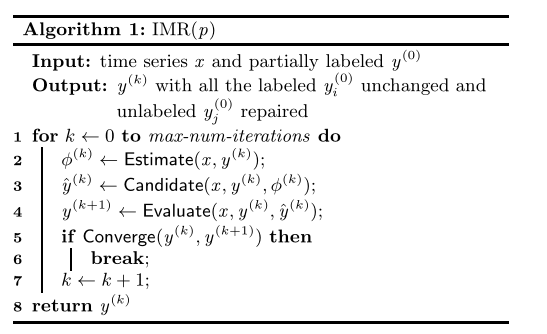
1. 修复算法

与现有的异常检测模型可能通过一次性修复过度改变数据不同（如引言中所述，如图 2 所示，并在第 6 节的实验中观察到），我们建议逐步修复数据，注意异常检测中的错误性质和数据修复中的最小变化原则，以便前迭代中的高置信度修复有助于后面步骤中的修复。

* 1. 迭代修复

让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。由于 y (0) 不完整（部分标记），为了初始化，如果 y (0) t 没有标记，我们分配 y (0) t = xt。回想一下，标记的值不应该被修复，即如果 y (0) t 被标记，则 y (k) t = y (0) t。算法 1 提出了迭代最小修复过程 IMR(p)，其输入是观察时间序列 x，部分标记为 y(0)。它输出 y (k)，所有标记的 y (0) t 不变，未标记的 y (0) t 修复。主要步骤包括： (S1) 参数估计，在第 2 行中，从 x 和当前 y (k) 中学习第 k 次迭代中 ARX(p) 的参数，用 φ(k) 表示。 (S2) 修复候选生成，在第 3 行，根据 ARX(p) w.r.t 计算可能的修复 ^y (k)。 x, y (k) 和 φ(k)。 (S3) 修复评估，在第 4 行，确定要接受的修复之一，y (k+1) t = ^y (k) t ，参考数据修复中的最小变化原则 [1]。

如第 5 行所示，该过程重复，直到修复收敛，例如，具有 |y (k) j − y (k+1) j | ≤ τ, j = 1, . . . , n. (4) 其中 τ 是收敛阈值，或达到最大迭代次数。设置 max-num-iterations 是一种避免在实践中等待收敛的补救措施（有关讨论和评估，请参见第 6.1.4 节）

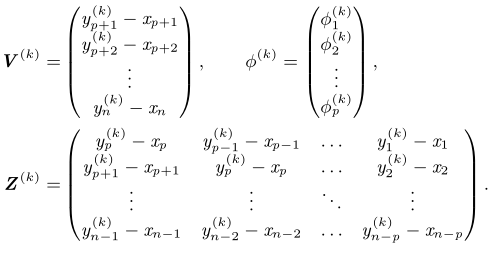


示例 5

（算法概述，示例 2 续）。再次考虑图 2 中的 x = {6, 10, 9.6, 8.3, 7.7, 5.4, 5.6, 5.9, 6.3, 6.8, 7.5, 8.5}。根据五个标记的数据点，我们分配 y (0) = {6, 5.6, 5.4, 8.3, 7.7, 5.4, 5.6, 5.9, 6.3, 6.8, 7.5, 8.5}，其中未标记的点由 y (0) t = xt 初始化，例如 y (0) 4 = x4 = 8.3。在每次迭代中，IMR 算法 (1) 学习参数，例如，对于 p = 1，φ(0) 1 = 0.5； (2) 生成候选修复，如 ^y (0) = {-, -, -, 6.2, 7.7, -, 5.6, 5.9, 6.3, 6.8, 7.5, -}； (3) 选择一个修复进行，并形成新的序列，假设 y (1) = {6, 5.6, 5.4, 6.2, 7.7, 5.4, 5.6, 5.9, 6.3, 6.8, 7.5, 8.5}。重复该过程直到收敛。最终输出为 y (7) = {6, 5.6, 5.4, 5.20, 5.39, 5.4, 5.6, 5.9, 6.3, 6.8, 7.5, 8.5}，RMS 误差为 0.03。以下示例中介绍了每个步骤的详细信息。

* 1. 参数估计

参数估计步骤S1（在算法1的第2行）估计ARX（p）的参数φ（k），给定x，y（k）。可以直接使用现有的方法，例如普通最小二乘法 [20] 或 Yule-Walker 方程 [7]。例如，通过普通最小二乘法，我们有 φ(k) = ((Z (k))′Z (k))−1(Z (k))′V (k) (5) 其中



示例 6

（对 y (0) 的参数估计，示例 5 继续）。考虑 x = {6, 10, 9.6, 8.3, 7.7, 5.4, 5.6, 5.9, 6.3, 6.8, 7.5, 8.5} 和 y (0) = {6, 5.6, 5.4, 8.3, 7.7, 5.4, 5.6, 5.9 , 6.3, 6.8, 7.5, 8.5}。给定顺序 p = 1，我们有 V (0) = {−4.4, −4.2, 0, 0, 0, 0, 0, 0, 0, 0, 0}′ 有 11 行和 1 列，并且 Z (0 ) = {0, -4.4, -4.2, 0, 0, 0, 0, 0, 0, 0, 0}' 有 11 行和 1 列。参考公式 5，参数估计为 φ(0) 1 = (-4.4) ∗ (-4.2) (-4.4)2 + (-4.2)2 = 0.5。由于迭代修复，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（参见我们在第 5 节中的方法）。

* 1. 候选生成修复

候选生成步骤 S2（在算法 1 的第 3 行）使用 ARX(p) 来推断候选修复 ˆy (k) = φ(k) ·(y (k) -x) +x，参考估计的参数 φ(k)。更具体地说，对于每个点 t， ˆy (k) t 由 ˆy (k) t = pX i=1 φ(k) i (y (k) t−i − xt−i) + xt (6) 给出到 y (k) t−1, . . . , y (k) t-p。我们注意到只有 |^y (k) t − y (k) t | > τ 需要参考方程 4 中的收敛条件来考虑。

示例 7（修复候选 ^y (0)，示例 6 继续）。考虑示例 6 中估计的参数 φ(0) 1 = 0.5。令阈值 τ = 0.1。参考等式 6，我们有 ^y (0) 4 = 0.5 ∗ (5.4 − 9.6) + 8.3 = 6.2 其中 |^y (0) 4 − y (0) 4 | = |6.2 - 8.3| = 2.1 > 0.1，且 ^y (0) 5 = 0.5 ∗ (8.3 − 8.3) + 7.7 = 7.7 with |7.7 − 7.7| = 0 < 0.1。修复候选是 ^y (0) = {+, +, +, 6.2, -, +, -, -, -, -, -+} 其中“+”对应于标记点，“-”表示没有候选。也就是说，我们只需要考虑一个候选 ^y (0) 4 进行修复。

* 1. 修复评估 修复评估步骤 S3（算法 1 的第 4 行）选择一个修复接受，即分配 y (k+1) t = ^y (k) t 上述生成的修复候选。遵循数据修复中的最小变化原则[1]，与其原始输入差异最小的修复具有更高的置信度。每次迭代的修复结果为： y (k+1) t = ( ^y (k) t if t = arg mini |^y (k) i − xi| y (k) t else . (7) 值得注意的是，只有一个在每次迭代中修复具有最小变化（最自信）的数据点，这比 NP-hard 最小化整体变化 w.r.t. 完整性约束 [1] 的问题更有效。

示例 8

（最小修复 ^y (1) t ，示例7 续). 由于例 7 中只有一个修复候选，即 ^y (0) 4 = 6.2，它是最小修复（在所有候选中）。第一次迭代后的序列变为 y (1) = { 6、5.6、5.4、6.2、7.7、5.4、5.6、5.9、6.3、6.8、7.5、8.5}。

请注意，数据修复中的最小更改原则 [1] 是基于人类或系统总是试图最小化错误的直觉。但是，不能保证最小更改修复始终对应于真实值。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有理论上的保证，因为对于错误与事实的偏离程度没有限制。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建议修复的正确性，这也类似于其他数据修复研究 [1, 8]。尽管如此，我们可以证明高效剪枝和增量计算是安全的（命题 9 和 10），即理论上保证了高效计算的最终结果的准确性与没有剪枝和增量计算的原始 IMR 的结果相同.

1. 收敛性分析

在本节中，我们分析迭代修复的收敛性，即 limk→+∞ Pn i=1 ? y (k+1) 我 - y (k) 我 ? = 0，这对于终止算法 1 是必不可少的。虽然一般收敛问题仍然存在，但我们在本节中研究提出的方法在某些特殊情况下的收敛性，原因有两个：（1）我们说明了基于 ARX 的方法确实是第 4.1 节中提出的带有静态参数（命题 2）的 IMR 的一个特例； (2) 我们在第 4.2 节中确定了另一种具有收敛保证的特殊情况，它可以在没有迭代的情况下在线修复流数据（更多细节和实验请参见第 4.2.2 节和第 6.2.2 节）。

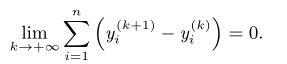
* 1. 静态参数

我们研究这个特殊情况是为了说明我们提出的 IMR 和现有 ARX 之间的关系。让我们首先分析 IMR 的收敛性（命题 1），然后说明它们在特定情况下的等价性（命题 2）。在算法 1 的第 2 行中，不是在每次迭代中动态更新参数 φ(k)，而是为所有迭代指定静态参数 φ(k) = φ(0)。命题 1. 在静态参数 φ(k) = φ, ∀k 的情况下，修复结果收敛，即 limk→+∞ nX i=1 ? y (k+1) 我 - y (k) 我 ? = 0. IMR(1) 的特例 我们在下面展示对于 p = 1，第 2.3 节中基于 ARX(p) 的修复是我们提出的具有静态参数 φ(0) 的 IMR(p) 的特例。这种等价性证明了我们提议的基本原理。命题 2. 对于静态参数 φ(k) = φ, ∀k 的 IMR(1)，算法 1 等价于基于 ARX(1) 的修复。

* 1. 收敛参数

我们现在考虑在每次迭代中动态更新的参数 φ(k)，在算法 1 的第 2 行中。如下面的命题 3 所示，如果动态参数收敛，则修复也收敛。这种收敛参数情况很有趣，因为在某些情况下可以直接计算相应的收敛修复结果，而无需迭代计算，如下图所示。

命题3. 如果参数收敛，limk→+∞ φ(k) = φ，那么修复也收敛



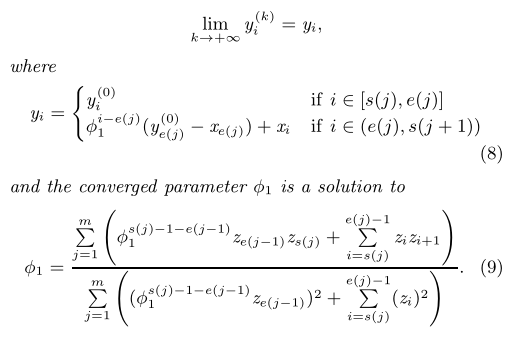
* + 1. IMR(1) 的特例

同样，我们考虑阶数 p = 1 的 IMR(1) 的特例。为了说明如何在没有迭代的情况下直接计算修复结果，我们首先说明任何 y(k)在算法 1 中生成的 t 可以表示如下，也就是 y (k) t 的出处。

引理 4. 对于 IMR(1)，我们可以用 y (k) t = φ(ks) 1 φ(ks−1) 1 来表示每个 y (k) t。 . . φ(k1) 1 (y (0) t−s − xt−s) + xt，其中 0 < k1 < · · · < ks−1 < ks < k 表示迭代次数，y (0) t−s 被标记真值，时间点 t−s+1, . . . , t 没有标记。

我们用多个（比如 m）段表示 y (0) 中的标记点。让 s(j) 和 e(j) 表示第 j 个标记段的起点和终点，j = 1，。 . . , 米。例如，图 2 中有 3 段标记数据点，s(1) = 1, e(1) = 2, s(2) = 4, e(2) = 5, s(3) = 9 , e(3) = 9. 让所有标记点 i 的 zi = y (0) i -xi。 （我们在未定义的段 0 上设置 ze(0) = 0。）

命题 5。对于 IMR(1)，如果参数收敛，则 limk→+∞ φ(k) 1 = φ1，则收敛的修复结果可以直接由



收敛参数 φ1 是

* + 1. IMR(1) 有一个标记段

我们考虑只有一个长度为 ℓ 的段在 y (0) 的开头被标记的情况，即 y (0) 1 , y (0) 2 , . . . , y (0) ℓ 被标记。在这种特殊情况下，可以直接计算收敛参数和修复结果而无需迭代，最重要的是，它可以通过将所有历史数据解释为一个标记的片段来实现高效的在线计算（详见第 6.2.2 节）。值得注意的是，在这种情况下不需要设置阈值。想法是：（1）我们首先在引理 6 中证明，在一定的输入下，每次迭代的估计参数确实是有界的； (2) 命题 7 说明有界参数导致收敛参数； (3) 最后，类似于命题 5，给定收敛参数，我们直接计算收敛修复，无需命题 8 中的迭代计算。

引理 6. 对于 IMR(1)，前 ℓ 数据点标记在 y (0) 中。如果输入满足 ???? ℓ−1P t=1 z (0) t z (0) t+1 ???? < ℓ−1P t=1 z (0) t z (0) t ，即 ??? ℓ−1X t=1 (y (0) t − xt)(y (0) t+1 − xt+1) ????? < ℓ−1X t=1 (y (0) t − xt)2, 那么我们有 |φ(k) 1 | < 1 在迭代 k 中，0 ≤ k ≤ n - ℓ。以下结论说明在每次迭代中使用有界参数时，参数会收敛

命题 7. 对于 IMR(1)，前 ℓ 个数据点标记在 y(0) 中，如果 |φ(k) 1 | < 1 在迭代 k 中，0 ≤ k ≤ n - ℓ，则参数收敛，即 limk→+∞ φ(k) 1 = φ1，值得注意的是条件 |φ(k) 1 |命题 7 中 < 1 的参数收敛可以在实际数据中普遍观察到。首先，参考[5]，实践中的大多数时间序列都是平稳的，保证有|φ(k) 1 | < 1 for p = 1。此外，对于非平稳情况，典型的处理方式是通过差分将其转换为平稳[5]。

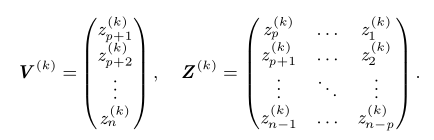
命题 8. 对于 IMR(1)，前 ℓ 个数据点标记在 y(0) 中，如果参数收敛，则 limk→+∞ φ(k) 1 = φ1，则收敛修复结果为 limk→+∞ y ( k) t = yt = ( y (0) i if i ∈ [1, ℓ] φi−ℓ 1 (y (0) ℓ − xℓ) + xi if i > ℓ (10) 其中收敛参数可以直接计算由 φ1 = (y (0) 1 − x1)(y (0) 2 − x2) + ··· + (y (0) ℓ−1 − xℓ−1)(y (0) ℓ − xℓ) (y (0) 1 − x1)2 + ··· + (y (0) ℓ−1 − xℓ−1)2 . (11) 命题 8 的方程 11 中的收敛参数 φ1 恰好是解，这并不奇怪命题 5 中公式 9 的一个标记段的特殊情况，其中 m = 1。

1. 有效参数估计

在算法 1 的三个主要步骤中，虽然修复候选生成和评估（在第 3.3 和 3.4 节中）对于最小修复是不可避免的，但我们在本节中展示了代价高昂的参数 φ(k) 估计（在第 3.2 节中）在我们的迭代修复场景中是可优化的。首先，我们发现在 5.1 节中，可以通过简单地删除值为 0 的行来修剪用于参数估计的矩阵 Z (k)、V (k)。此外，在 5.2 节中，可以设计增量计算，使参数估计的复杂度从 O(n) 降低到 O(1)。

* 1. 矩阵剪枝直觉。

回想一下，在估计公式 5 中的参数 φ(k) 时，我们需要考虑两个大小分别为 (n - p) × p 和 (n - p) × 1 的大矩阵 Z (k) 和 V (k)。 Z (k) 和 V (k) 中的值 y (k) i - xi 表示标记/修复值 y (k) i 与点 i 的输入值 xi 之间的 1051 差。在实践中，标注的数据往往是有限的，而修复后的数据不应该有明显的变化，参考修复的最小变化原则。也就是说，Z (k) 和 V (k) 中的大多数值等于 0。我们证明（在下面的命题 9 中）可以通过删除值为 0 的行来修剪稀疏矩阵。令 z (k) i = y ( k) i - xi 为简单起见。我们将方程 5 中的 V (k), Z (k) 重写为参数估计，



以下结论表明，相同的参数 φ(k) 仍然可以通过等式 5 计算，在删除 Z (k) 中的值等于 0 的行和 V (k) 中的相应行之后。

命题 9.

对于 Z (k) 中的任意一行，记为 Z (k)r ，如果整行的值都为 0，即 z (k) r+p−1 = z (k) r+p− 2 = 。 . . = z (k)r = 0，那么删除行 Z (k)r 和对应的行 V (k)r = 是安全的z (k) p+r ?分别来自矩阵 Z (k) 和 V (k)，它们仍然计算相同的 φ(k)。

示例 9（使用矩阵剪枝进行参数估计，示例 6 继续）。再次考虑 x = {6, 10, 9.6, 8.3, 7.7, 5.4, 5.6, 5.9, 6.3, 6.8, 7.5, 8.5} 和 y (0) = {6, 5.6, 5.4, 8.3, 7.7, 5.4, 5.6,示例 6 中的 5.9, 6.3, 6.8, 7.5, 8.5}。给定阶数 p = 1，我们有 V (0) = {−4.4, −4.2, 0, 0, 0, 0, 0, 0, 0, 0, 0}′ 大小为 11×1，Z (0) = {0, -4.4, -4.2, 0, 0, 0, 0, 0, 0, 0, 0}′ 大小为 11 × 1。如图9所示，除Z(0)中的第四行和第五行以及V(0)中的对应行之外的所有行都可以被删除。在矩阵剪枝之后，我们有大小为 2 × 1 的 V (0) = {−4.2, 0}′，大小为 2 × 1 的 Z (0) = {−4.4, −4.2}′。参考等式 5，参数由 φ(0) 1 = (−4.4) ∗ (−4.2) (−4.4)2 + (−4.2)2 = 0.5 估计。计算的参数与示例 6 中的相同。