C#下实现的K-Means优化[1]-「离群点检测」

#机器学习# #算法#

前言

在上一篇博文中,我和大家分享了「C # 下实现的多维基础K-MEANS聚类」(传送门)C#下实现的基础K-MEANS多维聚类 - xlxw - 博客园.在上篇文章中使用的是最传统的K-Means均值聚类方法,在上文中只是介绍了有一些能优化的方法但是没有具体的讲怎么去优化.所以在这篇博文中,我会和大家分享.我学到的关于我们前面说的聚类前的预处理-离群点的检测.

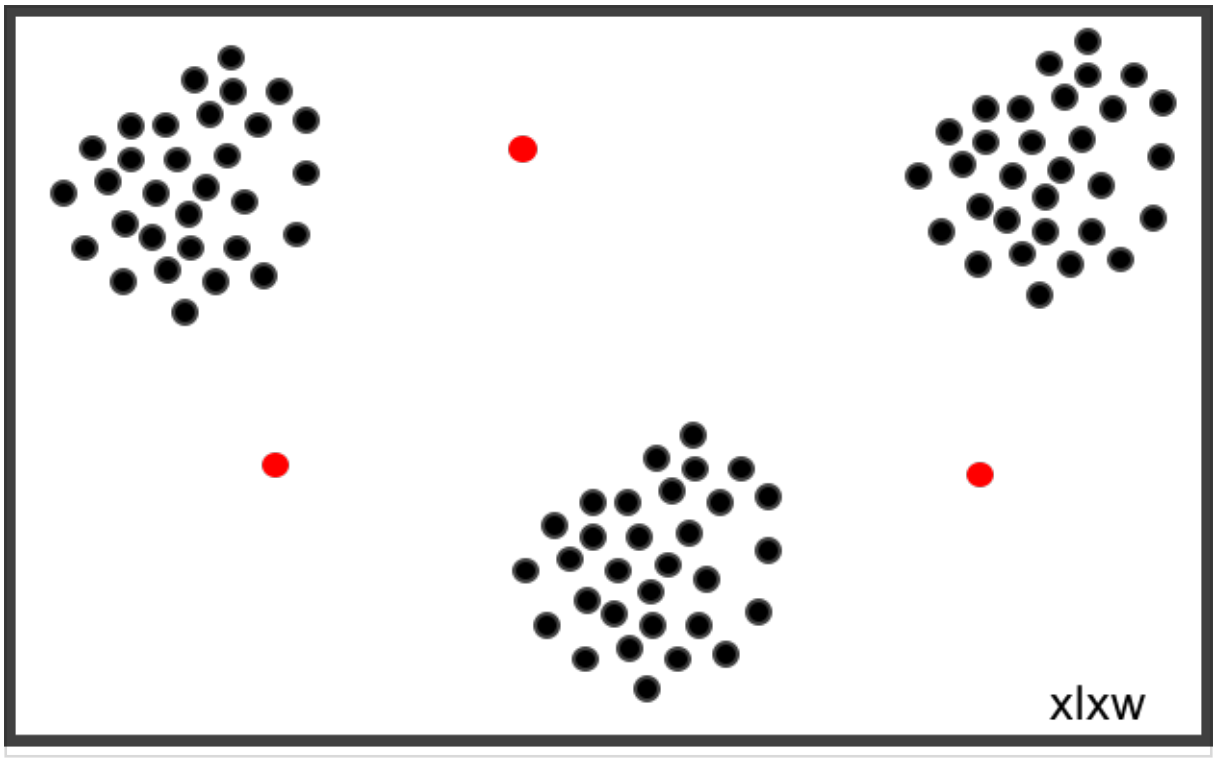
离群点的检测方法

离群点的检测是数据挖掘中很重要的部分.也是我们为了改进前一篇博文中的K-Means会因为异常点而产生较大的影响波动.而离群点的检测方法又是多种多样的.除了我会在本篇文中分享的基于密度的离群点检测方法-LOF方法外.还有许多的方法.在这里归纳一些其他的离群点检测的类别.

1. 基于统计分布的离群点检测方法
   基于先假设数据集符合如(泊松分布 / 高斯分布)的模型.将偏离统计规律严重的点筛出.
2. 基于距离的离群点检测方法
   这是最早运用于离群点检测的方法.具体原理是,先要指定参数p-数据个数 / d-阈值距离.在数据集中，若有一个点它离p个点的距离都大于d,那么就视这个点是一个离群点.
3. 基于偏差的离群点检测方法
   把偏离集中区域远的点作为离群点.
4. 基于深度的离群点检测方法
   基于随着深度(维度)的加深找到确定为离群点的数据.
5. 基于聚类的离群点检测方法
   经过改进，现在的聚类算法可以将那些不严格属于任何一个簇的数据定义为离群的点.

基于「LOF算法」的离群点检测方法

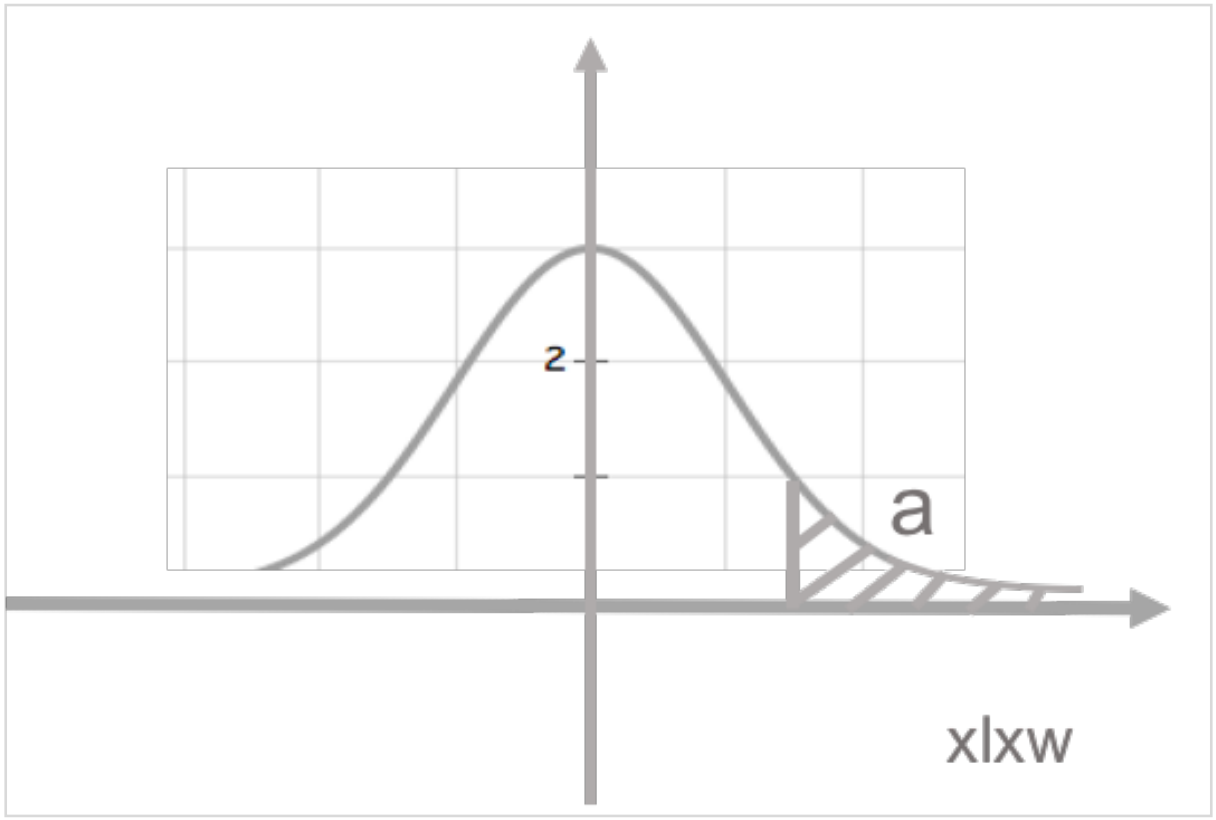
在上文中,提到过在K-Means聚类中，由于是均值聚类.采用的是欧几里德距离,故离群的数据点在样本集不是很丰富的情况下会对最终的聚类结果产生比较大的影响.但是离群值往往有比较有意思的特性值得我们去分析.所以我们一般先将所有的离群值先在聚类前剔除作单独分析.离群的点在我们视觉上可能会有很明显的感觉，比如下图中的红色点在我们看来就是离群的点



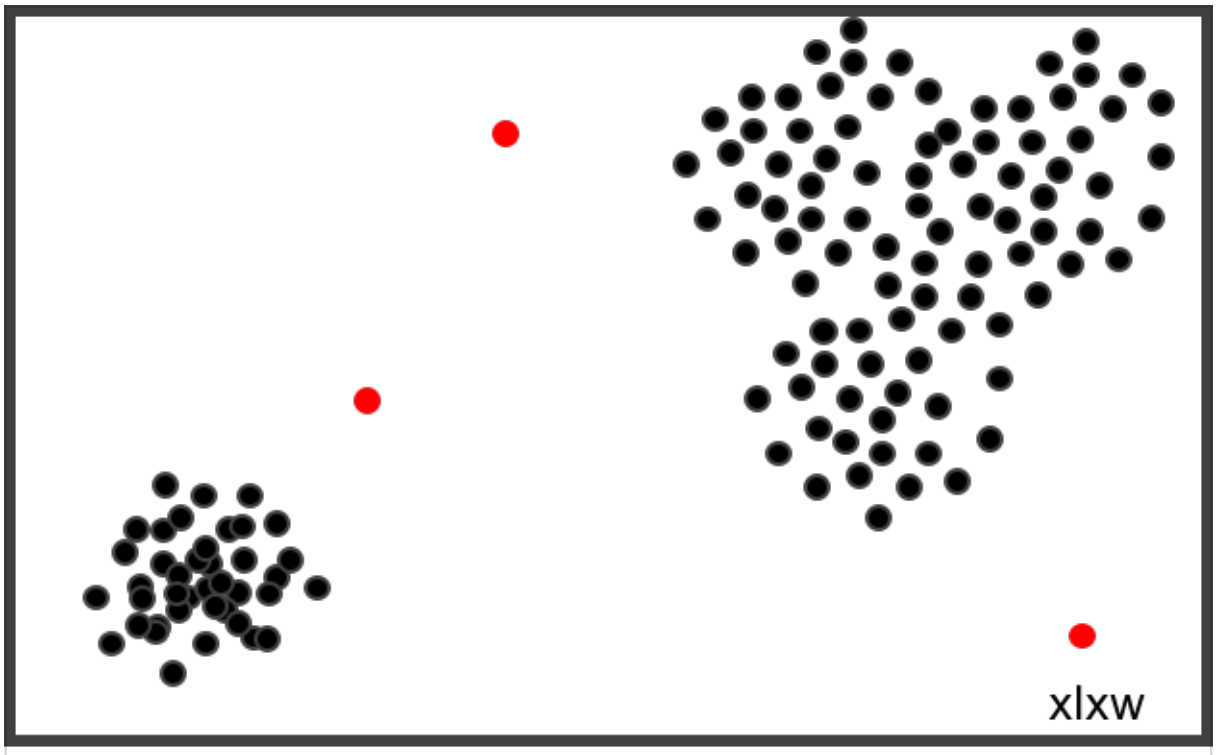
我们要做的K-Means前的预处理就是筛掉这些红色的点.那么具体的处理方法我们就要用到LOF算法了.接下来我们就来看一下LOF算法.

LOF算法

LOF算法全名是局部异常因子算法,Local Outlier Factor(LOF).异常检测的方法多是基于距离来判定的.基于距离的判别方法，较为适用于二维或高维坐标体系内异常点的判别.LOF也是基于距离来判定异常的因子的.还有的是基于分布来判定的,这里稍微的介绍一下.如下图:



上图中为分布判别方法.我们一般把右侧斜线部分就视为是异常的存在.然后回到LOF.我们要实现的是如下图这样的情况.

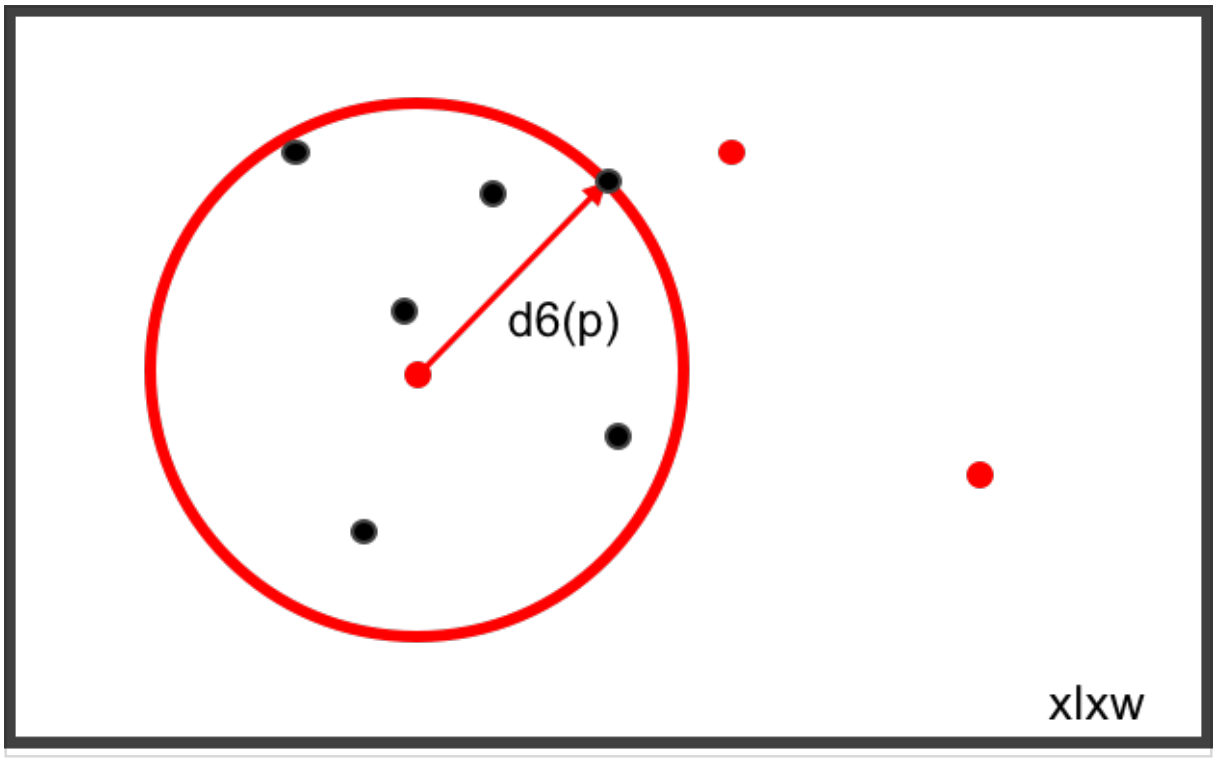


我们可以发现这里分为了两个簇，左下角的簇密度明显要高于右上角.而我们所用的LOF算法要实现的就是怎么样从密度情况不相同的这两个簇识别出对于它们两者而言的离群点

LOF算法中的定义

1. k−distance(p),简称dk(p)：点p的第k距离
2. d(p,x):两点p和x之间的距离
3. Nk(p):第k距离邻域
4. Reach-dist (p,x):可达距离
5. lrdk(p):局部可达密度
6. LOFk(p):局部离群因子

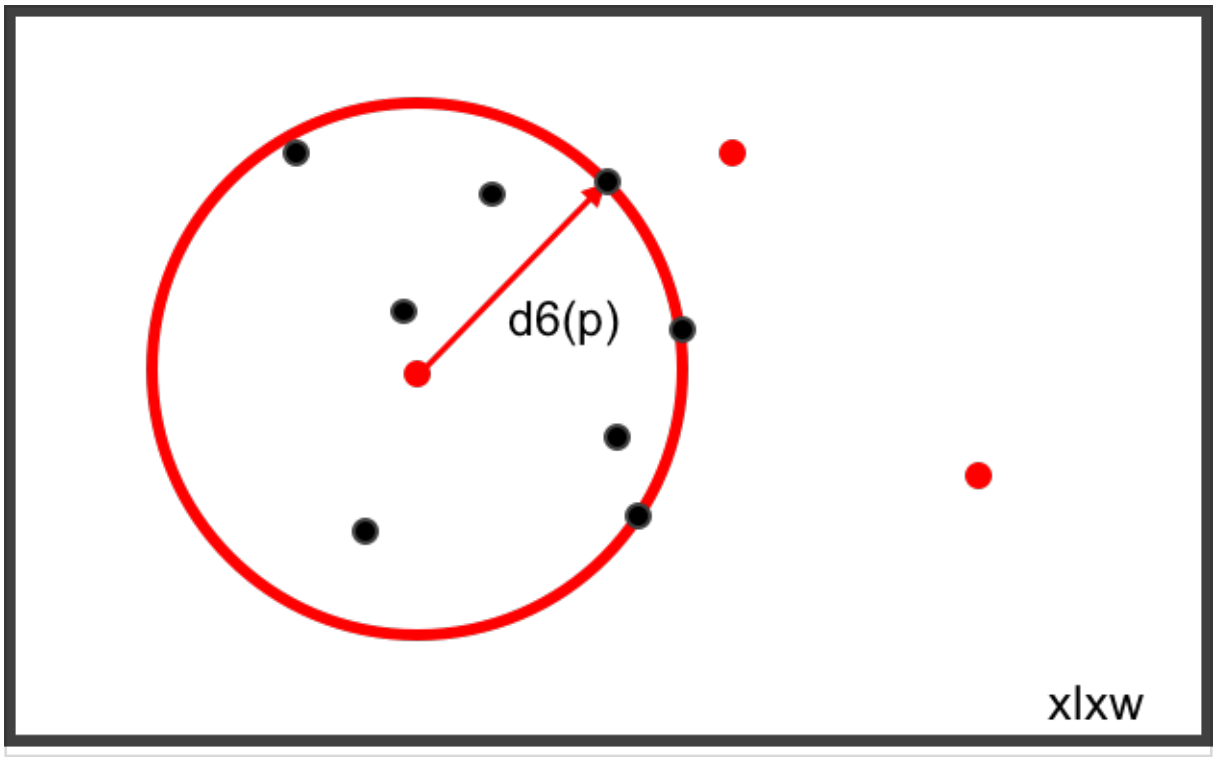
那么我们一起来看看它们之间的关系以及它们对于我们要计算的离群值有什么贡献.
1.k−distance(p),简称dk(p):点p的第k距离
如下图所示,简单的来说就是离p第k远的点



图中的黑点就是离p点(红色的圆心)第6远的点.箭头所表示的长度即为6-distance(p)即d6(p)

2.d(p,x):这个就是最简单的两点之间的欧几里得距离.即点p和x两个点之间的距离公式.

3.Nk(p):点p的第k距离邻域.即p的第k距离的以内所有点,包括第k距离由于可能会有多个点到圆心距离相同.因此p的第k邻域点的个数 |Nk(p)|≥k,取>条件如下图所示:

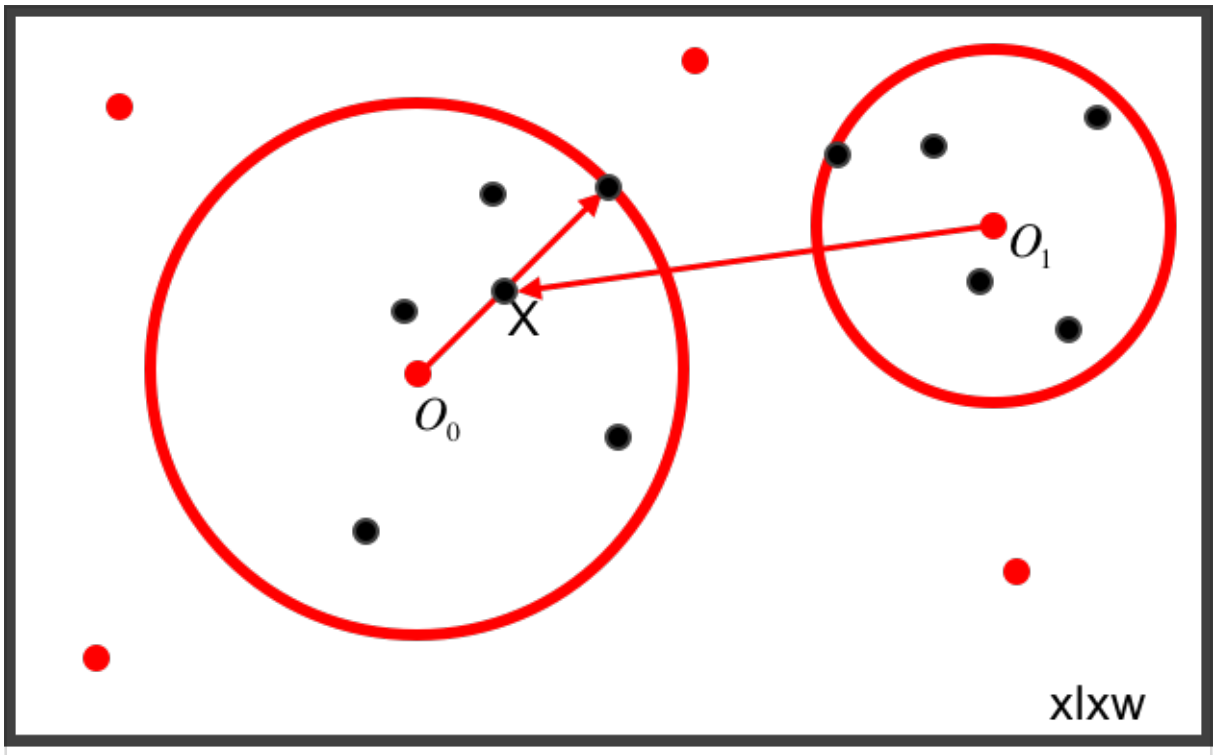


此时圆上d6(p)的点共有3个.所以可知(|Nk(p)| = 8) >6.

4.Reach-distk(p,x):可达距离

Reach-distk(p,x)=max|dk(p),d(p,x)|

我们来结合下图解释下上面的这个公式



在图中,我们用P来替代了O点.所以我们可以得到:O点到点X的第K可达距离为「O的第K距离与O / X间的距离」中较大的那一个距离.离点O最近的K个点，O到它们的可达距离被认为相等为dK(O).原因在于真是距离和dK(O)相比较小.

而我们来看上图.其中的O1到点X的第6可达距为

Reach-dist 6 = max|d6(O1),dl(O1,X)| = d6(O1).

O0到X的第6可达距离为Reach-dist 6 = max|d6(O0),d(O0,X)| = d6(O0).

5.lrdk(p):局部可达密度

$$lrdk(p)=\frac{1}{\frac{\sum_{X\in N_k(p)}Reach-dist_k(P,X)}{|N_k(p)|}}=\frac{(N_k(p))}{\sum_{X\in N_k(p)}Reach-dist_k(P,X)}$$

这个公式的意思就表示点p的第k邻域内的点到p的平均可达距离的倒数.

定义是是p的邻域点Nk(p)到p的可达距离.而不能反过来说.

lrdk(p)这个值代表的是一个密度，密度越高，我们认为这越可能是同一个簇，相反的来说.密度越低,是离群点的几率就越是高.而这样想.若我们的p和周围邻域点是同一簇的,那么计算得到的可达距离就更加可能是较小dk(p),由于局部可达密度是Reach-dist(p)的倒数,所以p和周围邻域点是同一簇这种情况的密度值较高.如果p和周围邻居点较远，那么可达距离可能都会取较大值d(p,X)，平均Reach-dist(p)越大导致局部可达密度较小,所以是离群点的可能性较大.

6.LOFk(p):局部离群因子

计算局部离群因子的公式如下所示:

$$LOF_k(X)=\frac{\sum_{P\in N_k(X)}\frac{lrd(p)}{lrd(x)}}{|N_k(X)|}$$

局部离群因子可以标示数据异常的水平，异常水平的大小表明了数据领域内数据点的孤立水平.

这个公式表示的是点X的邻域点Nk(X)的局部可达密度与点X的局部可达密度之比的平均数

如果这个比值越接近1，说明X和其邻域内点密度差不多，X可能和邻域同属一簇；如果这个比值越小于1，说明X处的密度高于其邻域点密度，X为密集点；如果这个比值越大于1，说明X的密度小于其邻域点密度，X越大越可能是异常的离群点.

LOF算法的优点以及缺点

优点

仅需要设置k的值,比较容易实现；算法由于是基于密度的，所以对X以及周围的K个点都进行了分析，降低了密度极大和密度极小对于整体数据集分析的影响.使结果比较精确.

缺点

若数据集确定了，离群点的确定和K的选取有关.当k选择不同时，离群点会发生改变.所以需要不断的调整参数来使结果符合预期.

LOF算法的实现步骤

算法实现的流程如下所示:

(1)选择合适的K

(2)获得数据集中每一个数据对象的邻域.并且通过对于欧几里德距离的计算出每个数据对象到其领域内所有数据对象点的距离.

(3)利用(1)中的结果,根据文章的上述公式,通过对所有的数据对象进行遍历,推导出所有数据对象的局部离群因子.

(4)通过定义的阈值，我们可以找到那些局部离群因子超过我们所设阈值的数据点并把它们作为离群点.

下面是C#中实现LOF算法的代码

```
private void LOFDist() {
            //xlxw.ver
            LOFk = 20;      //文章中的k
            maxLof = 3;     //文章中的阈值
```

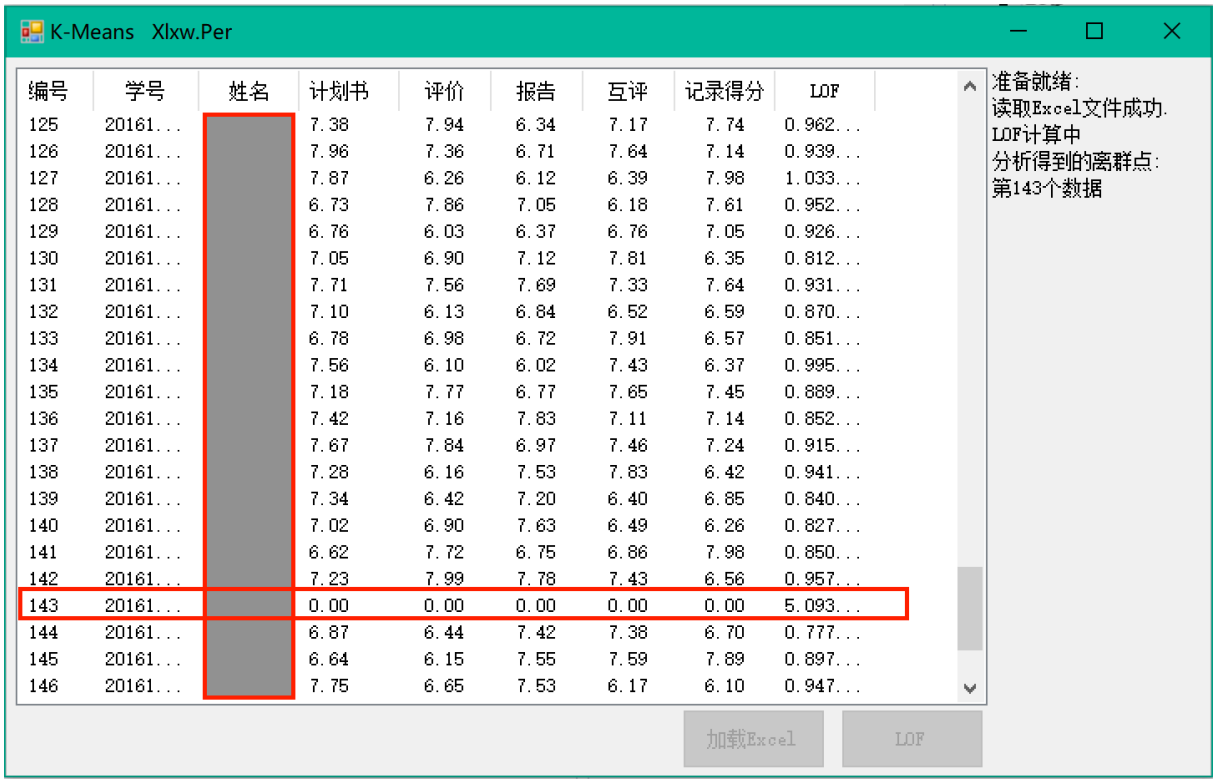
```
            NkLOF = new int[RowCount, LOFk];
            lrdk = new double[RowCount];
            LOFK = new double[RowCount];
            double tmpLofSum = 0;

            for (int i = 0; i < RowCount - 1; i++) {
                Nk(i);  //确定数据的领域Nk(p)以及计算出lrdk
            }
            richTextBox1.Text += "\nLOF计算中";
            //计算LOFK
            for (int i = 0; i < RowCount; i++) {
                tmpLofSum = 0;
                for (int j = 0; j < LOFk; j++) {
                    tmpLofSum += lrdk[NkLOF[i, j]]/ lrdk[i];
                }
                LOFK[i] = tmpLofSum / LOFk;
            }
            richTextBox1.Text += "\n分析得到的离群点:";
            //LOFK显示在ListView
            for (int i = 0; i < RowCount - 1; i++) {
                if (LOFK[i] > maxLof) richTextBox1.Text += "\n第" +
(i+1).ToString() + "个数据";
                XlsDataSh.Items[i].SubItems[8].Text = LOFK[i].ToString();
            }
        }

        private void Nk(int DataPtS) {
            double[] TmpDis = new double[RowCount];
            double tmpdis = 0,tmpNpKMax = 0,tmpSum = 0;

            //计算所有的距离
            for (int i = 0; i < RowCount - 1; i++) {
                tmpdis = 0;
                for (int j = 3; j < 7; j++) {
                    tmpdis +=
Math.Pow((System.Convert.ToDouble(XlsDataSh.Items[i].SubItems[j].Text) -
(System.Convert.ToDouble(XlsDataSh.Items[DataPtS].SubItems[j].Text))), 2);
                }
```

```
                TmpDis[i] = Math.Pow(tmpdis, 1.0/2);
            }

            double[] copy = new double[TmpDis.Length];
            TmpDis.CopyTo(copy, 0);

            //确定数据的领域Nk(p)
            for (int i = 0; i < LOFk; i++) {
                tmpNpKMax = 0;
                for (int j = 0; (j < RowCount - 1); j++) {
                    if (copy[j] > tmpNpKMax) {
                        tmpNpKMax = copy[j];
                        NkLOF[DataPtS, i] = j;
                    }
                }
                copy[NkLOF[DataPtS, i]] = 0;
            }

            //计算出lrdk
            for (int i = 0; i < LOFk; i++) {
                tmpSum += TmpDis[NkLOF[DataPtS, i]];
            }
            lrdk[DataPtS] = LOFk / tmpSum;


        }
    }
```

代码注释:

1. 同上一篇文章一样,这里用到了Excel中的数据集.数据从Excel的Xls文件中导出到了C#控件中的ListView里，这个控件的名字是:XlsDataSh,在程序的代码里可以看到这个控件的名字.
2. 第143个数据点是我作为监督对象设成的全0来检测LOF算法的.
3. 这里的K值(邻域)的设定是20.因为通过上一篇博文，可以看到我设定的分组数是5，所以在一个一百多人的班级我觉得20是比较合适的分组k值.
4. 阈值我设了3.由于正常情况的数据点应该在1的领域范围内.为了凸显异常程度所以我设置的阈值是3.

运行效果图



从图一中我们可以论证我们前面的结论.正常的数据点应该是在1附近的范围内的.从图二的异常点 - 第143号数据点可以看出.它的离群因子为5远大于1也大于我们设置的阈值，所以我们可以得出这一定是一个离群点.

参考

1. 「基于GPU的LOF算法加速」田畔 2014.4.22 论文
2. 「基于模糊聚类分析的数据异常知识发现方法」李建勋 2015.7硕士学位论文
3. 「一种改进的LOF异常点检测算法」周鹏、程艳云 2017.9.27论文网络出版