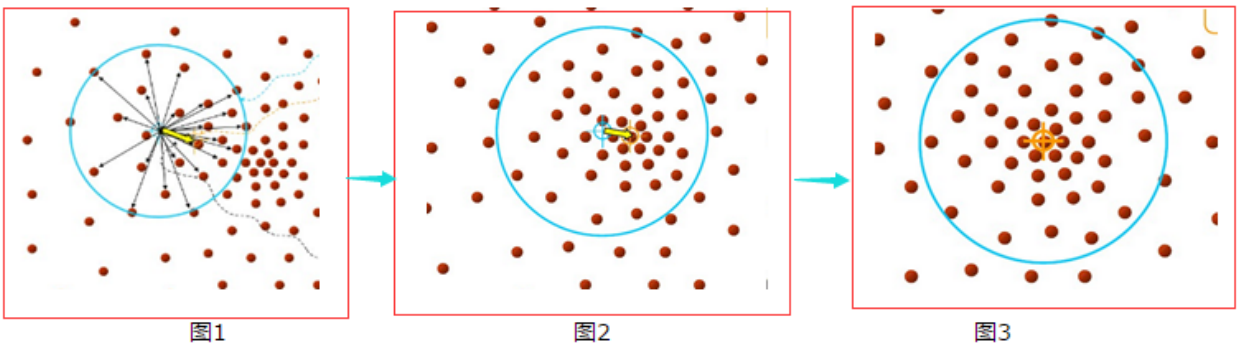
均值漂移算法用于集群数据点时，把数据点的分布看成是概率密度函数，希望在特征空间中根据函数分布特征找出数据点的**模式**，这些模式就对应于一群局部最密集分布的点。

虽然我们前面讲解了K-means算法，但K-means算法在实际应用时，需要知道我们要把数据划分为几个类别，如果类别数量出错，则往往难以得到令人满意的分类结果，而要划分的类别往往很难事先确定。这就是K-means算法的应用难点。

而均值漂移算法却不需要事先知道要集群的数量，这种算法可以在我们不知道要寻找多少集群的情况下自动划分最合适的族群，这就是均值漂移算法的一个很明显优势。

以上部分内容来源于博客文章，在此表示感谢。

1.5 MeanShift算法函数



由上图可以很容易看到，Mean-shift 算法的核心思想就是不断的寻找新的圆心坐标，直到密度最大的区域。

Mean-shift 算法函数

a）核心函数：sklearn.cluster.MeanShift(核函数：RBF核函数)

由上图可知，圆心(或种子)的确定和半径(或带宽)的选择，是影响算法效率的两个主要因素。所以在sklearn.cluster.MeanShift中重点说明了这两个参数的设定问题。

b）主要参数

bandwidth：半径(或带宽)，float型。如果没有给出，则使用sklearn.cluster.estimate_bandwidth计算出半径(带宽).（可选）

seeds :圆心（或种子），数组类型，即初始化的圆心。（可选）

bin_seeding：布尔值。如果为真，初始内核位置不是所有点的位置，而是点的离散版本的位置，其中点被分类到其粗糙度对应于带宽的网格上。将此选项设置为True将加速算法，因为较少的种子将被初始化。默认值：False.如果种子参数(seeds)不为None则忽略。

c）主要属性

cluster_centers_ : 数组类型。计算出的聚类中心的坐标。

labels_ :数组类型。每个数据点的分类标签。

2. 构建均值漂移模型来聚类数据

本文所使用的数据集和读取数据集的方式与上一篇文章**[机器学习020-使用K-means算法对数据进行聚类分析]**一模一样，故而此处省略。

下面是构建MeanShift对象的代码，使用MeanShift之前，我们需要评估**带宽**，带宽就是上面所讲到的距离中心点的一定距离，我们要把所有包含在这个距离之内的点都放入一个集合M中，用于计算偏移向量。

```
# 2.构建MeanShift对象，但需要评估带宽
from sklearn.cluster import MeanShift, estimate_bandwidth
#quantile分位数:浮点数，默认0.3

#在[0,1]之间，0.5意味着使用所有成对距离的中值
```

```
# n_samples:使用的样本数，不指定，就使用所有的样本
bandwidth=estimate_bandwidth(dataset_X,quantile=0.1,
                             n_samples=len(dataset_X))
print('预估带宽：',bandwidth)
# 指定质心：seeds=[[5,2]]，需要自行调整带宽
meanshift=MeanShift(bandwidth=bandwidth,bin_seeding=True) # 构建对象
meanshift.fit(dataset_X) # 并用MeanShift对象来训练该数据集

centroids=meanshift.cluster_centers_ # 质心的坐标，对应于feature0，feature1
print(centroids) # 可以看出有4行，即4个质心
labels=meanshift.labels_  # 数据集中每个数据点对应的label
# print(labels)

cluster_num=len(np.unique(labels)) # label的个数，即自动划分的族群的个数
print('cluster num: {}'.format(cluster_num))
```

----------**输**--------**出**----------

[[8.22338235 1.34779412] [4.10104478 -0.81164179] [1.18820896 2.10716418] [4.995 4.99967742]] cluster num: 4

----------**完**----------

可以看出，此处我们得到了四个质心，这四个质心的坐标位置可以通过meanshift.cluster_centers *获取，而 meanshift.labels* 得到的就是原来样本数据的label，也就是我们通过均值漂移算法自己找到的label，这就是无监督学习的优势所在：虽然没有给样本数据指定label，但是该算法能自己找到其对应的label。

同样的，该怎么查看该MeanShift算法的好坏了，可以通过下面的函数直接观察数据集划分的效果。

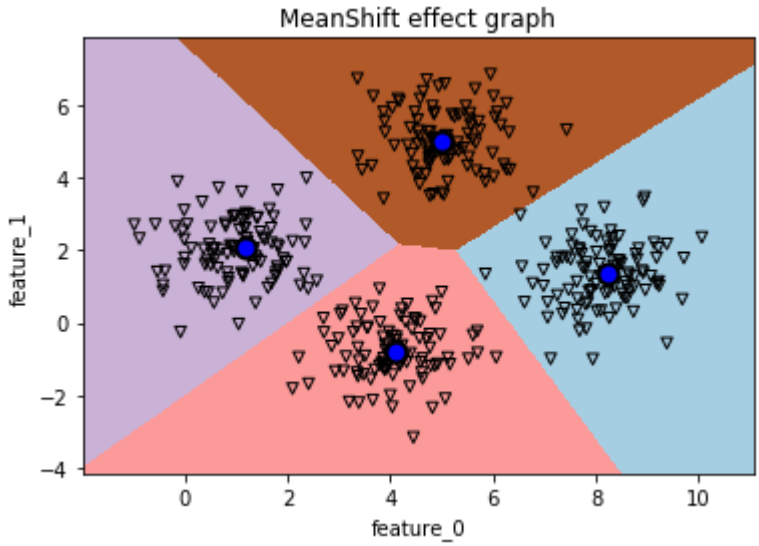
```
def visual_meanshift_effect(meanshift,dataset):
    assert dataset.shape[1]==2,'only support dataset with 2 features'
    X=dataset[:,0]
    Y=dataset[:,1]
    X_min,X_max=np.min(X)-1,np.max(X)+1
    Y_min,Y_max=np.min(Y)-1,np.max(Y)+1
    X_values,Y_values=np.meshgrid(np.arange(X_min,X_max,0.01),
                                  np.arange(Y_min,Y_max,0.01))
    # 预测网格点的标记
    predict_labels=meanshift.predict(np.c_[X_values.ravel(),Y_values.ravel()])
    predict_labels=predict_labels.reshape(X_values.shape)
    plt.figure()
    plt.imshow(predict_labels,interpolation='nearest',
               extent=(X_values.min(),X_values.max(),
                       Y_values.min(),Y_values.max()),
               cmap=plt.cm.Paired,
               aspect='auto',
               origin='lower')

    # 将数据集绘制到图表中
    plt.scatter(X,Y,marker='v',facecolors='none',edgecolors='k',s=30)

    # 将中心点绘制到图中
    centroids=meanshift.cluster_centers_
    plt.scatter(centroids[:,0],centroids[:,1],marker='o',
                s=100,linewidths=2,color='k',zorder=5,facecolors='b')
    plt.title('MeanShift effect graph')
    plt.xlim(X_min,X_max)
```

```
    plt.ylim(Y_min,Y_max)
    plt.xlabel('feature_0')
    plt.ylabel('feature_1')
    plt.show()

visual_meanshift_effect(meanshift,dataset_X)
```



########################小********结################################**

1，MeanShift的构建和训练方法和K-means的方式几乎一样，但是MeanShift可以自动计算出数据集的族群数量，而不需要人为事先指定，这使得MeanShift比K-means要好用一些。

2, 训练之后的MeanShift对象中包含有该数据集的质心坐标，数据集的各个样本对应的label信息，这些信息可以很方便的获取。

###

参考资料:

1, Python机器学习经典实例，Prateek Joshi著，陶俊杰，陈小莉译