2019算法设计与分析课程报告
时间序列异常点的检测和修复

1160300823 陈柯昊

2019 年 6 月 15 日

1 论文介绍

我选择的是Time Series Data Cleaning: From Anomaly Detection to Anomaly Repairing，2017年在VLDB发表的一篇文章。这篇文章最初吸引我是因为异常检测及修复对于时间数据的处理是一个很实际的场景，对数据进行合理的异常检测只是一个开端，正如文中所说，简单地删去异常数据会使得其他算法无法正常应用，所以一个准确、有效率的修复算法是很必要的。接近事实的数据修复也有助于其他算法的应用。

文章的主要贡献有:

1. 基于时间序列数据的迭代最小修复（IMR）的新框架
2. 对所提出的迭代最小修复的收敛的显式分析
3. 每次迭代中的参数的有效估计
4. 通过增量计算，将参数估计的复杂度从O（n）降低到O（1）

1.1 论文背景

时间序列数据经常被发现具有脏或不精确的值，例如GPS轨迹，传感器读取序列[15]，甚至股票价格[16]。例如，SALVEPAR（SY）的价格被误用为SYBASE（SY）的价格，两者在某些来源中共享相同的符号（SY）。

它与现实生活中实际发生的异常不同，例如，当冷空气涌入时，温度在一天内从20C突然变为10C。为了区分这类异常，我们提出采用一些标记的脏数据观察事实。

1.2 问题定义

考虑n个观察值的时间序列，$x = x[1], \ldots, x[n]$，其中每个$x[i]$是第i个数据点的值。为简洁起见，我们将$x[i]$写为x_i

设y表示x的标记/修复序列。每个y_i都是标记的真值或x_i的修复值。

给定时间序列x和部分标记的x的子集y，修复问题是确定未在y中标记的x_i的修复y_i。图表中给出了该论文中符号的定义:

符号	描述
x	n个数据点的观察序列
x_i	x中第i个数据的值，也被写作$x[i]$
y	标记的真值/x的修复序列
z	x与标记/修复的y的距离
$y^{(k)}$	第k次迭代的序列y
ϕ	p阶AR(p)/ARX(p)的参数
τ	提前定义的收敛阈值
Z, V	参数估计的输入矩阵
A, B	参数估计的中间值矩阵

表 1: 符号定义

1.3 算法介绍

本文中的修复算法与现有的修复算法不同，现有的修复算法可能通过一次修复过度修复数据，该文章提出逐步修复数据，并且注意到异常检测中的误差性质及数据修复中的最小变化原理，使得前一次迭代中的高

可信度信息可以帮助后续步骤的修复，称为迭代最小修复法IMR,(Iterative Minimum Repairing)。

1.3.1 设计思想

设$y^{(k)}$表示第k次迭代中的序列y,其中$y^{(0)}$是输入中部分标记的时间序列。因为$y^{(0)}$是不完整的(部分标记的)，为了初始化，如果$y_t^{(0)}$是未标记的，则指定$y_t^{(0)} = x_t$。标记值不应该被修复，例如：在$y_t^{(0)}$被标记的情况下，$y_t^{(k)} = y_t^{(0)}$。

算法1中描述了迭代最小修复过程IMR(p)，其输入是观察时间序列x，并且部分标记为$y^{(0)}$。它输出所有标记的$y_t^{(0)}$未改变且w未标记的$y_t^{(0)}$被修复的$y_t^{(0)}$。主要步骤包括：

1. **参数估计**,在第2行中，从x和当前$y^{(k)}$学习第k次迭代中的ARX（p）的参数，由$\phi(k)$表示。

2. **修复候选的产生**,在第3行中，根据关于$x, y^{(k)}, \phi(k)$的ARX(p)计算可能的修复$\hat{y}^{(k)}$。

3. **评估修复**，在第4行中，由数据修复中的最小更改原则确定要接受的修复之一，$y_t^{(k+1)} = \hat{y}_t^{(k)}$。如第5行所示，重复该过程，直到修复收敛。如

$$\left| y_j^{(k)} - y_j^{(k+1)} \right| \leq \tau, j = 1, \ldots, n \tag{1}$$

其中τ是收敛阈值，或达到最大迭代次数。

1.3.2 参数估计

在给定$x, y^{(k)}$的情况下，参数估计步骤1(算法1的第2行中)估计ARX(p)的参数$\phi(k)$。可以直接使用诸如普通最小二乘[20]或Yule-Walker方程[7]的现有方法。例如，通过普通最小二乘法，我们有

$$\phi^{(k)} = \left(\left(Z^{(k)} \right)' Z^{(k)} \right)^{-1} \left(Z^{(k)} \right)' V^{(k)} \tag{2}$$

Algorithm 1: IMR(p)

Input: time series x and partially labeled $y^{(0)}$

Output: $y^{(k)}$ with all the labeled $y_i^{(0)}$ unchanged and unlabeled $y_j^{(0)}$ repaired

```
for k ← 0 to max-num-iterations do
    φ^(k) ← Estimate (x, y^(k)) ;;
    ŷ^(k) ← Candidate (x, y^(k), φ^(k)) ;;
    y^(k+1) ← Evaluate (x, y^(k), ŷ^(k)) ;;
    if Converge (y^(k), y^(k+1)) then
        break;;
    k ← k + 1;
return y^(k)
```

其中

$$\boldsymbol{V}^{(k)} = \begin{pmatrix} y_{p+1}^{(k)} - x_{p+1} \\ y_{p+2}^{(k)} - x_{p+2} \\ \vdots \\ y_n^{(k)} - x_n \end{pmatrix} \tag{3}$$

,

$$\phi^{(k)} = \begin{pmatrix} \phi_1^{(k)} \\ \phi_2^{(k)} \\ \vdots \\ \phi_p^{(k)} \end{pmatrix} \tag{4}$$

,

$$Z^{(k)} = \begin{pmatrix} y_p^{(k)} - x_p & y_{p-1}^{(k)} - x_{p-1} & \dots & y_1^{(k)} - x_1 \\ y_{p+1}^{(k)} - x_{p+1} & y_p^{(k)} - x_p & \dots & y_2^{(k)} - x_2 \\ \vdots & \vdots & \ddots & \vdots \\ y_{n-1}^{(k)} - x_{n-1} & y_{n-2}^{(k)} - x_{n-2} & \dots & y_{n-p}^{(k)} - x_{n-p} \end{pmatrix} \tag{5}$$

因为采用了迭代修复，在线增量参数估计是必要的，这在以前的研究中没有研究过。

1.3.3 修复候选的产生

修复候选者生成步骤2(在算法1的第3行中)使用ARX(p)根据估计参数$\phi^{(k)}$来推断候选修复$\hat{y}^{(k)} = \phi^{(k)} \cdot \left(y^{(k)} - x\right) + x$。更具体的说，对于每个点$t$，$\hat{y}_t^{(k)}$由下式给出

$$\hat{y}_t^{(k)} = \sum_{i=1}^{p} \phi_i^{(k)} \left(y_{t-i}^{(k)} - x_{t-i}\right) + x_t \tag{6}$$

根据$y_{t-1}^{(k)}, \ldots, y_{t-p}^{(k)}$。我们注意到，只有具有$\left|\hat{y}_t^{(k)} - y_t^{(k)}\right| > \tau$的候选者才需要考虑之前等式中的收敛条件。

1.3.4 评估修复

修复评估步骤3(在算法1的第4行中)选择一个修复来接受，即，为上述生成的修复候选分配$y_t^{(k+1)} = \hat{y}_t^{(k)}$。遵循数据修复中的最小变化原则，从其原始输入中最小化的修复是优选的，具有更高的置信度。每次迭代中修复的结果是：

$$y_t^{(k+1)} = \begin{cases} \hat{y}_t^{(k)} & \text{if } t = \arg\min_i \left|\hat{y}_i^{(k)} - x_i\right| \\ y_t^{(k)} & \text{otherwise} \end{cases} \tag{7}$$

值得注意的是，在每次迭代中只修复了一个具有最小变化（最大限度）的数据点，这比最小化整体变化关于完整性约束的NP难问题更为有效。

数据修复中的最小变化原则[1]基于人或系统总是试图最小化其错误的直觉。但是，不能保证最小变化修复总是对应于真实值。因此，类似于其他基于最小变化的数据修复研究[1,8]，最终结果的准确性不太可能具有理论上的保证，因为对于误差可能与事实偏离的程度没有限制。出于这个原因，我们只能通过与实验中的基本事实进行比较来评估修复的正确性，类似于其他数据修复研究[1,8]。然而，我们可以证明高效的修剪和增量计算是安全的，即理论上保证最终结果的准确性与原始IMR的结果相同，而不需要修剪和增量计算。

文章随后对算法的收敛性进行了分析，这一部分被我进行了略过的处理，因为难度有些大，不是很容易理解。

在算法1的三个主要步骤中，虽然修复候选生成和评估对于最小修复是不可避免的，但具有高昂代价的的参数$\phi(k)$估计在迭代修复场景中是可

以优化的。首先，我们确定参数估计的矩阵$Z^{(k)}, V^{(k)}$可以通过简单地删除值为0的行来修剪。此外，可以设计增量计算，使得参数估计的复杂度从O（n）减少到O（1）。

1.3.5 矩阵修剪

回忆一下在算法1中估计参数$\phi^{(k)}$时，我们需要分别考虑两个大矩阵$Z^{(k)}$和$\boldsymbol{V}^{(k)}$，$Z^{(k)}$和$\boldsymbol{V}^{(k)}$中的值$y_i^{(k)} - x_i$表示标记/修复值$y_i^{(k)}$与点i的输入值$\mathcal{X}_i$之间的差。在实践中，标记数据通常是有限的，而根据修复的最小变化原则，修复数据不应该有显着的改变。也就是说，$Z^{(k)}$和$\boldsymbol{V}^{(k)}$中的大多数值等于0。稀疏矩阵只要通过删除掉存在0的行就可以完成修剪。

简单起见，令$z_i^{(k)} = y_i^{(k)} - x_i$。我们把算法1中的式子重写如下。

$$\boldsymbol{V}^{(k)} = \begin{pmatrix} z_{p+1}^{(k)} \\ z_{p+2}^{(k)} \\ \vdots \\ z_n^{(k)} \end{pmatrix} \tag{8}$$

$$Z^{(k)} = \begin{pmatrix} z_p^{(k)} & \cdots & z_1^{(k)} \\ z_{p+1}^{(k)} & \cdots & z_2^{(k)} \\ \vdots & \ddots & \vdots \\ z_{n-1}^{(k)} & \cdots & z_{n-p}^{(k)} \end{pmatrix} \tag{9}$$

去除$Z^{(k)}$中的值（其值等于0）和$\boldsymbol{V}^{(k)}$中的相应行之后，仍然可以计算相同的参数$\phi(k)$。

命题 1. *对于$Z^{(k)}$中的任一行，用$Z_r^{(k)}$来表示，如果整行全都为0，即$z_{r+p-1}^{(k)} = z_{r+p-2}^{(k)} = \ldots = z_r^{(k)} = 0$，那么分别从$Z^{(k)}$和$\boldsymbol{V}^{(k)}$中删除行$Z_r^{(k)}$和对应的行$\boldsymbol{V}_r^{(k)} = \left(z_{p+r}^{(k)}\right)$是安全的，即计算的是同一个参数$\phi(k)$。*

1.3.6 增量计算

在算法1的每次迭代中，参数$\phi(k)$都在等式2中在n个点上通过$Z^{(k)}$和$\boldsymbol{V}^{(k)}$来进行估计。但是根据修复最小变化原理，每次迭代中只有一点r被改变，即$y_r^{(k)} \neq y_r^{(k-1)}$。这就是说，$Z^{(k)}$和$\boldsymbol{V}^{(k)}$中大部分的点都是没有被改变的，

与$Z^{(k-1)}$和$\boldsymbol{V}(k-1)$的值是相同的。在下面的命题中，我们说明了参数$\phi(k)$是可以通过只考虑变化值而非整个$Z^{(k)}$和$\boldsymbol{V}^{(k)}$来进行估计。因此，每次迭代中的时间复杂度从线性时间降低到了常数时间。

1.3.7 算法执行流程

1.3.8 例子

1.4 理论和实验结论

2 领域综述

2.1 领域背景

2.2 方法介绍

3 改进点概述

3.1 stream and multi-varite

在实际应用中,时间序列往往以多维度、数据流的形式出现，这就要求本文中的算法是一个在线算法，并且

4 算法与分析

4.1 算法设计思想

4.2 算法过程描述

4.2.1 自然语言描述

4.2.2 伪代码描述

4.2.3 伪代码逐行解释

4.2.4 例子

4.3 算法的结果