目录

ILOF 算法:

来源：Incremental Local Outlier Detection for Data Streams.pdf

基本介绍:

1、 主体结构是基于对 LOF 算法进行改进而来

2、 主要提高的地方是大大减少了 LOF 算法的时间复杂度以及运行时间（就最近一段时间阅读的关于提高 LOF 算法时间复杂度的方案这种算法最好）

算法创新点:

1、 提出了当 LOF 算法中有新的数据点插入的时候，只有与数据点直接相关联的部分数据的三个指标数据需要更新，从而避免了全局数据的更新，极大的减少了算法的计算量（这一部分创新点是算法的精髓所在）

2、 提出了一种在算法运行过程中删除部分数据，减少内存的方案（这部分算法效果分析下来效果较差，尽量不要借鉴）

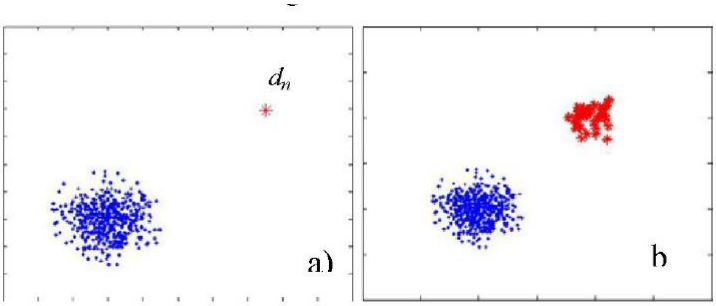
算法优缺点分析:

优点:

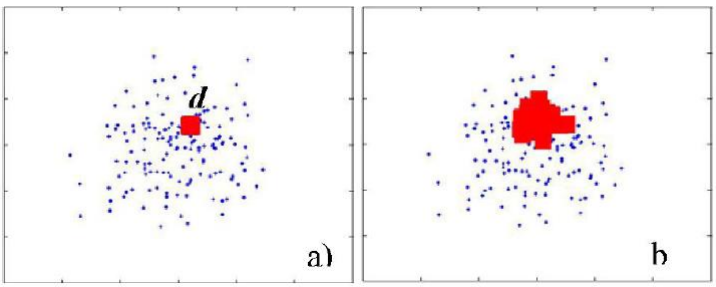
1、 在基于距离这一类异常检测算法中，很多都可以借鉴该算法，可以极大的降低算法的时间复杂度，这是目前为止差不多最佳的解决方案。

2、 在单个的异常点检测中效果极佳

3、 该算法可以解决两种特殊情况下的异常检测。如图所示:



这种异常状况是：在一定时间段内可能是局部异常，但是并不是全局异常（论文第三页有详细说明）



这种异常状况是：会覆盖本来的高密度区域（论文第三页有详细说明）

缺点分析:

1、 在实时更新的数据流中效果很差，因为需要保留所有数据，当数据搜集时常过长时，会造成内存爆炸，无法有如此大的内存来存储数据。

2、 在连续的异常序列中检测效果很差，特别是异常序列长度较大的时候

3、 直接运用到高维数据中效果也不佳

MILOF 算法

来源：Fast Memory Efficient Local Outlier Detection in Data Streams (Extended Abstract).pdf

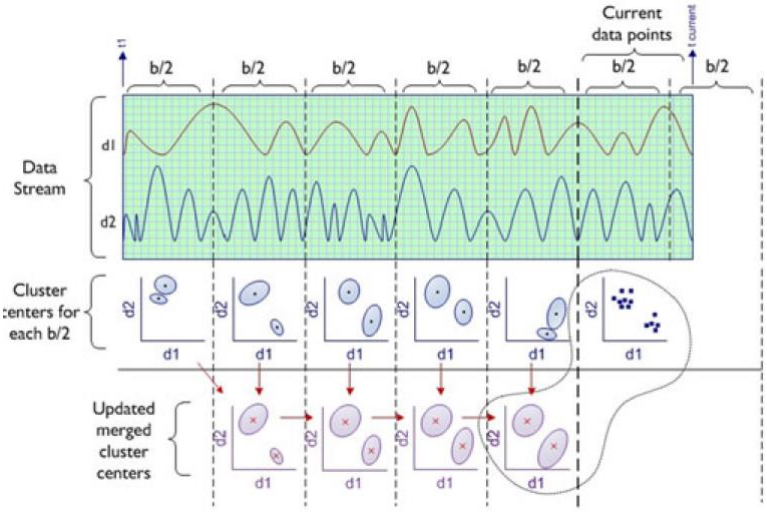
基本介绍：

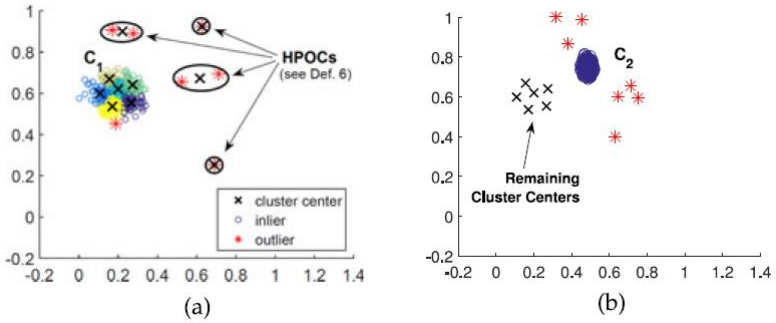
1、 该算法的核心点是解决在 LOF 算法中出现的内存问题，通过类似窗口限制，结合 K 均值聚类的算法，保证内存中所存储的数据不超过内存分配容量

2、 内存判断方案基本延续 LOF 判断法则

算法创新点：

1、 首先通过类似窗口限制方法，将数据限制在一定大小之类，接着利用 k 均值聚类方法得到新的聚类中心并删除部分数据，之后不断重复该过程，直至数据计算完毕。如下图：





2、 在数据计算 K 领域的时候有自己的解决方案，可以较快的帮助新插入的数据点找到 K 邻域，减少时间消耗

算法优缺点分析：

优点：

1、 对于如何解决算法的内存占用问题提供了一个解决方案，可以在算法程度上极大的解决内存问题

缺点：

1、 由于局部数据提取，所以最终的异常检测的算法精度会有所降低

2、 对连续异常序列检测效果较差

3、 在高维数据中效果较差

DILOF 算法：

来源：DILOF——EffectiveandMemoryEfficient LocalOutlierDetectioninDataStreams.pdf

基本介绍：

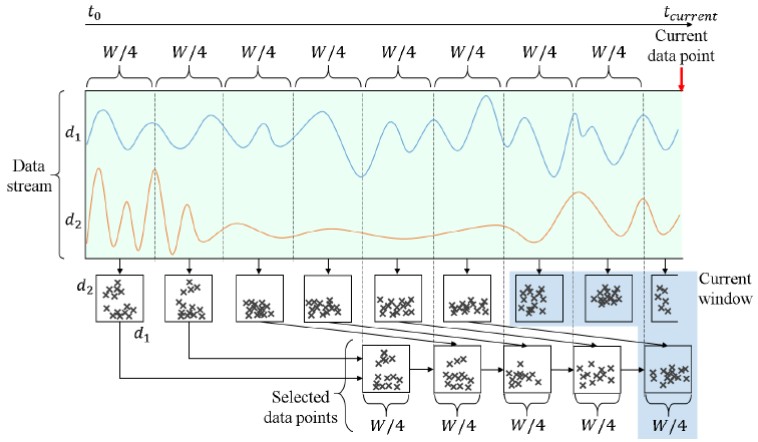
1、 主题思想基于 MILOF,ILOF,LOF 三种算法改进而来

2、 主要目的是为了综合综合性的解决 LOF 算法中的时间复杂度与空间复杂度问题

3、 提出了一种解决连续异常序列检测方案

算法创新点：

1、 改进了 MILOF 算法中的核心方案，不利用简单的聚类提取，而是利用密度提取方案，提取要删除区域的特征数据并保存在数据内存中，这样不断迭代下去，直至数据检测完成。



2、 提出了一种检测连续异常序列的方案（理论分析下来其实可行性并不太强），但是相较于 MILOF 与 LOF 有较大提升。

算法优缺点分析：

优点：

1、 在算法内存占用方面有较大的创新，同时在 MILOF 的基础上提高了检测的精度

2、 提出了自己的异常序列检测方案，在一定程度上提高了异常序列检测的精度

3、 讲特征提取问题转化为优化问题，创意性好

缺点：

1、 转化为优化问题之后实际在一定程度上又加大了计算量

2、 改进 MILOF 核心思路后，其实也在一定程度上复杂化了 MILOF 计算

CBLOF 算法：

来源：Discovering cluster-based local outliers.pdf

算法综述：

Step 1：首先选取一种聚类方法（论文中选取的是 Squeezer 聚类算法，其实如果数据不是具有几强的特征的话，任何聚类方法结果基本差不多）进行聚类，并得到一系列聚类。记录每个聚类中的点以及聚类大小。

Step 2：

$$(|C_1| + |C_2| + \cdots + |C_b|) \geqslant |D|^* \alpha \quad (1)$$

$$|C_b|/|C_{b+1}| \geqslant \beta \quad (2)$$

Then, the set of *large* cluster is defined as: $LC = \{C_i, |i \leqslant b\}$ and the set of *small* cluster is defined as: $SC = \{C_j | j > b\}$.

通过上述方法计算出 LC 与 SC.

Step 3：

$$\text{CBLOF}(t) = \begin{cases} |C_i|^* \min(\text{distance}(t, C_j)) & \text{where } t \in C_i, C_i \in SC \text{ and } C_j \in LC \text{ for } j = 1 \text{ to } b \\ |C_i|^* (\text{distance}(t, C_i)) & \text{where } t \in C_i \text{ and } C_i \in LC \end{cases}$$

在第一步和第二步的基础上按照上述计算公式分别计算出每个点的 CBLOF 值，并以此为评判标准来判断点异常与否。

其中 CBLOF 是利用数据点所在聚类区域的大小以及其与小聚类之间的距离之积。可以知道 CBLOF 越大意味着数据点处于高密度区域的概率越大，则是异常点的概率就越小。

优缺点分析：

有点：

1、 算法结构较为简单，适用于工程应用

2、 算法精度相对大多数常见数据来说已经足够

3、 考虑了数据点的密度特性

4、 算法的时间复杂度较低

缺点：

1、 算法结构过于简单，对于部分特殊数据检测效果会较差

2、 对于较长的连续异常序列检测效果会很差

3、 因为要存储大量数据，所以对内存要求很大

LOCI 算法：

来源：LOCI Fast outlier detection using the local correlation integral.pdf
算法综述：
Step 1：与 LOF 算法框架类似，主要不同点在于领域构造方式上，LOCI 邻域构造如下：

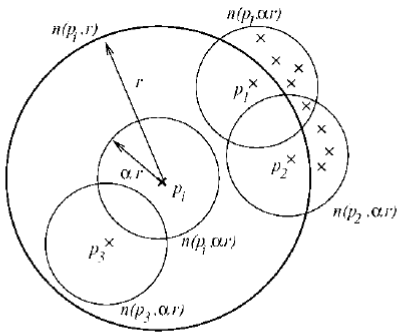


Figure 2. Definitions for n and $\hat{n}$—for instance $n(p_i, r) = 4$, $n(p_r, \alpha r) = 1$, $n(p_1, \alpha r) = 6$ and $\hat{n}(p_i, r, \alpha) = (1 + 6 + 5 + 1)/4 = 3.25$.

Step 2：了解邻域构造之后，按照下述公式求得每个点得 MDEF 值：

Definition 1 (MDEF) *For any p_i, r and α we define the* multi-granularity deviation factor *(MDEF) at radius (or scale) r as:*

$$MDEF(p_i, r, \alpha) = \frac{\hat{n}(p_i, r, \alpha) - n(p_i, \alpha r)}{\hat{n}(p_i, \alpha, r)} \quad (1)$$

$$= 1 - \frac{n(p_i, \alpha r)}{\hat{n}(p_i, \alpha, r)} \quad (2)$$

Step 3：再按照下述方式计算标准差：

$$\sigma_{MDEF}(p_i, r, \alpha) = \frac{\sigma_{\hat{n}}(p_i, r, \alpha)}{\hat{n}(p_i, r, \alpha)} \quad (3)$$

which is the normalized standard deviation $\sigma_{\hat{n}}(p_i, r, \alpha)$ of $n(p, \alpha r)$ for $p \in \mathcal{N}(p_i, r)$ (in Section 5 we present a fast, approximate algorithm for estimating σ_{MDEF}).

Step 4：按照 3σ原则，判断每个点是否处于相应得置信区间之内，并以此判断数据异常与否，其中 MDEF 是均值，σMDEF 是标准差
优缺点分析：
优点：
1、 再基于距离异常检测算法中提出了一个新的方向；算法整体计算过程较为简洁；算法精度较高
缺点：
1、 对连续异常检测效果较差；时间复杂度高；内存需求太大；不太适用于高维数据

SPOD 算法：

来源：基于局部信息熵的加权子空间离群点检测算法_倪巍伟.pdf

算法描述：

1、 该算法主要解决得是高维数据中出现的‘维度灾殃’现象，虽然又很多降维方法，但是现存得任何一种降维方法都不能完美的继承高维数据得所有特征

算法综述：

Step 1：通过信息熵方案选取每个数据点的离群属性与优选属性空间

熵是信息论中用来描述信息和随机变量不确定性的重要工具，设 X 为随机变量，其取值集合为 $S(X)$，$P(x)$表示 X 可能取值的概率，则 X 的熵定义为

$$E(X)=-\sum_{x\in S(X)}P(x)\log_2(P(x)).$$

变量的不确定性越大，熵也就越大，把它搞清楚所需要的信息量也就越大；熵值越小，不确定性越小．在此基础上，引入“局部属性熵”定义．

定义 5．局部属性熵(local entropy of attribute)．对 $p\in D$，$A_i\in A$，p 关于 A_i 的局部属性熵定义为

$$LEA_{A_i}(p)=-\sum_{q\in N_k(p)}\frac{dist(\Pi_{A_i}(p),\Pi_{A_i}(q))-d_{\min}}{d_{\max}-d_{\min}}\cdot\log_2\left(\frac{dist(\Pi_{A_i}(p),\Pi_{A_i}(q))-d_{\min}}{d_{\max}-d_{\min}}\right),$$

$$d_{\max}=\max\{dist(\Pi_{A_i}(p),\Pi_{A_i}(q))\mid q\in N_k(p)\},$$

$$d_{\min}=\min\{dist(\Pi_{A_i}(p),\Pi_{A_i}(q))\mid q\in N_k(p)\}.$$

定义 6．p 的离群属性(outlier attribute)．对 $p\in D$，$A_i\in A$，若满足如下关系：

$$LEA_{A_i}(p)\geqslant\frac{\sum_{q\in N_k(p)}LEA_{A_i}}{|N_k(p)|},$$

则称属性 A_i 为 p 的离群属性．

Step 2：根据不同属性得不同，在计算数据点之间得距离得时候，讲各属性得权重因素考虑在内。

Step 3：根据 LOF 算法得框架，计算核心距离、可达距离、局部可达密度、离群因子。其中距离公式按 STEP 2 中所提出得方案。

Step 4：计算每个数据得 SPOIF 值，以此作为评判数据异常与否得标准：

定义 17．p 的子空间离群影响因子(subsPac
outlier influence factor)．p 的子空间离群影响因
SPOIF 定义如下：

$$SPOIF(p)=\frac{den_{avg}(WN_k(p))}{den(p)},$$

其中 $den_{avg}(WN_k(p))=\frac{\sum_{q\in WN_k(p)}den(q)}{|WN_k(p)|}$.

优缺点分析：

优点：

对于高维数据异常检测提出了一个新的方向：能很好的解决单点异常的问题

确点：

与 LOF 基本一样，这里就不赘述了

DLOF 算法：

来源：一种基于密度的局部离群点检测算法 DLOF_胡彩平.pdf
算法描述：
DLOF 可以看作是一种简化版的 SPOD，在 SPOD 中计算每个数据的离群属性子集，而在 DLOF 中，在整体数据上计算每个属性的目标值，确定离群属性。再就是各种指标计算按照 LOF 算法来进行计算，整体结构变化不大，因此该算法不做过多描述。

NLOF 算法：

来源：NLOF_一种新的基于密度的局部离群点检测算法_王敬华.pdf
算法综述：
1、 首先利用 DBSCAN 聚类算法构造出一个异常点候选集，之后基于此候选数据集寻找异常点
2、 利用信息熵选择不同属性的权重
3、 利用 LOF 算法计算评价指标
4、 在算法过程中邻域更新过程中用了一个其它人提出的办法降低时间复杂度（但是效果远不如 ILOF）

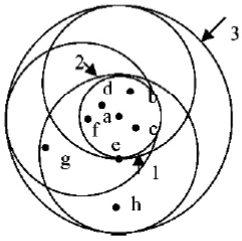


图 2　邻域查询优化示意图

具体过程与上述几种算法基本类似，这里不做过多赘述，具体细节可参考具体论文

SLOM 算法

该算法适合那种空间性数据，在日常应用场景中并不多见，利用的时候多会做一定程度的变形，因此具体细节参考具体论文为好。
来源：SLOM a new measure for local spatial outliers.pdf