BALANCE：用于根因定位的贝叶斯线性归因

Chaoyu Chen†
蚂蚁集团
中国
chris.ccy@antgroup.com

Hang Yu†
蚂蚁集团
中国
hyu.hugo@antgroup.com

Zhichao Lei
蚂蚁集团
中国
leizhichao.lzc@antgroup.com

Jianguo Li*
蚂蚁集团
中国
lijg.zero@antgroup.com

Shaokang Ren
蚂蚁集团
中国
renshaokang.rsk@antgroup.com

Tingkai Zhang
蚂蚁集团
中国
tingkai.ztk@antgroup.com

Silin Hu
蚂蚁集团
中国
husilin.hsl@antgroup.com

Jianchao Wang
蚂蚁集团
中国
luli.wjc@antgroup.com

Wenhui Shi
Ocean Base
中国
yushun.swh@oceanbase.com

arXiv:2301.13572v1 [cs.LG] 31 Jan 2023

摘要

根因分析（RCA）在分布式数据系统的运维中发挥着不可或缺的作用，因为它弥合了故障检测与系统恢复之间的鸿沟。现有研究主要关注多维定位或基于图的根因定位。本文探索了利用近期发展的可解释人工智能（XAI）框架开展根因分析的可能性。具体而言，我们提出 BALANCE（用于根因定位的贝叶斯线性归因），通过 XAI 中的归因视角对根因分析问题进行建模，并试图根据候选根因的行为解释目标关键绩效指标中的异常。BALANCE 由三个创新组件构成。首先，我们提出贝叶斯多重共线性特征选择（BMFS）模型，以正向方式根据候选根因预测目标关键绩效指标，同时促进稀疏性，并兼顾候选根因之间的相关性。其次，我们引入归因分析，以反向方式计算每个候选项的归因分数。第三，当存在多个关键绩效指标时，我们合并与各关键绩效指标相关的估计根因。我们在一个合成数据集以及三个真实世界根因分析任务上对所提出的 BALANCE 方法进行了广泛评估，即异常 SQL 定位、容器故障定位和 Exathlon 的故障类型诊断。结果表明，在准确率方面，BALANCE 优于最先进方法（SOTA），同时运行时间最短；其准确率至少比最先进方法高出 6%，优势显著用于实际任务。BALANCE 已部署到生产环境，以应对现实世界的根因分析问题；在线结果进一步证明了其在分布式数据系统中用于实时诊断的价值。

CCS 概念

• **软件及其工程**; • **信息系统 → 自治数据库管理**; • **计算方法学 →特征选择**; **正则化**;

关键词

根因分析，贝叶斯方法，错误SQL，故障诊断，分布式系统，归因分析，可解释人工智能

ACM参考格式：
Chaoyu Chen†, Hang Yu†, Zhichao Lei、Jianguo Li、Shaokang Ren、Tingkai Zhang、Silin Hu、JianchaoWang 和 Wenhui Shi。2023 年。BALANCE：用于根因定位的贝叶斯线性归因。载于《*2023* 年数据管理国际会议论文集》（*SIGMOD '23*）。ACM，美国纽约州纽约市，共 15 页。https://doi.org/XXXXXXX。XXXXXXX

1 引言

系统故障和事故可能对分布式数据系统产生巨大影响。这些系统广泛应用于现代信息技术（IT）和金融公司，因为相关故障可能导致系统瘫痪，进一步造成惊人的财务损失，并危及客户信任 [21]。Forbes 报道称，每年 IT 停机时间仅因收入损失就造成估计 265 亿美元的损失，更不用说包括客户和推荐流失在内的间接支出。因此，在故障演变为影响服务之前，必须及时、精准地开展故障诊断和恢复。故障诊断和恢复的一项核心任务是根因分析（RCA），它弥合了故障检测与恢复之间的鸿沟 [11, 13]。

目前，根因分析任务主要由经验丰富的站点可靠性工程师（SREs）完成。然而，随着架构规模和复杂度的提升，以及系统指标和事件的动态性与不可预测性不断增强，这类人工工作变得极其缓慢，原因在于

*通讯作者。† 两位作者对本项工作贡献相同。

允许免费制作本作品全部或部分内容的数字副本或纸质副本，用于个人或课堂用途，前提是这些副本不用于营利或商业利益，且首页保留此声明和完整引文。对于本作品中归 ACM 以外各方所有的内容，必须尊重其版权。注明出处后允许摘录。若要以其他方式复制、再版、发布到服务器或分发给他人，须事先获得明确许可和/或支付费用。请向 permissions@acm.org 申请许可。

SIGMOD '23，*2023*年*6*月*18*日至*23*日，美国华盛顿州西雅图© 2023年 计算机协会。ACM ISBN 978-1-4503-XXXX-X/18/06...$15.00
https://doi.org/XXXXXXX.XXXXXXX

其偏离了效率要求。事实上，正如 [19], 中所述，诊断分布式数据库系统中的间歇性慢查询的根因，可能需要数小时的人工工作。这促使研究人员投入大量精力，致力于设计基于机器学习的自动化根因分析算法，以节省时间并最终降低成本。

关于根因分析算法的文献大致可分为两类。第一类专注于多维根因定位 [5, 32, 47], ，旨在通过识别其对应多维属性中的故障指示组合，解释可加性关键绩效指标（关键绩效指标）的异常行为。这些算法的成功依赖于两个假设：1）关键绩效指标在每个维度上的值等于其属性值之和；2）所有关键绩效指标及其属性均可被监测。然而，在现实问题中，这两个假设可能过于严格；更实际的设定是在不作可加性假设且允许存在缺失数据的情况下，将异常归因于根因候选项。另一方面，第二类围绕基于图的根因分析算法展开 [14, 24, 38, 39]。这类方法通常首先基于服务调用追踪或因果发现算法 [46] 构建因果图，然后通过基于规则的遍历或随机游走找到根因节点。追踪图和基于规则的遍历在应用中的一个主要障碍是其具有侵入性，且通常需要艰苦地枚举所有追踪路径和规则。作为替代方案，研究人员采用因果发现方法来学习图结构，如 [39]所示。不幸的是，因果发现方法同时面临较高的计算复杂度和样本复杂度 [46]，，因此在处理大型图时可能会慢得令人难以接受；当图中所有指标的观测值数量较少时，还可能导致结果不准确。在获得图之后，随机游走方法具有启发式特征；当随机游走次数不够多时，可能无法收敛到根因。

在本文中，我们探索了替代方案，并将根因分析问题重新表述为特征归因问题 [16]。据我们所知，我们是首批从归因视角分析根因的研究者之一。作为可解释人工智能（XAI）中常用的工具，归因方法会为输入特征分配归因分数，其绝对值表示这些特征对模型预测或性能的重要性 [16]。类似地，我们旨在找出能够最佳解释告警关键绩效指标的根因。在根因分析问题中，候选原因的归因分数表示其与告警关键绩效指标的相关性或贡献度。以数据库系统为例，“坏 SQL”是指由于索引错误或执行计划发生变化而导致性能恶化的 SQL。这些 SQL 的性能下降通常会导致租户关键绩效指标出现异常，并可能严重影响用户体验。在这种情况下，目标（$\boldsymbol{y}$）是租户关键绩效指标，而候选原因（$\boldsymbol{X}$）是 SQL 指标。

随后，归因任务可以分为两个步骤完成：首先，构建一个前向模型，利用输入特征（即候选原因）预测输出（即告警关键绩效指标）；接下来，以反向方式通过归因方法评估输入特征的重要性。特别是在异常 SQL 定位示例中，候选 SQL 的数量 p因情况而异，最多可达数千条，而观测值的数量 n （相应时间序列的长度）通常较少，因为我们只关注异常附近的部分。换言之，在根因分析问题中，维度 p 可能大于样本量 n 。为解决这一问题并实现特征选择过程的自动化，我们采用稀疏线性模型作为前向模型，因为它具有很高的灵活性、效率和可解释性。此外，候选原因通常彼此相关，并且经常存在缺失值。为解决这些困扰线性模型的问题，我们提出了一种新颖的贝叶斯多重共线性特征选择（BMFS）模型。随后，我们从不同角度（包括敏感度和显著性）为每个候选原因提供归因分数。最后，当存在多个告警关键绩效指标，且每个指标分别归因于不同的根因集合时，我们会合并结果。我们将整体模型命名为 BALANCE（用于根因定位的贝叶斯线性归因）。

我们想指出，多维根因分析和基于图的根因分析都可以从归因的角度进行表述。具体而言，我们可以将多维根因分析视为把关键绩效指标中的异常归因于其属性组合。由此可知，多维根因分析中的加性约束可以移除，因此在这种情况下，我们只需考虑异常属性。另一方面，将图中的所有异常节点视为候选原因后，即使图结构不可用或无法可靠地学习（实践中这种情况很常见），BALANCE 也能高效地识别根因。换言之，由于线性回归模型常用于因果发现 [49]，BALANCE 也可以作为构建因果图的基础模块。给定图结构后，BALANCE 是随机游走的一种更佳替代方案，因为它无需进行大量随机游走，因此效率更高。

我们在四个数据集上验证了 BALANCE 的有效性。首先，我们生成具有不同数量输入特征、不同程度多重共线性、噪声和稀疏性，以及不同比例缺失值的合成数据，然后比较各种前向模型，包括所提 BMFS、Lasso、E-Net（弹性网络）和 ARD（自动相关性判定）。我们发现，BMFS 通常能够以相当甚至更短的运行时间，最准确地恢复底层真实回归系数，尤其是在输入特征之间存在多重共线性时。此外，我们利用所提方法解决了三个现实世界的根因分析问题。在第一个问题中，我们处理此前提到的异常 SQL 定位问题。结果表明，所提方法能够在每个案例少于 2 秒的时间内识别人类标注的根因 SQL，准确率最高可达 83.3%；相比之下，站点可靠性工程师平均需要 3 分钟。第二个应用处理容器故障定位问题，其目标是将容器中的异常追踪失败归因于容器的指标，例如 CPU 使用率、内存使用率、TCP 等，并促进自愈过程。所提方法能够达到 F_1-分数 0.86，至少比其他基线方法高 20%。最后，我们将 BAL-ANCE 应用于公开数据集 Exathlon [12], ，用于故障类型诊断，所得准确率再次高出

最先进方法 6%。需要注意的是，第一个应用处理的是同质的关键绩效指标和候选项，而后两个应用处理的是异质对象。BALANCE 在这三个场景中的优异表现表明，基于归因的根因分析可以成为应对一般性和实际根因分析案例的高效有效工具。

总而言之，我们的贡献包括：

- 据我们所知，我们是最早从可解释人工智能领域发展出的归因分析视角来表述根因分析问题的研究者之一。具体而言，我们通过将目标关键绩效指标中的异常归因于候选项根因的行为，解释该异常。
- 我们提出了一种新颖的前向模型 BMFS，能够在同时考虑候选项之间相关性的情况下，自动选择相关候选项。
- 我们将所提出的 BALANCE 方法应用于三个现实世界问题，包括异常 SQL 定位、容器故障定位，以及 Exathlon 的故障类型诊断。三个场景均展示了 BALANCE 的有效性。此外，我们还将 BALANCE 部署到生产环境中，用于前两个应用。

2 相关工作

由于所提出的模型与根因分析、归因和稀疏线性模型相关，下面将分别对它们进行简要介绍。

2.1 根因定位

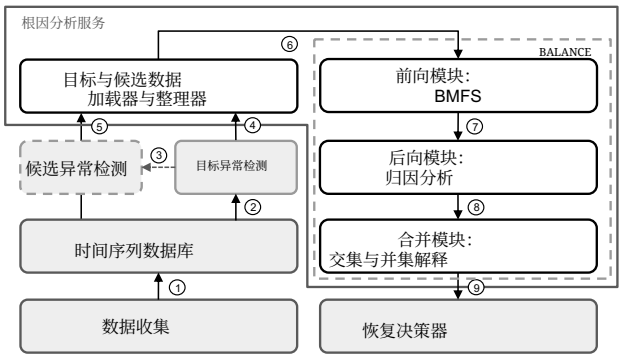
如前所述，根因定位大致有两种策略。第一种考虑多维根因定位方法，例如 Adtributor [5], Hotspot [32], 和 HALO [47]。具体而言，Adtributor [5] 通过选择最异常的维度值，在每个维度中查找根因。然而，Adtributor 假设根因位于单一维度中，这在实践中可能过于严格。为了将 Adtributor 扩展到多维根因的情形，Hotspot [32] 通过涟漪效应将关键绩效指标的异常传播到不同维度，并进一步通过用预测值替换不同维度组合的真实值，计算替换前后的差异，从而定义这些维度组合的归因分数。不幸的是，由此产生的搜索空间通常非常庞大。尽管 Hotspot 采用蒙特卡洛树搜索和分层剪枝策略来降低计算成本，但对于大规模实际问题而言，其速度仍可能过慢。Adtributor 和 Hotspot 的另一个缺点是，它们未能考虑维度之间可能存在的依赖关系。作为解决方案，HALO [47] 通过条件熵学习维度的分层依赖结构，然后在该结构中遍历查找根因。这类方法可以视为归因方法的特例，因为它们试图将关键绩效指标中的异常归因于相关多维属性的组合。需要注意的是，在这种情况下，目标关键绩效指标是沿每个维度的属性值之和。然而，在更一般的情形下，目标关键绩效指标可能以非加性方式受到根因候选项的影响。所提出的 BALANCE 方法为解决这一问题提供了一种方案。

第二种策略侧重于基于图的因果推断 [29]。MonitorRank [14] 和 Microhecl [17] 首次在分布式追踪系统提供的已知服务链架构上，引入了服务级根因分析，另一方面，对于没有已知图结构的应用，CauseInfer [10]和 CloudRanger [39]使用 PC算法 [30]构建因果图。获得图后，通常通过个性化 PageRank [40], 、广度优先搜索 [10, 17], 、随机游走 [14, 39], 等方法推断统计根因。为了使图更加可靠，OM Graph [24]在构建因果图时，考虑了来自知识图谱的先验知识，其中的实体代表分布式数据系统中的所有软硬件。此外，CloudRCA [48]通过基于多种数据源构建图来优化 OM Graph，这些数据源包括监控指标、日志和专家知识。需要注意的是，OM Graph 和 CloudRCA 仍然都使用 PC算法来构建图。如§1所述，由于整个架构具有大规模且复杂的特性，基于服务调用追踪或其他先验知识构建图可能会消耗大量资源；而鉴于异常期间指标时间序列的长度有限，利用因果发现算法（例如 PC算法）构建图又不够可靠。这些缺点阻碍了基于图的根因分析在实践中的应用。另一条研究路线则试图通过识别不同指标之间的滞后时间依赖来构建因果图 [3, 20]。例如，[1, 34] 采用格兰杰因果关系来推断因果依赖关系。然而，只有当监控指标的粒度细化到毫秒或秒级时，滞后时间依赖才会在云系统中出现 [1], ，而搭建如此细粒度的监控系统成本相当高。相比之下，即使图结构不可用，BALANCE 仍然能够发挥作用，并且还可以反过来辅助图的构建及后续的定位步骤。

2.2 归因方法

由于我们使用归因方法来解决根因分析问题，因此本节将回顾一些最先进（SOTA）的归因方法。归因方法旨在理解和解释模型为何做出某一特定预测，从而帮助赢得用户信任，并进一步深入了解如何提升模型性能。一般而言，归因方法可分为基于梯度的方法和基于扰动的方法。基于梯度的方法通过利用模型的梯度来计算归因值。该方向最初的尝试是计算模型目标输出相对于（with regard to）输入的绝对梯度，这也称为敏感性分析 [28]。然而，敏感性分析通常噪声较大，并且丢失了输入变化方向的信息。一种颇具吸引力的解决方案是将梯度与输入进行逐元素乘法（即，梯度×输入），以提高归因图的清晰度 [27]。对于高度非线性模型而言，上述对梯度信息的朴素使用存在一个主要缺陷：它只能提供给定输入邻域内函数局部行为的信息。为了解决这一问题，人们提出了许多方法，包括逐层相关性传播（LRP） [4], 、积分梯度 [33],和 DeepLIFT [27]。另一方面，基于扰动的方法通过移除或改变输入特征，再测量添加扰动前后的输出差异，来获得该输入特征的归因。特征遮挡 [44] 会直接

依次移除每个特征，因此需要进行 p 次模型评估，其中 p 是特征数量。为了减轻沉重的计算负担，局部可解释模型无关解释（LIME） [25] 采用分组特征遮挡。随后利用这些分组拟合局部 Lasso回归，并将所得系数视为归因值。遗憾的是，特征遮挡和 LIME 都存在一个缺陷，即对模型中超参数的选择高度敏感。换言之，这些方法可能以不可预测的方式受到用户定义参数的影响，并且无法保证解释具有无偏性和忠实性。解决这一问题的一种方法是采用 SHAP [18], , 利用经典的 Shapley值为合作博弈中的参与者分配贡献。事实证明，Shapley值是唯一符合一致性要求的归因方法，并具有若干与人类直觉相符的独特公理 [31]。然而，其计算复杂度高达 $O(2^p)$，令人望而却步。这推动了近期旨在降低复杂度的研究 [37]。在 BALANCE 中，我们将候选根因与目标关键绩效指标之间的模型指定为线性模型。因此，几乎所有上述归因方法都可以统一起来 [2], , 从而使所提出的模型具备多种良好性质，后续章节将对此进行讨论。



① 将数据收集并聚合到时间序列数据库中
② 对目标执行实时异常检测（异常检测）
③ 对候选项触发异常检测，以减少其数量（可选）
④ 加载异常目标
⑤ 加载与目标对应的（异常）候选项
⑥ 训练前向BMFS模型
⑦ 计算BMFS选定候选项的归因
⑧ 合并并排序由多个目标产生的根因
⑨ 将根因发送给恢复决策器

图 1：BALANCE 的总体框架。

2.3 用于特征筛选的稀疏线性模型

由于我们在 BALANCE 中采用了稀疏线性前向模型，本节将简要回顾相关文献。最常见的稀疏线性模型是 Lasso [35]，其旨在寻找能够最佳描述观测特征与输出之间线性关系的系数，同时通过 ℓ_1-范数惩罚项使系数保持稀疏。将 Lasso 直接应用于根因分析面临三个问题：首先，候选原因之间通常存在相关性，但由于 ℓ_1 范数的性质，Lasso在处理相关特征方面表现不佳 [50]。其次， ℓ_1 范数前的调节参数在实践中是未知的，该参数用于平衡数据保真度与系数稀疏性之间的权衡，通常通过交叉验证网格搜索来选择。因此，对于每个数据分区，Lasso算法都必须针对调节参数的每个候选值运行一次，而每个数据分区的

数据集，从而带来沉重的计算负担。第三，缺失值是常态而非例外，但 Lasso 无法直接处理缺失值。作为对第一个问题的补救措施，E-Net（弹性网络）通过对系数结合 ℓ_1 范数和 ℓ_2 范数来提出，从而为相关系数 [50]引入了分组效应。这一优点伴随着额外的调节参数和显著的计算开销。另一方面，另外两个问题可以通过贝叶斯模型加以解决，例如 ARD（自动相关性判定） [36]。调节参数被假设为随机变量，其后验分布可以通过期望最大化从数据中推断得到。遗憾的是，该模型未考虑特征之间的相关性。随后，研究者进一步提出了一些方法，通过借助相关收缩先验的力量来缓解这一缺陷，例如组逆伽马-伽马先验 [7] 和相关尖峰与平板先验 [26]。然而，这类方法需要一些实际中无法获得的先验信息，例如

分组信息 [7]或零系数的先验分布 [26]。在本文中，我们提出 BMFS 以克服上述问题，具体内容将在第 §4.1 节中详细讨论。

3 问题构建

本节首先介绍 BALANCE 的整体框架。随后，我们深入探讨根因分析部分，提出一些期望，并讨论这些期望如何促成所提方法。

BALANCE 的整体框架如图 1 所示。我们首先收集原始数据，包括目标关键绩效指标和候选根因，并将其存储在时间序列数据库中。具体而言，令 $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \ \boldsymbol{x}_p]$ 表示与目标关键绩效指标或派生指标 $\boldsymbol{y}$（例如租户关键绩效指标）相关联的 p 候选根因或基础指标（例如每个 SQL 对应的指标）。由于监控所有基础指标（即候选根因） $\boldsymbol{X}$ 通常会消耗过多资源，因此仅采用实时异常检测来监控目标关键绩效指标 $\boldsymbol{y}$。尽管不是实时的，但对 $\boldsymbol{X}$ 的异常检测可以在 $\boldsymbol{y}$ 上触发告警后启动，因为在根因分析期间，我们只需关注异常的 $\boldsymbol{X}$。随后，我们从数据库中收集告警发生前及告警期间、长度为 n 的异常 $\boldsymbol{y}$ 和 $\boldsymbol{X}$ 数据，并将其输入所提根因分析服务 BALANCE，以查找根因。得到的估计根因将作为恢复决策器的输入，由其生成自愈方案。

理想情况下，我们希望根因分析模块能够满足以下期望：

d1。观测值的数量（即时间序列的长度）n 通常较少，因为我们只考虑异常发生之前及期间的短时间序列，以解释目标对象的异常行为 $\boldsymbol{y}$。*d2*。候选项的数量 p 不固定，并且在每种情况下都可能很多。因此，根因分析模块应具备足够的灵活性，以处理 $p \leq n$ 和 $p > n$的情况。后一种情形更具挑战性，在实践中可以发现，

例如，在异常SQL示例中，正在运行的SQL数量会随时间变化，可能达到数百甚至数千条，而 $n = 61$。在这种情况下， $p > n$。

d3。根因分析模块应输出所有可能的根因，即使它们彼此相关，同时还应移除所有不相关的候选项。在实践中，由于候选项之间的交互通常随时间变化且具有双向性，因此枚举并验证候选项之间所有可能的因果关系是不可行的。因此，更好的做法是列出所有相关的根因，为站点可靠性工程师提供更多信息。*d4*。根因分析模块应能够处理监控系统中经常存在的缺失数据。*d5*。根因分析过程应当高效。

d6. 结果应当具有可解释性，且候选项的重要性应具有可比性，以便用于后续排序和决策。

另一方面，归因方法旨在通过为模型的每个输入特征 j分配一个标量归因值 r_j来解释模型的输出行为，该值有时也称为“相关性”“特征重要性”或“贡献度”。形式化地说，给定目标输出 $\boldsymbol{y}$，归因方法的目标是确定每个输入特征 $\boldsymbol{x}_j \in \boldsymbol{X}$对输出 $\boldsymbol{y}$ 的贡献 $\boldsymbol{r} = [r_1, \cdots \ r_p] \in \mathbb{R}^p$。因此，利用归因方法进行根因分析是很自然的。在归因框架下，首先构建一个前向模型 $\boldsymbol{y} = f(\boldsymbol{X})$，用于根据 $\boldsymbol{X}$预测 $\boldsymbol{y}$。然后使用归因方法解释目标关键绩效指标 $\boldsymbol{y}$ 的异常行为，将其归因于候选根因 $\boldsymbol{X}$。在本文中，我们进一步采用贝叶斯稀疏线性模型作为前向模型，并特别关注候选项之间的相关性。我们希望从以下三个角度说明采用前向模型的动机：

- 首先，所提出的前向模型能够很好地满足上述所有要求。具体而言，得益于线性模型的高效性，每当目标关键绩效指标发出告警时，我们都会针对数据拟合一个不同的线性模型，从而成功解决候选项数量随具体情况变化的问题（即 *d2*）。通过利用回归系数的稀疏性，稀疏线性模型能够自动选择与目标相关的候选项，并进一步处理大-*p*-小-*n* 问题（即 *d1, d2*）。在进一步考虑候选项之间的多重共线关系后，可以满足 *d3*。所提出模型中的贝叶斯框架通过在推断其余参数的同时推断缺失数据的分布，便于处理缺失数据（即 *d4*）。通过学习调优参数的变分后验分布，而不是通过网格搜索估计这些参数，所提出的方法通常比常用的频率学派对应方法更高效（即*d5*）。最后，线性模型非常适用于归因方法，具体讨论将在下文展开（即*d6*）。

- 其次，归因方法得出的结果（即估计的根因）应忠实反映站点可靠性工程师试图理解的底层过程。为此，第 2.2 节所述的方法通常采用复杂的前向模型（即神经网络）来预测 $\boldsymbol{y}$ 给定 $\boldsymbol{X}$，以最大化预测准确率。另一方面，由于前向模型具有黑箱性质，还需进一步应用另一个可解释模型来近似前向模型所学到的内容，从而最大化描述准确率。例如，LIME [25]和 SHAP [18] 都采用局部线性解释模型，而 DeepLIFT [27] 则将神经网络的非线性组件线性化。在根因分析场景下，由于 *d2*、*d1*和 *d5*，训练神经网络作为前向模型是不可行的。此外，我们的目标并不是预测 $\boldsymbol{y}$ 给定 $\boldsymbol{X}$，因此预测准确率并非我们的主要关注点。相反，我们更加关注描述误差。通过将前向模型指定为线性模型，我们可以将描述误差降至零。

- 最后，如§2.2所述，Shapley值被认为是唯一满足所有符合人类直觉的公理的归因方法； [2]中已证明，当应用于线性模型且使用零基线时，几乎所有上述归因方法，包括梯度×输入、积分梯度、DeepLIFT和特征遮挡，都会生成精确的Shapley值。我们将在§4.2中讨论所提出的线性模型如何满足所有公理。

构建前向模型后，我们通过找出对 $\boldsymbol{y}$的异常行为贡献最大的 $\boldsymbol{X}$子集，将 $\boldsymbol{y}$中的异常归因于候选项 $\boldsymbol{X}$。最后，我们根据归因分数对根因进行排序；当存在多个目标关键绩效指标时，再合并结果。

4 方法

本节将分别详细介绍BALANCE中的三个模块。

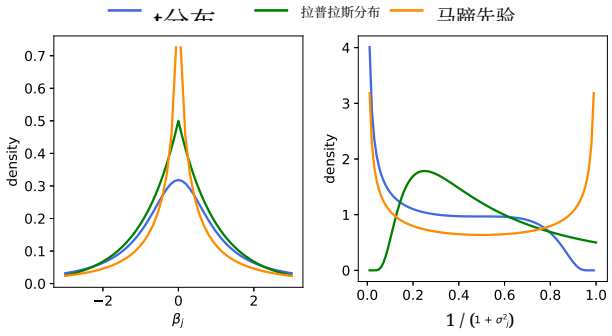
4.1 前向模块：贝叶斯多重共线特征选择

4.1.1 贝叶斯表述。 如§3所述，我们选择稀疏线性模型作为前向模型，即 $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$，其中 $\boldsymbol{y} \in \mathbb{R}^n$ 表示异常附近单个目标关键绩效指标的长度为 n 的时间序列， $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ 表示相应 p 候选根因的时间序列， $\boldsymbol{\beta} \in \mathbb{R}^p$ 为稀疏系数向量，而 $\boldsymbol{\epsilon} \sim \mathrm{N}(0, \alpha^{-1}\boldsymbol{I})$表示方差为 α^{-1}的独立高斯噪声。等价地，我们可以从概率角度对该问题进行表述：

$$p(\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{X}, \alpha) = \mathcal{N}(\boldsymbol{X}\boldsymbol{\beta}, \alpha^{-1}\boldsymbol{I}) \propto \exp\left(\frac{\alpha}{2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\right). \quad (1)$$

我们假设 α服从非信息性的杰弗里斯先验，即 $p(\alpha) \propto 1/\alpha$。

现在让我们聚焦于 $\boldsymbol{\beta}$ 的先验分布。我们的目标是同时促进稀疏性，并处理特征 $\boldsymbol{X}$ 之间的多重共线性。为促进稀疏性，通常会采用拉普拉斯分布（即 ℓ_1-范数的贝叶斯对应形式）或学生 t 分布。然而，这里我们选择马蹄先验 [9]，因为与拉普拉斯先验和学生 t 先验相比，它具有一定优势。马蹄先验可以表示为



(a) 促进稀疏性的先验 (b) 收缩权重密度

图 2：常用促进稀疏性的先验的密度，包括 Student's T分布、拉普拉斯分布和马蹄先验 (a)，以及相应收缩权重的密度 $1/(1+\sigma^2)$ (b)。

高斯尺度混合：

$$p(\beta_j|\sigma_j, g) = \mathcal{N}(\mathbf{0}, g\sigma_j^2), \tag{2}$$

$$p(\sigma_j) = C^+(0,1), \tag{3}$$

其中， $C^+(0,1)$ 是定义在正实数 $\mathbb{R}^+$上的标准半柯西分布， g和 σ_j 分别是全局收缩参数和局部收缩参数。当 g 变小时，全局收缩参数 g会将 $\boldsymbol{\beta}$中的所有系数 β_j收缩至零。另一方面，局部收缩参数 σ_j 的重尾半柯西先验允许一些 β_j 逃离收缩。因此，最终得到的 $\boldsymbol{\beta}$ 将是稀疏的。与其他常用的促进稀疏性的先验（例如 Student's T分布和拉普拉斯分布）相比，从图 2(a) 可以看出，马蹄先验在 0 和非常大的 β_j处具有更高的密度。换言之，它能够更好地区分 $\boldsymbol{\beta}$中的零元素和非零元素。换一种角度来看，鉴于 σ_j 服从定义在 σ_j上的半柯西先验，我们可以得到 [9]:

$$\frac{1}{1+\sigma_j^2} \sim \text{Be}\left(\frac{1}{2}, \frac{1}{2}\right), \tag{4}$$

其中，Be(0.5,0.5) 表示形状参数为 0.5 的贝塔分布，我们将 $1/(1+\sigma^2)$ 称为收缩权重。如图 2(b) 所示，马蹄先验的收缩权重密度函数（即橙色曲线）呈 U 形（即马蹄形）；它在 0.5 处达到最低值，但在 0 和 1 处无界，这表明该先验倾向于使 σ_j^2 要么非常小，要么非常大。然而，Student's T 分布和拉普拉斯分布先验（见蓝色和绿色曲线）不具备这一优良特性。因此，在处理未知稀疏性和较大的非零元素时，马蹄先验更为稳健。

另一方面，为了处理多重共线性，我们通常会采用 g 先验 [45]:

$$p(\boldsymbol{\beta}|g, \sigma^2, X) = \mathcal{N}\left(\mathbf{0}, g\sigma^2(X^TX)^{-1}\right), \tag{5}$$

其中， g和 σ^2 是决定 $\boldsymbol{\beta}$总体方差的标量，不同特征 $\boldsymbol{X}$之间的相互依赖关系则通过 $\boldsymbol{X}^T\boldsymbol{X}$进行经验刻画。由此得到的 $\boldsymbol{\beta}$ 后验分布在 $\boldsymbol{\beta}$ 中各元素之间呈现出比其他先验分布更强的相关性，例如拉普拉斯分布（即 ℓ_1-范数）和具有对角协方差的高斯分布（即

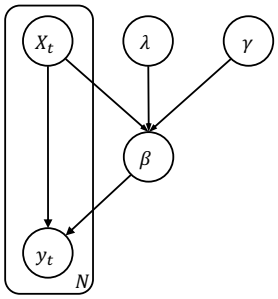


图3：BMFS 的图表示。标记为 N 的方框表示其中子图的重复次数为 N。

ℓ_2-范数）。为兼顾两者的优势，我们提出了一种新颖的先验分布，将马蹄先验与g先验无缝融合：

$$p(\boldsymbol{\beta}|g, \boldsymbol{\sigma}, X) = \mathcal{N}\left(\mathbf{0}, g\,\text{diag}(\boldsymbol{\sigma})\boldsymbol{X}^T\boldsymbol{X}\,\text{diag}(\boldsymbol{\sigma})\right), \tag{6}$$

$$p(\sigma_j) = C^+(0,1), \tag{7}$$

其中， $\boldsymbol{\sigma}$ 是一个 p 维向量。我们将上述先验称为相关马蹄先验。为便于推导变分推断算法，我们进一步利用典型指数族形式 [22] 对上述先验进行重新参数化，得到：

$$\begin{aligned} p(\boldsymbol{\beta}|\gamma, \boldsymbol{\lambda}, X) &= \mathcal{N}\left(\mathbf{0}, \left(\gamma\,\text{diag}\left(\boldsymbol{\lambda}^{\frac{1}{2}}\right)\boldsymbol{X}^T\boldsymbol{X}\,\text{diag}\left(\boldsymbol{\lambda}^{\frac{1}{2}}\right)\right)^{-1}\right) \\ &\propto \exp\left(\frac{p}{2}\log\gamma + \frac{1}{2}\sum_{j=1}^{p}\log\lambda_j \right. \\ &\quad \left. - \frac{\gamma}{2}\boldsymbol{\lambda}^{\frac{T}{2}}\left((\boldsymbol{X}^T\boldsymbol{X})\circ(\boldsymbol{\beta}\boldsymbol{\beta}^T)\right)\boldsymbol{\lambda}^{\frac{1}{2}}\right), \end{aligned} \tag{8}$$

$$p(\lambda_j) = \frac{1}{\pi}\lambda_j^{-\frac{1}{2}}(\lambda_j+1)^{-1}, \quad \forall\lambda_j > 0, \tag{9}$$

其中， $\gamma = 1/g$、 $\lambda_j = 1/\sigma_j^2$、 $\boldsymbol{\lambda}^{1/2}$ 表示向量 $\boldsymbol{\lambda}$ 中各元素的逐元素平方根， $\boldsymbol{\lambda}^T/^2$ 表示 $\boldsymbol{\lambda}^{1/2}$ 的转置，而 $p(\lambda_j)$ 可视为贝塔素分布或复合伽马分布。由于我们对全局收缩参数 γ 没有任何先验知识，因此采用非信息性杰弗里斯先验，即 $p(\gamma) \propto 1/\gamma$。

综上，整体贝叶斯模型可分解为：

$$p(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda}, \alpha, \gamma|\boldsymbol{X}) = p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\beta}, \alpha)p(\boldsymbol{\beta}|\gamma, \boldsymbol{\lambda}, \boldsymbol{X})p(\boldsymbol{\lambda})p(\gamma)p(\alpha). \tag{10}$$

相应的图表示见图3。

4.1.2 变分推断。 我们的总体目标是推断后验分布 $p(\boldsymbol{\beta}, \boldsymbol{\lambda}, \alpha, \gamma|\boldsymbol{y}, \boldsymbol{X})$。然而，无法获得该后验分布的闭式表达式，因此我们遵循变分推断框架，寻找一个与精确后验在 Kullback-Leibler（KL）散度意义下最接近且易于处理的变分分布 $q(\boldsymbol{\beta}, \boldsymbol{\lambda}, \alpha, \gamma)$。为简便起见，我们采用平均场近似，将变分分布分解为：

$$q(\boldsymbol{\beta}, \boldsymbol{\lambda}, \alpha, \gamma) = q(\boldsymbol{\beta})\prod_{j=1}^{p} q(\lambda_j)q(\alpha)q(\gamma). \tag{11}$$

平均场近似的一个优点是，可以通过将 KL 散度关于各因子的泛函导数设为零 [6]，来确定每个因子的函数形式。在获得变分分布的函数形式后，我们进一步通过自然梯度下降 [15, 43] 递归地更新这些分布的参数。值得强调的是，与标准梯度相比，自然梯度通常能够得到更简单的表达式（即每次迭代所需的计算量更少）和更快的收敛速度（即所需的迭代次数更少）[15, 42, 43]。因此，当我们关注算法效率时，自然梯度更受青睐。此外，尽管我们在 Eq-11 中假设各变分分布彼此独立，但在对其进行优化以最小化 KL 散度时，它们的参数会以一种直接的方式相互依赖，具体体现在下文给出的这些参数的更新规则中。换言之，我们仍能在一定程度上捕捉变量 $\boldsymbol{\beta}$、$\boldsymbol{\lambda}$、α 和 γ 之间的相互作用。下面让我们深入推导每个变分分布。

对于 $q(\boldsymbol{\beta})$，当$\beta_j \in \mathbb{R}$ 时，我们可以将其指定为高斯分布；当 $\boldsymbol{\beta}$ 被约束为非负时，则可以指定为对数正态分布。在第一种情况下，使KL散度最小的 $q(\boldsymbol{\beta})$可以更新为：

$$q(\boldsymbol{\beta}; \boldsymbol{h}_\beta, \boldsymbol{J}_\beta) \propto \exp\left(-\frac{1}{2}\boldsymbol{\beta}^T \boldsymbol{J}_\beta \boldsymbol{\beta} + \boldsymbol{h}_\beta^T \boldsymbol{\beta}\right), \tag{12}$$

其中，$\boldsymbol{J}_\beta$和 $\boldsymbol{h}_\beta$ 分别表示 β的精度矩阵（即协方差的逆矩阵）和势向量。由精度矩阵和势向量参数化的高斯分布称为典范形式，这种形式便于采用如下所示的简洁更新规则 [15, 43]。$\boldsymbol{h}_\beta$ 和 $\boldsymbol{J}_\beta$ 称为典范参数或自然参数，随后可以将相应的均值参数计算为$\boldsymbol{\beta} = \boldsymbol{J}_\beta^{-1}\boldsymbol{h}_\beta$以及$\mathrm{Cov}[\boldsymbol{\beta}] = \boldsymbol{J}_\beta^{-1}$。具体而言，$\boldsymbol{J}_\beta$和 $\boldsymbol{h}_\beta$的更新规则可以推导如下：

$$\begin{aligned}\boldsymbol{J}_\beta^{\{\kappa\}} &= (1-\rho)\boldsymbol{J}_\beta^{\{\kappa-1\}} + \rho\Big(\langle\alpha\rangle \boldsymbol{X}^T\boldsymbol{X} \\ &\quad + \langle\gamma\rangle \operatorname{diag}\left(\langle\boldsymbol{\lambda}^{\frac{1}{2}}\rangle\right)\boldsymbol{X}^T\boldsymbol{X}\operatorname{diag}\left(\langle\boldsymbol{\lambda}^{\frac{1}{2}}\rangle\right)\Big), \end{aligned}\tag{13}$$

$$\boldsymbol{h}_\beta^{\{\kappa\}} = (1-\rho)\boldsymbol{h}_\beta^{\{\kappa-1\}} + \rho\langle\alpha\rangle\boldsymbol{X}^T\boldsymbol{y}, \tag{14}$$

其中，·表示相应变分分布下的期望，ρ 表示使用线搜索方法（例如 Armijo规则） [43]确定的步长。为更直观地理解上述推导过程，我们以 $\boldsymbol{h}_\beta$的更新规则为例。在公式-14中，$\boldsymbol{h}_\beta$的自然梯度为$\alpha\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{h}_\beta$。当我们以步长 ρ沿自然梯度方向更新 $\boldsymbol{h}_\beta$时，可以得到公式-14。$\boldsymbol{J}_\beta$的更新规则以相同方式推导，后续的更新规则也同理。

另一方面，当我们将 $\boldsymbol{\beta}$ 限制为非负值，并使用对数正态分布作为变分分布时，我们进一步将 $q(\boldsymbol{\beta})$ 分解为 $\prod_j q(\beta_j)$，而 $q(\beta_j)$ 可写为：

$$q(\beta_j; h_{\beta_j}, \zeta_{\beta_j}) \propto \frac{1}{\beta_j}\sqrt{\zeta_{\beta_j}}\exp\left(-\frac{1}{2}\zeta_{\beta_j}\beta_j^2 + h_{\beta_j}\beta_j\right), \tag{15}$$

其中，在每次迭代中，κ 自然参数 $\boldsymbol{h}_\beta$ 和 $\boldsymbol{\zeta}_\beta$ 可更新为：

$$\begin{aligned}\boldsymbol{h}_\beta^{\{\kappa\}} &= (1-\rho)\boldsymbol{h}_\beta^{\{\kappa-1\}} + \rho\Big(-\boldsymbol{c}_1 \circ \left(1 - 2\langle\log\boldsymbol{\beta}\rangle\right) \\ &\quad + \boldsymbol{c}_2 \circ \left(1 - \langle\log\boldsymbol{\beta}\rangle\right) + 1\Big), \end{aligned}\tag{16}$$

$$\boldsymbol{\zeta}_\beta^{\{\kappa\}} = (1-\rho)\boldsymbol{\zeta}_\beta^{\{\kappa-1\}} + \rho(2\boldsymbol{c}_1 - \boldsymbol{c}_2), \tag{17}$$

$$\begin{aligned}\boldsymbol{c}_1 = \operatorname{diag}\Big(&\langle\alpha\rangle\boldsymbol{X}^T\boldsymbol{X} \\ &+ \langle\gamma\rangle \operatorname{diag}\left(\langle\boldsymbol{\lambda}^{\frac{1}{2}}\rangle\right)\boldsymbol{X}^T\boldsymbol{X}\operatorname{diag}\left(\langle\boldsymbol{\lambda}^{\frac{1}{2}}\rangle\right)\Big) \circ \langle\boldsymbol{\beta}^2\rangle, \end{aligned}\tag{18}$$

$$\begin{aligned}\boldsymbol{c}_2 = \langle\alpha\rangle\boldsymbol{X}^T\boldsymbol{y} - \text{off-diag}\Big(&\langle\alpha\rangle\boldsymbol{X}^T\boldsymbol{X} \\ &+ \langle\gamma\rangle \operatorname{diag}\left(\langle\boldsymbol{\lambda}^{\frac{1}{2}}\rangle\right)\boldsymbol{X}^T\boldsymbol{X}\operatorname{diag}\left(\langle\boldsymbol{\lambda}^{\frac{1}{2}}\rangle\right)\Big) \circ \langle\boldsymbol{\beta}\rangle, \end{aligned}\tag{19}$$

其中，$\circ$表示 Hadamard（或逐元素）乘积，$\boldsymbol{\beta}^2$ 表示 $\boldsymbol{\beta}$的逐元素平方，off-diag(·)表示矩阵的非对角部分，即将对角线替换为零向量。随后，我们可以计算均值参数：$\log\boldsymbol{\beta} = \boldsymbol{h}_\beta \oslash \boldsymbol{\zeta}_\beta$、$\mathrm{Var}[\log\boldsymbol{\beta}] = 1 \oslash \boldsymbol{\zeta}_\beta$、$\boldsymbol{\beta} = \exp(\log\boldsymbol{\beta} + \mathrm{Var}[\log\boldsymbol{\beta}]/2)$以及$\boldsymbol{\beta}^2 = \exp(2\log\boldsymbol{\beta} + 2\mathrm{Var}[\log\boldsymbol{\beta}])$，其中 $\oslash$ 表示逐元素除法。

对于 $q(\lambda_j)$，我们遵循 [22] 并将其函数形式指定为：

$$q(\lambda_j; d_j) = \frac{1}{E_1(d_j)}(\lambda_j + 1)^{-1}\exp\left(-d_j(\lambda_j + 1)\right), \tag{20}$$

其中，$E_1(x) = \int_x^\infty exp(-t)/t dt$ 表示指数积分函数。因此，$\boldsymbol{d}$ 的更新规则为：

$$\begin{aligned}\boldsymbol{d}^{\{\kappa\}} &= (1-\rho)\boldsymbol{d}^{\{\kappa-1\}} \\ &\quad + \rho\langle\gamma\rangle\Big(\text{off-diag}\left(\boldsymbol{X}^T\boldsymbol{X} \circ \langle\boldsymbol{\beta}\boldsymbol{\beta}^T\rangle\right)\langle\boldsymbol{\lambda}^{\frac{1}{2}}\rangle \circ \boldsymbol{c}_3 \\ &\quad + \frac{1}{2}\operatorname{diag}\left(\boldsymbol{X}^T\boldsymbol{X} \circ \langle\boldsymbol{\beta}\boldsymbol{\beta}^T\rangle\right)\Big), \end{aligned}\tag{21}$$

$$\boldsymbol{c}_3 = \left(\langle\boldsymbol{\lambda}^{\frac{1}{2}}\rangle \circ \boldsymbol{d} - \Gamma(1.5) \circ \boldsymbol{d}^{\frac{1}{2}}\right) \oslash \left(\langle\boldsymbol{\lambda}\rangle \circ \boldsymbol{d} - 1\right). \tag{22}$$

均值参数 $\boldsymbol{\lambda}$ 和 $\boldsymbol{\lambda}^{\frac{1}{2}}$可计算如下：[1]

$$\langle\boldsymbol{\lambda}\rangle = \Gamma(-1, \boldsymbol{d}) \oslash \Gamma(0, \boldsymbol{d}), \tag{23}$$

$$\langle\boldsymbol{\lambda}^{\frac{1}{2}}\rangle = \Gamma(1.5)\Gamma(-0.5, \boldsymbol{d}) \oslash \Gamma(0, \boldsymbol{d}), \tag{24}$$

其中，Γ 表示输入只有一个时的伽马函数，而[1] 表示输入有两个时的上不完全伽马函数；步长 ρ 仍由 Armijo规则确定。

对于$q(\alpha)$和 $q(\gamma)$，我们将其指定为伽马分布，相应的更新规则为：

$$q(\alpha; a_\alpha, b_\alpha) \propto \alpha^{a_\alpha - 1}\exp(-b_\alpha\alpha), \tag{25}$$

$$q(\gamma; a_\gamma, b_\gamma) \propto \gamma^{a_\gamma - 1}\exp(-b_\gamma\alpha), \tag{26}$$

[1] $\Gamma(c, d) = U(1-c, 1-c, d)/\exp(d)$，其中 U表示 Tricomi 合流超几何函数。

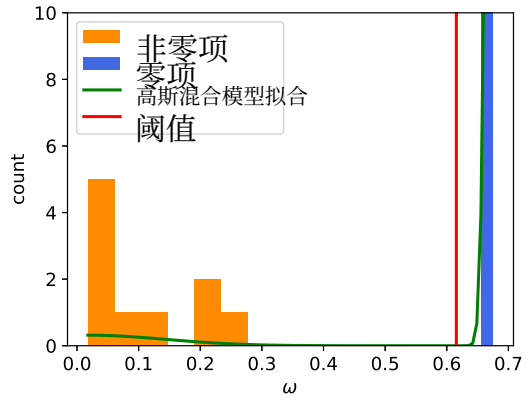


图 4：合成数据集中 ω 的经验分布：橙色和蓝色柱分别表示 ω 中相对于$\langle\beta\rangle$ 非零项和零项的直方图，绿色曲线表示对经验分布拟合高斯混合模型（GMM）后得到的密度，红色曲线表示所选阈值$\hat{}\omega$。

其中

$$a_\alpha^{\{\kappa\}} = (1-\rho)a_\alpha^{\{\kappa-1\}} + \frac{\rho n}{2}, \tag{27}$$

$$b_\alpha^{\{\kappa\}} = (1-\rho)b_\alpha^{\{\kappa-1\}} + \frac{\rho}{2}\left(\boldsymbol{y}^T\boldsymbol{y} + \mathrm{tr}(\langle\boldsymbol{\beta}\boldsymbol{\beta}^T\rangle\boldsymbol{X}^T\boldsymbol{X}) - 2\boldsymbol{y}^T\boldsymbol{X}\langle\boldsymbol{\beta}\rangle\right), \tag{28}$$

$$a_\gamma^{\{\kappa\}} = (1-\rho)a_\gamma^{\{\kappa-1\}} + \frac{\rho p}{2}, \tag{29}$$

$$b_\gamma^{\{\kappa\}} = (1-\rho)b_\gamma^{\{\kappa-1\}} + \frac{\rho}{2}\Big(\langle\boldsymbol{\lambda}\rangle^T \,\mathrm{diag}\left(\boldsymbol{X}^T\boldsymbol{X}\circ\langle\boldsymbol{\beta}\boldsymbol{\beta}^T\rangle\right) + \langle\boldsymbol{\lambda}^{\frac{1}{2}}\rangle^T \,\text{off-diag}\left(\boldsymbol{X}^T\boldsymbol{X}\circ\langle\boldsymbol{\beta}\boldsymbol{\beta}^T\rangle\right)\langle\boldsymbol{\lambda}^{\frac{1}{2}}\rangle\Big). \tag{30}$$

在实践中，$\boldsymbol{y}$ 和 $\boldsymbol{X}$ 中的条目通常会缺失。在这种情况下，我们通过最小化上述KL散度，为$X_{i,\ j}$ 生成点估计，并为 y_i 生成贝叶斯估计，即:

$$\hat{X}_{i,j} = \langle\boldsymbol{\beta}_j\boldsymbol{\beta}_j^T\rangle^{-1}\langle\boldsymbol{\beta}_j\rangle y_i, \tag{31}$$

$$q(y_i) = \mathcal{N}\left(\boldsymbol{X}_{i,:}\langle\boldsymbol{\beta}\rangle, \langle\alpha\rangle^{-1}\right), \tag{32}$$

其中，向量 $\boldsymbol{j}$ 表示 $\boldsymbol{X}$中第 i 行缺失值的索引，而 $\boldsymbol{X}_{i,:}$ 表示 $\boldsymbol{X}$ 的第 i 行。

4.1.3 软阈值化。 回想一下，σ_j上的半柯西先验会产生收缩权重 $1/(1+\sigma_j^2)$ 的 U形先验（参见式（4））。由于 $\lambda_j = 1/\sigma_j^2$，收缩权重也可以等价地表示为 $\lambda_j/(1+\lambda_j)$。这一 U形先验促使 λ_j 要么非常小，要么非常大，从而能够自动区分 $\boldsymbol{\beta}$ 中的零项和非零项。得益于这一先验，我们发现，对于所有 j，经验分布 $\hat{}\,p(\omega)$ 在 $\lambda_j/(\lambda_j+1)$ 上的密度也近似呈 U形，其中期望 λ_j 是对变分后验分布 $q(\lambda_j)$ 取的，如图 4 所示。正如预期的那样，对于大多数真实非零项，ω 的值接近于零，且其密度随 ω 增大而降低；而对于大多数真实零项，ω 的值则远离零。

为区分非零项与零项，我们将阈值设为 $\hat{}\omega = \arg\min \hat{}\,p(\omega)$，即 U形密度的谷值，并设置 $\beta_j = 0 \,\text{if}\, \lambda_j/(\lambda_j+1) > \hat{}\omega$。具体而言，

算法 1 贝叶斯多重共线性特征选择（BMFS）

输入： 一个已发出告警的目标关键绩效指标 $\boldsymbol{y}$ 以及与之关联的候选根因 $\boldsymbol{X}$ 及 $\boldsymbol{y}$;
输出： 回归系数 $\boldsymbol{\beta}$;
初始化所有变分分布的参数;
重复
  如果 $\boldsymbol{\beta}$ 只能取正值**那么**
    按照公式-15更新 $q(\boldsymbol{\beta})$;
  else
    按照公式-12更新 $q(\boldsymbol{\beta})$;
  结束如果
  计算$\boldsymbol{\beta}$和Cov$[\boldsymbol{\beta}]$，给定 $q(\boldsymbol{\beta})$;
  按照公式-20更新 $q(\boldsymbol{\lambda})$，并按照公式-23计算$\boldsymbol{\lambda}$，按照公式-24计算$\boldsymbol{\lambda}^{\frac{1}{2}}$;
  根据公式 25 更新 $q(\alpha)$，根据公式 26 更新 $q(\gamma)$，计算它们的期望;
  如果 $\boldsymbol{X}$ 和 $\boldsymbol{y}$ 中存在缺失值**则**
    按照公式 31 和公式 32 填补缺失值;
  结束如果
直到收敛;
执行第 §4.1.3 节中介绍的软阈值化;

我们对 $\hat{}\,p(\omega)$ 拟合一个双成分高斯混合模型（GMM）：两个高斯分布的均值分别固定为 ω 在所有 j 中的最小值和最大值，而方差和权重则根据数据进行学习。然后，我们找到对应于 GMM 密度最小值的 $\hat{}\omega$。

我们在算法 1 中总结了 BMFS 的整个流程。

4.2 后向模块：归因分析

由于我们的前向模型是线性模型，一个看似颇具吸引力的归因分数选择是回归系数，即

$$r_j = |\langle\beta_j\rangle|, \tag{33}$$

其中，r_j表示归因分数，$|\cdot|$表示绝对值。事实上，对于线性模型，回归系数 β_j 等于 $\boldsymbol{y}$ 关于 $\boldsymbol{x}_j$ 的梯度，因此它描述了 $\boldsymbol{y}$ 对 $\boldsymbol{x}_j$ 的敏感度。值 β_j 量化了 $\boldsymbol{x}_j$ 的微小变化对 $\boldsymbol{y}$ 的影响。请注意，我们得到的是 β_j 的分布，而非点估计。遵循贝叶斯模型可解释人工智能（XAI）的框架 [8]，本节使用 β_j 的均值来计算归因分数。请注意，也可以使用其他值，例如分位数、众数，以及当存在多个众数时，众数的交集或并集。由于 $q(\beta_j)$ 服从高斯分布，因此均值是合适的选择。

遗憾的是，高敏感度并不表示对目标关键绩效指标中异常的贡献很高。例如，假设目标关键绩效指标是候选根因的汇总或聚合结果，且β_j非零且相对较大，但$\boldsymbol{x}_j$较小。我们通常会忽略$\boldsymbol{x}_j$，因为其尺度较小，无法对整体目标产生太大贡献。为此，我们采用梯度×输入方法，在该方法中，归因分数可计算为:

$$r_j = |\langle\beta_j\rangle x_j|. \tag{34}$$

在归因文献中，该值被称为显著性 [27]。现在，仅当β_j和$\boldsymbol{x}_j$本身都相对较大时，才会选取候选根因。

然而，我们的最终目标是将目标关键绩效指标中的异常归因于候选根因，因此，我们关注的并不是x_j的绝对值。相反，我们旨在通过$\boldsymbol{x}_j$的变化来解释 $\boldsymbol{y}$在异常期间的变化。换言之，我们关注的是某个候选项的边际效应，并希望了解：将候选项中的异常部分替换为正常部分后，目标会如何变化。回顾一下，在我们的案例中，异常或告警时间是给定的。因此，我们将时间序列正常部分的均值或中位数作为基线，并计算异常与基线之间的差值 Δx_j。由此得到的归因分数定义为：

$$r_j = |\langle\beta_j\rangle \Delta x_j|. \tag{35}$$

为进一步确保归因分数不受 $\boldsymbol{y}$尺度影响，我们最终将归因分数计算如下：

$$r_j = \left|\frac{\langle\beta_j\rangle \Delta x_j}{\Delta y}\right|. \tag{36}$$

值得强调的是，由于采用了所提出的前向模型，上述归因分数满足 Shapley 值的所有理想公理，包括完备性、虚设参与者、对称性、线性和连续性 [2]。下面我们简要介绍这些公理在根因分析中的含义，并讨论所提出的模型如何满足这些公理。

- 当归因之和等于 $\boldsymbol{y}$ 在正常时期与异常时期的取值之差时，完备性便得到满足。显然，式 36 满足这一要求。
- 虚设参与者：如果目标 $\boldsymbol{y}$取决于候选项$\boldsymbol{x}_j$，那么 $r_j = 0$。所提出的相关马蹄先验会自动排除无关候选项$\boldsymbol{x}_j$，并将对应的β_j设为 $= 0$。因此， $r_j = 0$。
- 对称性：如果目标 $\boldsymbol{y}$取决于两个候选项 $\boldsymbol{x}_j$和 $\boldsymbol{x}_k$，但与它们的顺序无关，那么 $r_j = r_k$。在我们的前向模型中考虑多重共线性，即可满足这条公理。
- 线性：线性模型天然满足这一性质。
- 连续性：对于两个几乎相同的输入，生成的归因也应几乎相同。通过在我们的前向模型中捕捉多重共线性，也可以满足这条公理。

此外，显然上述公理与 §3 中提到的理想根因分析（RCA）的要求一致。这进一步强化了我们的看法：归因分析天然适用于根因分析。

4.3 合并模块：交集与并集解释

在本小节中，我们将注意力转向存在多个目标关键绩效指标的情况。在这种情况下，我们首先按照上述两个小节中的两个步骤，计算每个目标关键绩效指标对应的每个候选根因的归因分数 r_{jk}。随后，对于每个 k，我们将跨 j的归因分数 r_{jk}按降序排序，并将此步骤称为排序。接着，我们保留每个目标关键绩效指标的前 κ个根因 $\boldsymbol{y}_k$。请注意，当 $\boldsymbol{\beta}$足够稀疏时（即 $\boldsymbol{\beta}$中的非零项数量小于 κ），可以省略筛选步骤。在完成排序和筛选后，我们通过对根因集合执行交集或并集操作来合并这些集合。交集操作表示仅当某个根因影响所有目标关键绩效指标时，才选择该根因；而并集操作则检测传播至至少一个目标关键绩效指标的所有可能根因。

5 实验

本节首先评估 BALANCE 在合成数据上的性能。随后，我们展示 BALANCE 在三个现实世界应用中的性能，包括异常 SQL定位、容器故障定位和 Exathon故障类型诊断。第一个应用主要关注目标关键绩效指标与候选根因之间的同质对，而后两个应用则考虑异质对。

实现细节： 我们采用 BALANCE 框架；对于前向模块，我们从估计准确率和运行时间两个方面，将 BMFS 与其他最先进方法进行比较，包括 Lasso、E-Net 和 ARD。后向模块保持不变。为进行公平比较，我们以原始马蹄先验替换相关马蹄先验，并将所得方法视为 ARD。因此，ARD 可被视为对 BMFS 的消融研究，即移除其中的 g-先验部分。需要注意的是，在原始 ARD 中， [36], 使用 Student's T-先验来鼓励稀疏性。根据我们的分析，所有比较方法的计算复杂度均为 O(p^2 max(n, p))。除新加入的 R 软件包"fsMTS"外，所有方法均使用 Python 3.9 实现。所有实验均在同一台配备 Intel Xeon E5-2682 v4 @ 2.50GHz 处理器和 16GB 内存的 Linux服务器上进行，以便直接比较所有方法的运行时间。

5.1 合成数据模拟

在此，我们按如下方式生成合成数据：

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{37}$$

其中， $\boldsymbol{X} \in \mathbb{R}^{n\times p}$、 $\boldsymbol{y} \in \mathbb{R}^n$、 $\boldsymbol{\beta} \in \mathbb{R}^p$ 和 $\boldsymbol{\epsilon} \sim$服从 N (0， α^{-1})。为了向 $\boldsymbol{X}$ 和 $\boldsymbol{\beta}$引入多重共线性，我们假设

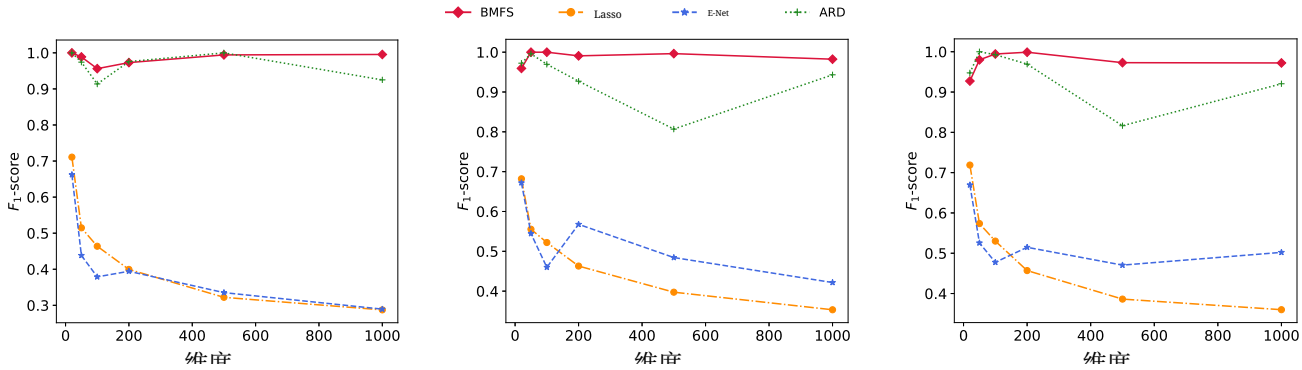
$$\boldsymbol{\beta} = \boldsymbol{Q}\boldsymbol{b}, \tag{38}$$

$$\boldsymbol{X} = \boldsymbol{Z}\boldsymbol{Q}^{-1}, \tag{39}$$

其中， $\boldsymbol{b} \in \mathbb{R}^{p_z}$、 $\boldsymbol{Z} \in \mathbb{R}^{n\times p_z}$ 和 $\boldsymbol{Q} \in \mathbb{R}^{p_z\times p_z}$表示某个矩阵 $\boldsymbol{W}$的 QR分解的正交部分。注意， $p_z \le p$。通过指定 $\boldsymbol{W}$为单位矩阵并令 $p_z = p$，所得特征 $\boldsymbol{X}$彼此独立（即不存在多重共线性）。另一方面，通过指定 $\boldsymbol{W}$ 为随机矩阵并令 $p_z < p$，我们向$\boldsymbol{X}$引入了一些多重共线性（即部分多重共线性）。最后，通过指定 $\boldsymbol{W}$ 为选择矩阵，其中每一行只有一个非零元素，该元素可以是 1 或 -1，并且 $p_z < p$， $\boldsymbol{X}$中任意两列之间的相关性可以为 1 或 -1。我们将这种情况称为"完全多重共线性"。

我们研究了 p 与 n之间的比率、多重共线性水平、噪声水平、稀疏性水平以及缺失数据比例对所提出的 BALANCE 方法性能的影响。如前所述，我们将 BMFS 与 Lasso、E-Net 和 ARD 进行了比较。具体而言，对于 Lasso，候选集

[2]a. 数据集不包含任何个人身份信息（PII）。b. 数据集经过脱敏和加密处理。c. 实验过程中采取了充分的数据保护措施，以防止数据复制泄露风险，并在实验结束后销毁数据集。d. 数据集仅用于学术研究，不代表任何真实业务场景。



(a) 缺失 (b) 部分 (c) 完美

图5：在不存在多重共线性（a）、存在部分多重共线性（b）和存在完全多重共线性（c）的情况下，所有基准测试方法得出的 F_1-得分随维度 p 的变化。

表 1：不同多重共线性水平（缺失、部分和完美）及不同维度的合成数据结果 p 平均值基于 100 次试验。此处，$n = 100$、$\alpha^{-1} = 0.01$以及非零系数的比例 $= 0.5\%$。

	p	Lasso			E-Net			ARD			BMFS		
		F_1-分数	MSE	时间（秒）	F_1-分数	MSE	时间（秒）	F_1-分数	MSE	时间（秒）	F_1-分数	MSE	时间（秒）
Absent	20	0.7109	6.90e-05	4.70e-01	0.6623	7.490e-05	4.65e-01	**0.9999**	**2.60e-05**	**2.34e-01**	**0.9999**	**2.60e-05**	2.35e-01
	50	0.5148	4.10e-05	5.06e-01	0.4383	6.60e-05	5.08e-01	0.9743	1.50e-05	3.10e-01	**0.9887**	**1.30e-05**	**2.72e-01**
	100	0.4638	3.00e-05	5.84e-01	0.3789	5.10e-05	5.88e-01	0.9137	1.70e-05	4.47e-01	**0.9562**	**1.20e-05**	**4.10e-01**
	200	0.3995	1.70e-05	8.59e-01	0.3943	3.60e-05	8.62e-01	**0.9758**	**4.00e-06**	**3.20e-01**	0.9730	**4.00e-06**	3.51e-01
	500	0.3221	9.00e-06	1.99e+00	0.3352	2.60e-05	2.00e+00	**0.9999**	**1.00e-06**	5.32e-01	0.9943	**1.00e-06**	**4.94e-01**
	1000	0.2873	5.74e-06	3.01e+00	0.2894	1.96e-05	3.25e+00	0.9249	1.87e-05	2.12e+00	**0.9955**	**9.35e-07**	**2.00e+00**
Partial	20	0.6821	4.16e-02	4.72e-01	0.6720	2.85e-02	4.77e-01	**0.9724**	**2.86e-02**	**2.54e-01**	0.9593	2.87e-02	4.15e-01
	50	0.5554	1.90e-02	5.50e-01	0.5445	1.32e-02	5.84e-01	0.9958	**1.30e-02**	**2.91e-01**	**0.9999**	**1.30e-02**	3.70e-01
	100	0.5221	9.36e-03	6.18e-01	0.4598	6.48e-03	6.76e-01	0.9696	6.32e-03	3.82e-01	**0.9999**	**6.32e-03**	**3.36e-01**
	200	0.4630	4.56e-03	8.52e-01	0.5678	3.25e-03	9.52e-01	0.9271	3.17e-03	1.06e+00	**0.9908**	**3.16e-03**	**4.59e-01**
	500	0.3974	1.93e-03	1.87e+00	0.4843	1.36e-03	2.24e+00	0.8069	1.32e-03	1.40e+00	**0.9963**	**1.31e-03**	**6.55e-01**
	1000	0.3532	9.19e-04	2.89e+00	0.4216	6.45e-04	3.83e+00	0.9431	6.22e-04	2.13e+00	**0.9824**	**6.18e-04**	**2.09e+00**
Perfect	20	0.7188	4.24e-02	4.74e-01	0.6696	**2.84e-02**	4.82e-01	**0.9474**	2.93e-02	**2.83e-01**	0.9273	3.02e-02	4.13e-01
	50	0.5736	1.92e-02	5.11e-01	0.5257	1.35e-02	5.42e-01	**0.9999**	**1.33e-02**	**2.67e-01**	0.9799	1.34e-02	3.61e-01
	100	0.5301	9.78e-03	5.83e-01	0.4775	6.82e-03	6.39e-01	0.9927	6.66e-03	**3.15e-01**	**0.9944**	**6.66e-03**	3.85e-01
	200	0.4570	4.36e-03	7.82e-01	0.5147	3.11e-03	9.04e-01	0.9693	3.03e-03	5.49e-01	**0.9988**	**3.03e-03**	**4.50e-01**
	500	0.3862	1.52e-03	1.81e+00	0.4705	1.07e-03	2.20e+00	0.8164	1.03e-03	1.53e+00	**0.9729**	**1.03e-03**	**8.11e-01**
	1000	0.3599	7.62e-04	2.87e+00	0.5022	5.23e-04	3.69e+00	0.9205	5.07e-04	**2.11e+00**	**0.9722**	**4.99e-04**	2.12e+00

表 2：在不同噪声标准差（噪声标准差）水平下的合成数据结果，取 100 次试验的平均值。此处，$n = 100$、$p = 1000$ 和非零系数的比例$= 0.5\%$。

噪声标准差	Lasso			E-Net			ARD			BMFS		
	F_1-得分	MSE	时间（秒）	F_1-分数	MSE	时间（秒）	F_1-分数	MSE	时间（秒）	F_1-分数	MSE	时间（秒）
0.01	0.8466	9.07e-04	2.91e+00	0.8820	6.150e-04	3.84e+00	0.9250	6.16e-04	2.14e+00	0.9789	**6.10e-04**	**2.05e+00**
0.1	0.3746	8.81e-04	2.88e+00	0.4562	6.06e-04	3.71e+00	0.9231	5.87e-04	**2.06e+00**	0.9625	**5.80e-04**	2.11e+00
1.0	0.3353	**1.07e-03**	**3.14e+00**	0.3294	1.05e-03	4.57e+00	0.3596	1.32e-03	3.19e+00	0.4054	1.39e-03	3.51e+00
3.0	0.1150	2.19e-03	3.90e+00	0.1337	**2.09e-03**	6.19e+00	0.0863	1.08e-02	3.38e+00	0.0881	1.16e-02	**3.37e+00**

表3：不同非零系数百分比（非零系数%）下合成数据的结果，取100次试验的平均值。其中，$n = 100$、$p = 1000$、以及 $\alpha^{-1} = 0.01$。

非零系数百分比	Lasso			E-Net			ARD			BMFS		
	F_1-分数	MSE	时间（秒）	F_1-分数	MSE	时间（秒）	F_1-分数	MSE	时间（秒）	F_1-分数	MSE	时间（秒）
0.2%	0.4482	2.16e-04	2.95e+00	0.4054	1.19e-04	3.53e+00	0.9766	1.08e-04	1.97e+00	**0.9999**	**1.07e-04**	**1.93e+00**
0.5%	0.3927	8.20e-04	2.97e+00	0.4838	5.87e-04	3.96e+00	0.9296	5.67e-04	2.30e+00	**0.9726**	**5.63e-04**	**1.94e+00**
1%	0.3810	2.19e-03	3.02e+00	0.5448	1.76e-03	4.58e+00	0.8913	1.75e-03	2.30e+00	**0.9654**	**1.71e-03**	**1.97e+00**
2%	0.3687	6.13e-03	2.95e+00	0.5673	5.47e-03	5.37e+00	0.7407	5.52e-03	2.51e+00	**0.9238**	**5.31e-03**	**1.99e+00**
5%	0.3694	1.61e-02	3.33e+00	0.4853	1.58e-02	6.71e+00	0.4409	1.57e-02	2.85e+00	**0.5483**	**1.56e-02**	**2.65e+00**

调节参数在该区间内均匀分布的 30 个取值 $[-2, 2]$ 在对数尺度下。对于 E-Net，常用的惩罚参数

表4： X**中不同缺失值比例下合成数据的结果，取100次试验的平均值。这里，** $n = 100$、 $p = 1000$、 $\alpha^{-1} = 0.01$**，以及非零系数的比例=** 0.5%。

缺失值	Lasso			E-Net			ARD			BMFS		
	F_1-得分	MSE	时间（秒）	F_1-得分	MSE	时间（秒）	F_1-得分	MSE	时间（秒）	F_1-得分	MSE	时间（秒）
10%	0.4306	**6.65e-04**	3.72e+00	0.3791	6.87e-04	3.98e+00	0.7461	7.92e-04	4.23e+00	**0.7494**	7.92e-04	**3.60e+00**
20%	0.4076	**7.46e-04**	**3.63e+00**	0.4226	7.84e-04	3.98e+00	0.7074	8.90e-04	5.47e+00	**0.7835**	8.04e-04	4.65e+00
30%	0.4083	6.97e-04	**3.45e+00**	0.4381	7.99e-04	3.93e+00	0.6721	7.90e-04	6.80e+00	**0.7714**	**6.78e-04**	5.94e+00
40%	0.4103	8.41e-04	**3.64e+00**	0.3983	7.69e-04	4.41e+00	0.6726	8.89e-04	9.81e+00	**0.7670**	**7.69e-04**	8.64e+00
50%	0.3474	9.73e-04	**4.01e+00**	0.3327	1.06e-03	4.97e+00	0.6170	9.17e-04	1.40e+01	**0.6727**	**8.68e-04**	1.30e+01

在 ℓ_1 和 ℓ_2 范数之前的参数分别从 $[-2, 2]$ 对数尺度下等间距选取的 10 个值中选择， ℓ_1 比率参数则从 $[0.1, 0.5, 0.9]$ 中选取。最优调优参数通过交叉验证选定。由于这两种方法无法显式处理缺失值，我们用相应候选项的均值替换缺失值。我们考虑三个指标，即关于估计 $\boldsymbol{\beta}$与真实 $\boldsymbol{\beta}$之间零模式的 F_1-分数、估计 $\boldsymbol{\beta}$ 与真实 $\boldsymbol{\beta}$之间的均方误差（MSE），以及运行时间。需要注意的是， F_1-分数是精确率和召回率的调和平均值，其中精确率定义为模型识别出的真实根因数量与模型给出的全部根因数量之比，召回率定义为模型识别出的真实根因数量与真实根因总数之比。结果汇总于表 1–4 中。为突出 BMFS 在零模式恢复方面的优势，我们进一步在图 5 中绘制了所有方法的 F_1-分数随 p 变化的曲线。

从表格中可以看出两个主要趋势。首先，就估计准确率和运行时间而言，BMFS 通常能够达到与最先进方法相当或更好的性能，尤其是在维度 p较高且多重共线性程度较高时。另一方面，BMFS 和 ARD 优于 E-Net 和 Lasso，这表明自适应地学习

通过变分推断从数据中估计调优参数，比通过穷举网格搜索进行估计更具优势。网格搜索会将调优参数限制在预先设定的候选项集合中，因此，设计不当的集合可能导致性能不佳。此外，可以观察到，当数据中存在多重共线性时，BMFS 和 E-Net 的性能分别优于 ARD 和 Lasso，这符合预期。事实上，采用 ℓ_1-范数惩罚的 Lasso 在理论上最多只能选出 n 个候选项（当 $p > n$时），即使所有候选项都具有相关性。更确切地说，如果存在两个或更多高度共线的候选项，Lasso 只会随机选择其中一个。这解释了与 BMFS 和 E-Net 相比，ARD 和 Lasso 的不足之处。

其次，BMFS 的性能对维度数量、多重共线性水平、噪声水平、稀疏度水平以及缺失值比例均具有稳健性。具体而言，我们可以看到，对于 $p \le n$ 和 $p > n$，贝叶斯方法始终优于频率学派方法。此外，贝叶斯方法通过从数据中推断缺失值的分布，提供了一种直接处理缺失值的方式，但随着缺失数据比例的增加，其耗时也会增加。相比之下，我们在应用频率学派方法之前采用均值插补法；当缺失值比例较大时，这会导致估计不准确。

Tab表 5：不同租户关键绩效指标及其对应的 SQL 指标 指标。

租户 KPI	SQL 指标
SQL_SELECT_RT	*CPU*_时间
逻辑读_	逻辑读*(*逻辑_读*)*
*SQL*_队列_时间	队列_时间
RPC 包输入/输出_ _	*rpc*_计数

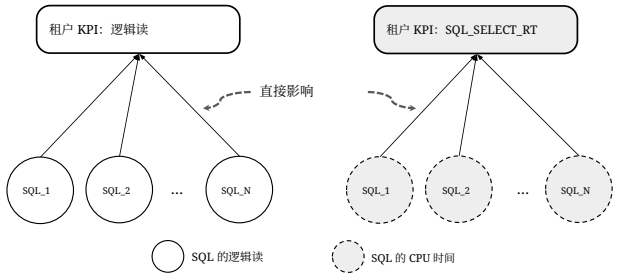


图 6：异常 SQL 定位的目标和候选项。

5.2 BadSQL 定位

数据库服务是企业日常业务至关重要的基础设施。因此，确保数据库服务的高可用性是重中之重。以 iSQUAD [19], 为代表的既有工作，专注于从专家给出的典型类型或历史数据中确定间歇性慢 SQL 的故障类型。与慢查询不同，慢查询会大量出现在慢查询日志中，而异常 SQL 定位是一个更为全面的问题，因为我们不仅考虑运行时间，还考虑逻辑读、RPC 次数等指标。异常 SQL 不一定总能在慢查询日志中找到，因为它们可能通过 CPU 或内存使用率而非运行时间，导致租户关键绩效指标出现异常。如引言中所述，我们旨在找出可疑且应对租户关键绩效指标中检测到的异常负责的“坏 SQL”（即候选根因 $\boldsymbol{X}$），这些异常对应目标关键绩效指标 $\boldsymbol{y}$。

如表 5 所示，租户关键绩效指标用于监控租户的性能，而 SQL 指标则反映每条 SQL 的性能。结合专家知识和离线数据分析，我们发现，几乎所有异常 SQL 问题都可以通过两类租户关键绩效指标反映出来，即 *SQL_SELECT_RT* 和 *LOGIC_READS*。因此，这两个关键绩效指标的告警就是我们根因分析模块的触发条件。此外，我们发现，租户关键绩效指标可以视为相关 SQL 指标的汇总。例如，租户关键绩效指标 *SQL_SELECT_RT*受所有 SQL 的 *cpu_time* 指标影响，而另一个关键绩效指标 *LOGIC_READS* 则与 *lr* 指标（即逻辑读）相关。因此，BALANCE 在此应用中考虑了同质 $\boldsymbol{X}$和$\boldsymbol{y}$。图 6 展示了 BALANCE 在异常 SQL 定位中需要分析的目标与候选对。

表6：异常 SQL 定位结果。

方法	#命中数	#未命中数	准确率	时间（秒）	#推荐数
ARD	68	24	0.7556	1.92e+00	2.7
Lasso	64	24	0.7111	3.15e+00	2.1
E-Net	64	26	0.7222	2.95e+00	2.3
fsMTS	49	41	0.5444	1.20e+01	5.0
BMFS	75	15	0.8333	1.78e+00	2.3

此外，在线监控平台上的租户关键绩效指标以固定间隔的时间序列形式保存，而 SQL 指标则通过批量采样收集：每个批次收集采集时间前 10 秒内所有 SQL 的指标，下一次采集在上一次采集完成存储过程后开始，因此每两次采集之间会存在由存储时间造成的细小且不均匀的时间间隔。为解决这一问题，我们将每分钟内的 SQL 指标进行聚合，并将其视为时间序列。因此，$\boldsymbol{X}$和 $\boldsymbol{y}$的时间戳可能无法对齐。因此，在与 $\boldsymbol{y}^t$ 对齐时，我们同时使用 $\boldsymbol{X}^t$和 $\boldsymbol{X}^{t-1}$；当 $\boldsymbol{x}_j^t$或 $\boldsymbol{x}_j^{t-1}$前的系数不为零时，将 SQL ^{x_j} 输出为可能的异常 SQL。

最后，虽然只有两个目标关键绩效指标，但相关 SQL p 的数量可能非常庞大（例如数千条甚至更多），并且因具体情况而异。SQL 往往彼此相关，因为在一条 SQL 语句中对 WHERE 或 LIMIT条件进行小幅修改，就会被定义为一条新的 SQL。缺失数据的比例也可能非常高。此外，还要求在 1 分钟内找到根因，并且结果应具有可解释性。总而言之，在异常 SQL 定位问题中，第 3 节所述的全部 6 项要求都应得到满足。为了合并两个关键绩效指标产生的根因，我们为每个关键绩效指标保留排名前 3 的根因（即，在合并模块 $\kappa = 3$中），并计算两组根因的并集，作为最终推荐结果。

为了评估所提方法的性能，我们从 Ocean Base 的 SQL诊断平台[41]收集了 90 个样本。这些样本涵盖 50 多个不同的租户，每个样本都将租户关键绩效指标和 SQL 指标保存为 61 分钟的时间序列（即 $n = 61$），其中包括 1 分钟的异常部分。请注意，p 的规模会扩大到数百甚至数千。因此，$p > n$ 在这种情况下。SQL 指标中通常存在缺失数据，缺失比例最高可达 50%。为满足实时根因分析任务的要求，我们选取了三项评估标准：准确率、推荐数和运行时间。对于每个样本，如果真实根因位于所提方法推荐的 SQL 集合中，则将该样本记为“命中”；否则记为“未命中”。准确率定义为命中数占样本总数的比例。同样，我们比较了不同的前向模型。由于 Lasso 和 E-Net 无法处理缺失数据，我们对租户关键绩效指标中的缺失值采用线性插值进行填补，并将 SQL 指标中的缺失值填为零。此外，我们还考虑了另一种基准方法 fsMTS（多变量时间序列特征选择）[23]。该方法考虑滞后的时间依赖性，并从 $\boldsymbol{X}_t, \cdots \boldsymbol{X}_{t-\tau}$ 中筛选特征，其中 τ表示时间滞后。随后，该方法采用集成方法，结合互相关、图形化Lasso、随机森林等方法的结果。由于 fsMTS 只能判断某个特征是否被选中，我们对每个关键绩效指标调整调节参数，使其仅选出 $\kappa = 3$ 个候选项。所有方法的结果见表6。

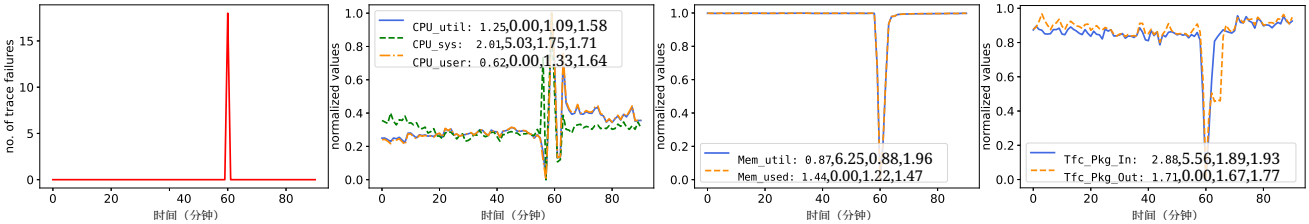
可以看出，所提方法的准确率最高，推荐数最少，计算耗时也最低。其准确率为 83.33%，至少比基线方法高 8%。推荐数仅为 2.3，平均运行时间仅为 1.78 秒。需要指出的是，在保证完整性的同时，推荐数应尽可能少，这有助于站点可靠性工程师将注意力集中在少数根因上，从而深入排查并更高效地解决问题。另一方面，从表6可以看出，BMFS 和 E-Net 的表现分别优于 ARD 和 Lasso，因为前两种方法能够处理数据中的多重共线性。我们还发现，贝叶斯方法（即 BMFS 和 ARD）的表现优于其频率学派对应方法（即 E-Net 和 Lasso），因为它们能够从数据中自适应地学习调节参数的分布以及缺失值。我们已将所提方法部署到生产环境，线上结果显示其准确率最高可达 95%，这进一步证明了 BALANCE 的优势。最后，fsMTS 的结果最差，可能是因为：1）对于按分钟分组的时间序列而言，时间依赖性并非主要关注点；2）集成中的基学习器（例如图形化Lasso）也需要仔细调整惩罚参数，但此处仅使用了默认值。

5.3 容器故障定位

本节将所提出的方法进一步应用于另一种实际场景，即容器故障定位。在该场景中，一旦与某个容器相关的跟踪失败数量出现异常，所提出的 BALANCE 方法就会找出能够最佳解释该异常的容器指标，而根因分析结果将进一步促成重启和流量限流等自愈操作。通过自动处理所有跟踪失败，可以保证云原生系统的高可用性。

在此案例中，目标关键绩效指标 $\boldsymbol{y}$ 是与某个容器相关的跟踪失败数量，候选根因 $\boldsymbol{X}$ 是该容器的 10 项指标，例如 CPU 使用率、内存使用率、入向和出向流量、TCP连接数等。请注意，在此设置下，$\boldsymbol{y}$和 $\boldsymbol{X}$具有异质性。此外，容器指标之间存在很强的相关性。例如，CPU 使用率这一指标是另外两项指标之和：CPU用户态使用率和 CPU 系统使用率；此外，以字节为单位的入向流量、以字节为单位的出向流量、以数据包为单位的入向流量以及以数据包为单位的出向流量彼此密切相关。缺失数据也时有出现，比例最高可达 10%。

为评估我们方法的质量，我们从蚂蚁集团的故障诊断平台共收集了100个样本。对于每个样本，我们可以获取告警时间，以及异常发生前和异常期间 $\boldsymbol{X}$和 $\boldsymbol{y}$的30分钟时间序列。在此情况下，$n = 30$、$p = 10$ $p < n$ 以及。我们将所提出的前向模型BMFS与Lasso、E-Net、ARD和fsMTS进行对比，并从精确率、召回率、F_1-分数和运行时间等方面评估所有模型。结果列于表7中。需要注意的是，将BALANCE部署到生产环境后也取得了相似的结果。在 F_1-分数和运行时间方面，BMFS均远胜于Lasso、E-Net、ARD和fsMTS。事实上，BMFS所得的 F_1 分数比排名第二的方法ARD高出20%以上。具体而言，BMFS和E-Net的精确率分别与ARD和Lasso相近，而前两种方法的召回率要高得多，这表明ARD和Lasso错误地遗漏了一些相关指标，也凸显了考虑候选根因之间多重共线性的优势。



(a) 目标关键绩效指标 y (b) CPU相关指标 (c) 内存相关指标 (d) 流量相关指标

图7：流量的可视化 故障数量 y 以及相关的容器指标：每个指标对应的四个值在图例中依次表示 ARD、Lasso、E-Net 和 BMFS 的得分。 **he**

表7：容器故障定位结果。

方法	召回率	精确率	F_1-分数	时间（秒）
ARD	0.4784	0.98804	0.6266	3.54e-1
Lasso	0.5083	0.5219	0.4712	4.96e-1
E-Net	0.7352	0.5520	0.5925	4.58e-1
fsMTS	0.4826	0.5244	0.4655	7.01e-1
BMFS	0.8180	0.9136	0.8569	1.52e-1

在图7中，我们以目标关键绩效指标和相关候选根因为例进行描绘，并依次展示图中每个候选根因对应的ARD、Lasso、E-Net和BMFS的归因分数。可以看出，对于高度相关的根因，BMFS会产生相近的归因分数（即§2.2中的连续性公理），而其他方法所得的归因分数并不总是遵循这一公理。此外，我们可以观察到，BMFS和ARD的 F_1-分数均高于E-Net和Lasso，这说明根据数据自适应地学习调优参数分布具有优势。最后，尽管在该场景中 $\boldsymbol{X}$ 和 $\boldsymbol{y}$ 之间存在异质性，所提出的BALANCE仍然运行良好，这可能得益于线性归因方法 [18]与模型无关的特性。另一方面，我们注意到，异常期间异质的 $\boldsymbol{x}_j$ 和 $\boldsymbol{y}$ 之间的关系可以用线性模型很好地近似，因为类似的现象（即波形异常）通常会同时发生在二者上。例如，$\boldsymbol{x}_j$中的异常尖峰通常会导致 $\boldsymbol{y}$出现尖峰，而$\boldsymbol{x}_j$的突增则会导致 $\boldsymbol{y}$增加或减少。

5.4 Exathlon数据集故障类型诊断

我们进一步研究了 BALANCE 在公开基准 Exathlon [12],上的性能。该基准包含在为期 2.5 个月的时间里，从一个由 4 个节点组成的集群中运行的 10 个 Spark 流处理应用收集的真实数据轨迹。执行过程中，轨迹中注入了 5 种类型的故障，包括突发输入（类型1）、突发输入直至崩溃（类型2）、输入停滞（类型3）、CPU争用（类型4）和进程故障（类型5）。每次注入故障的持续时间约为 15 至 30 分钟。对于每次故障注入，都会收集来自 Spark 的 2,283 项指标的时间序列。原始 Exathlon 会先对多变量时间序列进行异常检测，然后将整个多变量时间序列中的异常归因到各个单独的时间序列。本文基于 Exathlon 构建了一项新的故障类型诊断任务（命名为 Exathlon-ftd），用于评估我们的方法。Exathlon-ftd 按以下方式构建。

首先，我们为该数据集定义目标关键绩效指标 $\boldsymbol{y}$ 和候选指标 $\boldsymbol{X}$。目标关键绩效指标应受到各种故障类型的影响，并可用于触发根因分析流程。为此，我们在对每次注入的所有指标进行去趋势处理后，使用 3σ 规则执行异常检测，并找出所有情况下异常指标的交集。我们从交集中选择指标处理_延迟 作为目标关键绩效指标 $\boldsymbol{y}$，因为从理论和实证角度来看，处理时间都会受到上述所有故障类型的影响。随后，通过图1中的候选异常检测模块，将在 $\boldsymbol{y}$ 异常期间相同时间段内出现异常的指标选为候选指标 $\boldsymbol{X}$。

其次，一旦从 $\boldsymbol{X}$中选出根因指标，我们便建立规则来确定故障类型。为简单起见，我们仅关注前4种事件，并定义4条规则（R1-R4），根据估计的根因指标确定故障类型：

R1。如果有关流量的指标，例如最后_完成_批次_记录和已处理_记录，位于估计的根因集合中，那么问题出在输入（即流量）上。如果这些指标呈上升趋势，故障类型则为“突发输入”或“崩溃前突发输入”；如果呈下降趋势，则为“停滞输入”。

R2。如果不满足*R1*，且估计的根因集合包含集中在单个节点上的 CPU 指标，那么故障类型为“CPU争用”。其依据是，流量问题可能导致 CPU 指标出现异常，但 CPU争用问题反过来不会影响流量指标。

R3。如果上述两条规则均不满足，且估计的根因集合包含执行器_活动_任务指标，那么故障类型为“突发输入”或“崩溃前突发输入”。我们注意到，流量增加和 CPU争用问题都会导致执行器_活动_任务异常增加。然而，CPU争用问题通常源于单个节点上的另一个程序，该程序占用了该节点上所有可用的 CPU核心，因此，估计根因集合中的 CPU 根因指标通常集中在单个节点上（即*R2*）。由于不满足这一条件，我们可以得出结论：此处的故障类型为“突发输入”。

R4。如果上述三条规则均不满足，则故障类型为“未知”。

经过这两个步骤，共收集了 73 个根因分析任务案例。对于每个案例，我们分别按分钟聚合 $\boldsymbol{X}$ 和 $\boldsymbol{y}$，并分别保留前面 60 分钟和 5 分钟，

表8：Exathlon故障类型分析结果。

方法	#命中数	#未命中数	准确率	时间（秒）	#推荐数
ARD	54	19	0.7397	1.29e+00	3.4
Lasso	51	22	0.6986	1.58e+00	4.4
E-Net	49	24	0.6712	1.67e+00	4.7
fsMTS	36	37	0.4932	4.45e+01	4.9
BMFS	58	15	**0.7945**	1.60e+00	3.2

以及数据集中可用的 根_因_状态 时间。因此， $n = 66$。异常检测后的候选项数量 X 大约扩展至 $p = 150$。随后，我们在构建的 Exathlon-ftd 任务上比较所有方法。结果如表 8 所示。BMFS 再次优于所有对比方法，准确率最高，推荐的根因指标数量最少。BMFS 的运行时间位居第二，仅略长于 ARD。

6 结论与未来工作 K

本文的主要贡献在于从可解释人工智能的角度审视根因分析问题，为其提供了新的视角。具体而言，我们假设目标关键绩效指标的异常行为可以由相关候选根因解释，并提出了一种名为 BALANCE 的新型基于归因的根因分析方法。具体来说，我们首先学习一个贝叶斯稀疏多重共线性模型，根据候选根因预测目标关键绩效指标。随后，我们通过计算候选根因的归因分数，将目标关键绩效指标的异常行为归因于这些候选项。实证结果表明，所提方法在合成数据和三个真实世界问题上均取得了优异表现，这些问题包括异常 SQL 定位、容器故障定位和故障类型诊断。我们注意到，前一种情况下的目标关键绩效指标与候选根因具有同质性，而后两种情况下则具有异质性；其中，$p > n$ 用于第一个和第三个应用， $p \leq n$ 用于第二个应用。我们已将 BALANCE 部署到生产环境，为分布式数据系统中的异常 SQL 和容器问题提供实时根因诊断。尽管这些应用尚不全面，但它们表明 BALANCE 有望成为实际根因分析任务的通用工具。

未来的工作中，我们希望探索基于归因的根因分析与基于图的根因分析之间的联系。此外，我们希望将 BALANCE 中的线性模型推广到非线性前向模型，例如广义线性模型；这类模型具有更强的表达能力，更适合描述异构指标之间的关系。

参考文献

[1] Pooja Aggarwal、Ajay Gupta、Prateeti Mohapatra 等。2020年。通过使用黄金信号挖掘日志中的因果依赖关系，实现云应用中运行故障的定位。载于 面向服务计算国际会议。137–149。[2] Marco Ancona、Cengiz Oztireli 和 Markus Gross。2019年。使用用于夏普利值近似的多项式时间算法解释深度神经网络。载于 国际机器学习会议（*ICML*）。272–281。[3] Charles K Assaad、Emilie Devijver 和 Eric Gaussier。2022年。时间序列因果发现方法的综述与评估。人工智能研究期刊 73（2022年），767–819。[4] Sebastian Bach、Alexander Binder、Grégoire Montavon 等。2015年。通过逐层相关性传播解释非线性分类器决策的像素级解释。*PloS One* 10，7（2015年）。[5] Ranjita Bhagwan、Rahul Kumar、Ramachandran Ramjee 等。2014年。Adtributor：广告系统中的收入调试。载于 第 11 届 USENIX 研讨会网络化系 y统设计 g与实 p现 *(NSDI)* 第 43–55 页。

[6] Christopher M Bishop 和 Nasser M Nasrabadi。2006年。模式识别与机器学习。第 4 卷。[7] Jonathan Boss、Jyotishka Datta、Xin Wang 等。2021年。面向块相关预测变量稀疏回归的组逆伽马伽马收缩。arXiv 预印本 arXiv:2102.10670（2021年）。[8] Kirill Bykov、Marina M-C Höhne、Adelaida Creosteanu 等。2021年。解释贝叶斯神经网络。arXiv 预印本 arXiv:2108.10346（2021年）。[9] Carlos M Carvalho、Nicholas G Polson 和 James G Scott。2009年。通过马蹄铁先验处理稀疏性。载于人工智能与统计（AIStat）。73–80。[10] Pengfei Chen、Yong Qi、Pengfei Zheng 等。2014年。Causeinfer：在大规模分布式系统中基于层次因果图的自动化分布式性能诊断。载于 IEEE计算机通信会议（INFOCOM）。1887–1895。[11] Yingnong Dang、Qingwei Lin 和 Peng Huang。2019年。AIOps：现实世界的挑战与研究创新。载于 IEEE/ACM 第 41 届软件工程国际会议：配套论文集（ICSE-Companion）。4–5。[12] Vincent Jacob、Fei Song、Arnaud Stiegler 等。2021年。Exathlon：面向时间序列的可解释异常检测基准。VLDB基金会会议录（PVLDB）（2021年）。[13] Vimalkumar Jeyakumar、Omid Madani、Ali Parandeh 等。2019年。ExplainIt!——面向时间序列数据的声明式根因分析引擎。载于 ACM数据管理国际会议（SIGMOD）会议录。333–348。[14] Myunghwan Kim、Roshan Sumbaly 和 Sam Shah。2013年。面向服务的架构中的根因检测。ACM SIGMETRICS性能评估评论 41，1（2013年），93–104。[15] Wu Lin、Mohammad Emtiyaz Khan 和 Mark Schmidt。2019年。采用指数族近似混合的快速简洁自然梯度变分推断。载于 国际机器学习会议（ICML）。3992–4002。[16] Pantelis Linardatos、Vasilis Papastefanopoulos 和 Sotiris Kotsiantis。2020年。可解释人工智能：机器学习可解释性方法综述。Entropy 23，1，2020，18。[17] Dewei Liu、Chuan He、Xin Peng 等。2021年。MicroHECL：大规模微服务系统中的高效根因定位。载于 IEEE/ACM 第 43 届软件工程国际会议：软件工程实践（ICSE-SEIP）。338–347。[18] Scott M Lundberg 和 Su-In Lee。2017年。一种解释模型预测的统一方法。神经信息处理系统进展（NIPS）30（2017年）。[19] Minghua Ma、Zheng Yin、Shenglin Zhang 等。2020年。云数据库中间歇性慢查询的根因诊断。VLDB基金会会议录 13，8（2020年），1176–1189。[20] Raha Moraffah、Paras Sheth、Mansooreh Karami 等。2021年。时间序列分析的因果推断：问题、方法与评估。Knowledge and Information Systems（2021年），1–45。[21] Vinod Nair、Ameya Raul、Shwetabh Khanduja 等。2015年。学习层次化监控系统以检测和诊断服务问题。载于第 21 届 ACM SIGKDD知识发现与数据挖掘国际会议会议录。2029–2038。[22] Sarah E Neville、John T Ormerod 和 MP Wand。2014年。连续稀疏信号收缩的平均场变分贝叶斯：陷阱与补救措施。Electronic Journal of Statistics 8，1（2014年），1113–1151。[23] Dmitry Pavlyuk。2022年。fsMTS：多变量时间序列特征选择。[24] Juan Qiu、Qingfeng Du、Kanglin Yin 等。2020年。一种基于因果关系挖掘和知识图谱的云应用性能异常根因诊断方法。Applied Sciences 10，6（2020年），2166。[25] Marco Tulio Ribeiro、Sameer Singh 和 Carlos Guestrin。2016年。“我为什么要相信你？”解释任意分类器的预测。载于第 22 届 ACM SIGKDD知识发现与数据挖掘国际会议会议录。1135–1144。[26] Veronika Ročková 和 Edward I George。2014年。利用尖峰与 slab 先验处理多重共线性。Metron 72，2（2014年），217–229。[27] Avanti Shrikumar、Peyton Greenside 和 Anshul Kundaje。2017年。通过传播激活差异学习重要特征。载于国际机器学习会议（ICML）。3145–3153。[28] Karen Simonyan、Andrea Vedaldi 和 Andrew Zisserman。2014年。卷积网络深处：图像分类模型与显著性图的可视化。载于 ICLR 研讨会。[29] Jacopo Soldani 和 Antonio Brogi。2022年。（微）基于服务的云应用中的异常检测与故障根因分析：综述。ACM计算综述（CSUR）55，3（2022年），1–39。[30] Peter Spirtes 和 Clark Glymour。1991年。一种快速恢复稀疏因果图的算法。Social Science Computer Review 9，1（1991年），62–72。[31] Yi Sun 和 Mukund Sundararajan。2011年。多线性函数的公理化归因。载于 第 12 届 ACM电子商务会议会议录。177–178。[32] Yongqian Sun、Youjian Zhao、Ya Su 等。2018年。Hotspot：具有多维属性的可加性KPI异常定位。IEEE Access 6（2018年），10909–10923。

[33] Mukund Sundararajan、Ankur Taly 和 Qiqi Yan。2017年。深度网络的公理化归因。载于 国际机器学习会议（*ICML*）。3319–3328。[34] Jörg Thalheim、Antonio Rodrigues、Istemi Ekin Akkus 等。2017年。Sieve：从微服务受监控指标中提取可执行洞见。*arXiv* 预印本 *arXiv:1709.06686*（2017年）。[35] Robert Tibshirani。1996年。通过 Lasso 进行回归收缩与筛选。英国皇家统计学会杂志：*B*辑（方法论）58，1（1996年），267–288。[36] Michael E Tipping。2001年。稀疏贝叶斯学习与相关向量机。机器学习研究杂志（*JMLR*）1，6月（2001年），211–244。[37] Guanchu Wang、Yu-Neng Chuang、Mengnan Du 等。2022年。通过贡献者选择加速沙普利解释。载于 国际机器学习会议（*ICML*）。[38] Hanzhang Wang、Phuong Nguyen、Jun Li 等。2019年。Grano：面向云原生分布式数据平台的交互式图根因分析。*VLDB*基金会会议录12，12（2019年），1942–1945。[39] Ping Wang、Jingmin Xu、Meng Ma 等。2018年。Cloudranger：云原生系统的根因识别。载于 第 *18* 届 *IEEE/ACM* 集群、云与网格计算国际研讨会（*CCGRID*）。492–502。[40] Jianping Weng、Jessie Hui Wang、Jiahai Yang 等。2018年。公有云多层服务异常的根因分析。*IEEE/ACM* 网络学报26，4（2018年），1646–1659。[41] Zhenkun Yang、Chuanhui Yang、Fusheng Han、Mingqiang Zhuang、Bing Yang、Zhifeng Yang、Xiaojun Cheng、Yuzhong Zhao、Wenhui Shi、Huafeng Xi、Huang Yu、Bin Liu、Yi Pan、Boxue Yin、Junquan Chen 和 Quanqing Xu。2022年。OceanBase：每分钟 7.07 亿 tpmC 的分布式关系数据库系统。*VLDB*基金会会议录15，12（2022年），3385–3397。

[42] Hang Yu、Songwei Wu、Luyin Xin 等。2020年。利用自动稀疏性确定进行稀疏网络的快速贝叶斯推断。机器学习研究杂志 21（2020年），124–1。[43] Hang Yu、Luyin Xin 和 Justin Dauwels。2019年。用于图模型选择的变分 Wishart 近似：单尺度与多尺度模型。*IEEE* 信号处理汇刊 67，24（2019年），6468–6482。[44] Matthew D Zeiler 和 Rob Fergus。2014年。卷积网络的可视化与理解。载于欧洲计算机视觉会议（*ECCV*）。818–833。[45] Arnold Zellner。1986年。关于使用 g先验分布评估先验分布和贝叶斯回归分析。贝叶斯推断与决策技术 （1986年）。[46] Kai Zhang、Chao Tian、Kun Zhang 等。2021年。采用反向顺序剪枝和并行化策略的快速 PC算法。*arXiv* 预印本 *arXiv:2109.04626*（2021年）。[47] Xu Zhang、Chao Du、Yifan Li 等。2021年。HALO：面向层次结构的云系统故障定位。载于第*27*届 *ACM* 知识发现与数据挖掘会议论文集。3948–3958。[48] Yingying Zhang、Zhengxiong Guan、Huajie Qian 等。2021年。CloudRCA：云计算平台的根因分析框架。载于第*30*届 *ACM* 国际信息与知识管理会议（*CIKM*）论文集。4373–4382。[49] Xun Zheng、Bryon Aragam、Pradeep K Ravikumar 等。2018年。无撕裂的有向无环图：用于结构学习的连续优化。神经信息处理系统进展（*NIPS*）31（2018年）。[50] Hui Zou 和 Trevor Hastie。2005年。通过弹性网络进行正则化与变量选择。英国皇家统计学会杂志：*B*辑（统计方法）67，2（2005年），301–320。