软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software [doi: 10.13328/j.cnki.jos.006977]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向数据库配置优化的反事实解释方法*

朱 霄, 邵心玥, 张 岩, 王宏志

(哈尔滨工业大学 计算学部, 黑龙江 哈尔滨 150001)

通信作者: 王宏志, E-mail: wangzh@hit.edu.cn

摘 要: 数据库性能受数据库配置参数的影响, 参数设置的好坏会直接反映到数据库性能表现上, 因此, 数据库调参方法的优劣至关重要. 然而, 传统的数据库调参方法存在诸多局限性, 例如无法充分利用历史调参数据、浪费时间人力资源等. 而反事实解释方法是一种对原数据进行少量修改, 从而将原预测改变为期望预测的方法, 其起到的是建议的作用. 这种作用可以用于数据库配置优化, 即对数据库配置进行少量修改, 从而使得数据库的性能表现得到优化. 因此, 提出面向数据库配置优化的反事实解释方法, 对于在特定负载条件下性能表现不佳的数据库, 所提方法可以对数据库配置进行修改, 生成相应的数据库配置反事实, 从而优化数据库性能. 进行两种实验, 分别用于评估反事实解释方法的优劣以及验证其优化数据库的效果, 实验结果表明: 综合各个评估指标, 提出的反事实解释方法要优于其他的经典反事实解释方法, 并且生成的反事实能够确实有效地提高数据库性能.

关键词: 反事实解释; 数据库配置优化; 数据库智能化; 神经网络

中图法分类号: TP311

中文引用格式: 朱霄, 邵心玥, 张岩, 王宏志. 面向数据库配置优化的反事实解释方法. 软件学报. http://www.jos.org.cn/1000-9825/6977.htm

英文引用格式: Zhu X, Shao XY, Zhang Y, Wang HZ. Counterfactual Interpretation Method for Database Configuration Optimization. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/6977.htm

Counterfactual Interpretation Method for Database Configuration Optimization

ZHU Xiao, SHAO Xin-Yue, ZHANG Yan, WANG Hong-Zhi

(Faculty of Computing, Harbin Institute of Technology, Harbin 150001, China)

Abstract: The database performance is affected by the database configuration parameters. The quality of parameter settings will directly affect the performance of the database. Therefore, the quality of the database parameter tuning method is important. However, traditional database parameter tuning methods have many limitations, such as the inability to make full use of historical parameter tuning data, wasting time and human resources, and so on. The counterfactual interpretation methods aim to change the original prediction to the expected prediction by making small modifications to the original data. The method plays a role of suggestion, and this can be used for database configuration optimization, namely, making small modifications to the database configuration to optimize the performance of the database. Therefore, this study proposes a counterfactual interpretation method for database configuration optimization. For databases with poor performance under specific load conditions, this method can modify the database configuration and generate corresponding database configuration counterfactuals to optimize database performance. This study conducts two kinds of experiments to evaluate the counterfactual interpretation method and verify the effect of optimizing the database. The experimental results show that the counterfactual interpretation methods proposed in this study are better than other typical counterfactual interpretation methods in terms of various evaluation indicators, and the generated counterfactuals can effectively improve database performance.

Key words: counterfactual interpretation; database configuration optimization; AI for DB; neural network

* 基金项目: 国家自然科学基金 (62232005); 四川省科技计划 (2020YFSY0069)
收稿时间: 2022-11-14; 修改时间: 2023-02-15; 采用时间: 2023-06-05; jos 在线出版时间: 2023-10-18

数据库管理系统 (database management system, DBMS) 用于建立、使用以及维护数据库, 用户和数据库管理员都通过其对数据库进行操作. 而 DBMS 在工作过程中, 自身也有很多参数需要进行配置 (MySQL 有上百个配置参数; 而 Oracle 则有上千个配置参数), 如最大连接数、各种缓冲区内存大小等. 各个参数的设置对于 DBMS 性能的影响很大, 而这些参数的设置和组合不能简单地进行复用, 因为最佳配置取决于诸多要素, 针对当前环境找到数据库的最佳配置是一个 NP 困难问题[1].

关于深度学习, 近几年来, 在深度神经网络 (deep neural networks, DNN) 的推动下, 深度学习在图像处理、目标识别、机器翻译等相关领域的研究中取得了重要进展. 深度神经网络成功的一个关键因素是它的网络足够深, 相比于传统的浅层机器学习模型, 它可以通过大量非线性网络层的复杂组合对原始数据在各种抽象层面上进行特征提取, 从而挖掘出更多隐含的特征. 然而, 由于大多数深度学习模型有着高复杂性、低透明度的特点, 如同黑盒一般, 所以人们很难理解其中的原理, 也无法判断模型是否可靠[2]. 同时, 深度学习模型的这种不可解释性在很多领域都有潜在风险, 尤其是在国防安全领域[3]. 因此, 现在有越来越多的科研人员开始致力于深度学习模型可解释性的研究, 反事实解释方法就是其研究成果之一.

反事实解释方法生成的反事实有着类似于建议的作用, 而这种作用同样也可以应用在数据库的性能优化上, 本文旨在探究出一种面向数据库配置优化的反事实解释方法, 让数据库也能够受益于深度学习强大的学习力.

本文理论部分, 从特征梯度和特征重要性的角度出发, 提出了两种反事实解释方法, 分别是基于梯度排序的反事实解释方法以及基于特征重要性排序的反事实解释方法. 本文实验部分, 借助 MySQL 数据库以及 YCSB 性能测试工具, 采集并生成了数据库配置参数数据集; 然后针对该数据集, 搭建、训练和测试了 MLP 和 CNN 两类模型, 并对两者进行了分析和比较; 最后进行了反事实评估实验以及测试反事实优化数据库效果实验, 实验结果表明: 综合各个评估指标, 本文提出的反事实解释方法要优于其他的经典反事实解释方法, 并且生成的反事实能够确实有效地提高数据库性能.

本文主要贡献如下.

(1) 本文收集各类数据库配置优化相关的数据并整合生成 databaseConfiguration 数据集, 解决目前该研究方向缺乏开源数据的问题, 后续会对该数据集进行进一步完善, 并考虑将其开源以供其他学者研究使用.

(2) 本文在实验过程中搭建、训练和测试两种神经网络模型, 并对二者从多个方面进行对比分析, 然后选择更优者作为后续使用的预测模型, 降低模型不优带来的影响, 同时提高了实验的准确度.

(3) 本文首次将反事实解释方法引入到数据库配置优化领域, 并且经过反事实评估实验验证, 文中所提方法生成的反事实相比于其他对比方法更为优质, 这说明本文方法的优异性.

(4) 本文进行数据库性能测试实验以验证反事实优化数据库的效果, 实验结果显示, 文中所提方法生成的反事实对数据库的各个性能指标的优化效果显著, 这说明本文方法在数据库配置优化领域的可行性.

本文第 1 节介绍相关工作, 包括数据库配置优化的相关方法以及反事实解释方法的相关研究. 第 2 节介绍论文预备知识, 包括神经网络模型以及反事实解释方法. 第 3 节提出和研究两种反事实解释方法, 分别是基于梯度排序的反事实解释方法以及基于特征重要性排序的反事实解释方法, 并且从目标函数、算法流程以及复杂度分析 3 个方面对这两种方法进行了介绍. 第 4 节介绍论文实验, 包括实验数据集的采集, 实验模型的搭建、训练和测试, 反事实评估实验以及测试反事实优化效果实验. 第 5 节总结全文并展望未来工作.

1 相关工作

1.1 数据库配置优化的相关方法

以往的数据库调参方法可以分为 4 类, 分别是专家手工进行调整的调参方法、借助数据库调参辅助工具进行调整的调参方法、基于启发式算法的调参方法以及基于人工智能的调参方法[4].

(1) 专家手工调整的方法: 该方法是指, 数据库领域的专家根据自己的专业知识和调参经验决定最终的调整方案[4]. 该方法存在两个问题: 其一, 数据库参数繁多, 仅通过人工手段来决定最终的参数配置方案, 很难得到最佳配

置. 其二, 由于为各个参数样本进行负载实验耗费的时间较长, 所以数据库专家得到最终参数配置方案的时间也往往较长, 这在现实应用场景中, 带来的实际效益并不大.

(2) 借助调参辅助工具的方法: 数据库调参辅助工具分为基于规则的调参工具或者基于经验模型的调参工具.

基于规则的调参工具是由生产厂商根据数据库本身的特点而设计的, 这类工具可以收集当前数据库的信息并反馈给用户, 同时给出参数调整建议. MySQLTuner 就是该类工具的代表, 它能够收集数据库的状态信息, 根据这些信息, 使用固定的规则提出参数调整建议[4].

基于经验模型的调参工具从大量的测试中总结“模糊规则”, 探究数据库性能与其配置参数之间的联系, 从而帮助使用者进行参数的调整. Wei 等人提出的基于模糊规则的调优工具[5]就是该类工具的代表, 它首先使用自动负载仓库 (automatic workload repository, AWR) 提取数据信息; 然后从不同配置不同负载下, 数据库的表现中总结模糊规则; 最后基于经验模型来进行配置参数的推荐.

借助调参辅助工具的方法相比于专家手工调整的方法, 得到的参数配置方案更优, 耗时也更短. 但是该方法也存在以下问题: 前一类工具使用的规则固定, 应用场景很有限; 后一类工具总结出的模糊规则不具有复用性, 一旦进行场景的切换就需要重新进行统计.

(3) 基于启发式算法的方法: 该类方法利用启发式的算法自动化整个调参过程. Zhu 等人[6]提出了一个基于搜索的调参系统, 其核心算法主要由两部分组成: 首先, 通过划分与分治取样, 把整个 N 维参数空间离散化, 每个参数的取值范围划分成 k 段, 每一段随机取一个值, 这样整个参数空间就转化成 k^N 个点组成的样本空间. 然后, 利用有界递归搜索算法, 每次在抽样空间中随机选取 k 个样本测试, 得到表现最好的点 C_0. 再以 C_0 为中心划分出下一次的抽样区域, 这样迭代, 直到不能得到表现更好的点或者达到资源限制.

基于启发式算法的方法相比于借助调参辅助工具的方法, 应用场景更加广泛. 但是其也存在类似的问题, 例如不具备复用性, 不能充分利用历史数据.

(4) 基于人工智能的方法: 该类方法利用人工智能方法, 例如深度强化学习, 来优化调参过程. Zhang 等人提出了一种基于深度确定性策略梯度算法 (deep deterministic policy gradient, DDPG) 的调参系统 CDBTune[7], 它将工作负载、配置参数、性能指标等数据向量化成神经网络可以识别的张量, 然后, 在不同工作负载下, Actor 可以根据当前环境的状态给出合适的参数值, Critic 对 Actor 的行为进行评估, Actor 基于 Critic 的评估进行更新, 最后, Critic 基于每次得到的反馈值进行更新. Li 等人提出的 QTune[8]也是基于深度强化学习的调参系统, 它相比于 CDBTune, 有如下优势: 1) QTune 在查询计划级别对负载特征进行编码, 对执行开销做预估计, 从而提高了调参对不同负载的适应能力. 2) QTune 采用了一种面向调参问题的深度强化学习算法 (double-state deep deterministic policy gradient, DS-DDPG), 进一步提高了调参表现. 3) QTune 增加 Query2Cluster 模块, 更好地满足了不同用户对数据库性能的需求.

基于人工智能的两种方法, 即 CDBTune 与 QTune, 相比于前 3 类方法能得出更好的解, 而且也能更好地利用历史调参数据. 但是其在参数调优的过程中, 主要考虑调参方案对数据库性能的影响, 而忽略了调整后的参数可能会对整个操作系统所带来的影响, 例如系统资源浪费、系统性能降低等, 该问题也存在于前 3 类方法中, 而本文所研究的反事实解释方法, 则对此进行了分析与改进, 具体内容将会在第 2.2 节中进行陈述.

1.2 反事实解释方法的相关研究

关于反事实解释方法的研究, 近几年来, 深度学习可解释性领域的研究者们提出了很多的反事实解释方法.

一部分方法基于算法来求反事实, 例如 Le 等人提出的 GRACE 方法[9], 该方法首先对特征进行排序, 然后对前端的特征不断施加扰动来达到改变预测结果的目的; White 等人提出的 CLEAR 方法[10], 该方法针对黑盒模型, 对于数据求单条反事实; Sharma 等人提出的 CERTIFAI 方法[11], 该方法使用自定义遗传算法, 针对黑盒模型, 对于数据求多条反事实; Guidotti 等人提出的 LORE 方法[12], 该方法也是针对黑盒模型, 对于数据求多条反事实; 除了上述方法, Mothilal 等人提出的 DiCE 方法[13]同样也是基于算法来求解反事实. 这类方法通常求得反事实的速度较快, 而且求得的反事实与原数据距离比较近, 但是生成的反事实可能会不够真实.

还有一部分方法采用基于模型的方式来求反事实, 例如 Yang 等人提出的 MCS 方法[14], 它使用生成式对抗网络 (GAN) 来对数据空间进行探究, 通过生成模型和判别模型的对抗, 来使得生成的反事实更加贴近于原数据空间, 这类方法求得的反事实会更加贴近于现实, 但是和基于算法的方法相比, 这种方法生成反事实的速度会慢很多, 而且生成的反事实也会离原数据比较远.

2 预备知识

本文提出并研究了面向数据库配置优化的反事实解释方法, 下面将介绍其中涉及的相关知识和基本概念.

2.1 神经网络模型

在深度学习任务中, 3 类最常用的神经网络模型分别是多层感知机、卷积神经网络以及循环神经网络.

(1) 多层感知机: 多层感知机 (multi-layer perception, MLP) 在单层神经网络的基础上, 在输入层和输出层之间引入了一个或多个隐藏层 (hidden layer)[15]. 图 1 是一个多层感知机实例.

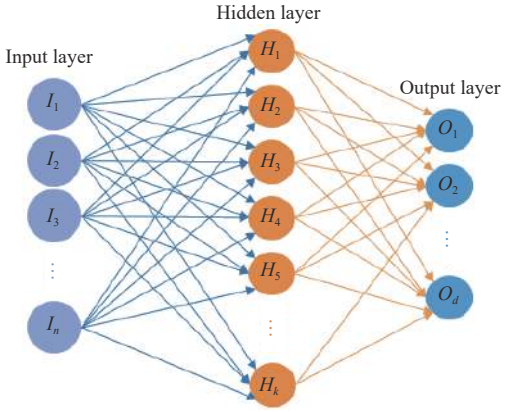

图 1 MLP 示意图

多层感知机的隐藏层和输出层都是全连接层, 它是一个全连接神经网络, 它的提出是为了解决单层神经网络无法拟合非线性问题的弊端. 多层感知机引入了激活函数, 将隐藏层的输出通过激活函数进行非线性变换, 从而达到能够拟合非线性问题的目的, 常见的激活函数有 ReLU 函数、Sigmoid 函数、tanh 函数等.

多层感知机最常应用的数据集就是表格数据集, 例如 Adult 数据集、German Credit 数据集等.

(2) 卷积神经网络: 卷积神经网络 (convolutional neural network, CNN) 在多层感知机的基础之上, 添加了卷积层和池化层等[16]. 图 2 是一个卷积神经网络实例.

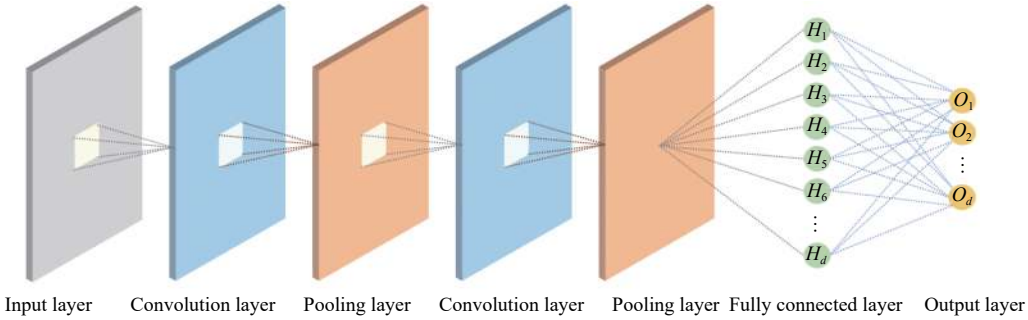

图 2 CNN 示意图

卷积神经网络的提出解决了传统神经网络参数多、占用内存大以及训练难度高的问题. 因为多层感知机采用的是全连接的方式, 所以当前一层的神经元数为 a , 后一层的神经元数是 b 时, 将两层进行连接的话, 将会产生

$a\times b+b$ 个参数. 卷积神经网络中的卷积层不再采用全连接的方式, 而是将后一层的神经单元与前一层的一小块区域通过卷积核进行连接, 这一小块区域的大小与卷积核的大小一致, 同时还提出了权重共享的概念, 共享同一层的卷积核, 这使得卷积神经网络的参数大大减少.

卷积神经网络由卷积层、池化层以及全连接层 3 部分组成. 卷积层的作用主要是特征提取, 池化层的作用主要是降低维度、压缩数据, 而全连接层的作用主要是预测分类.

卷积神经网络最常用于图像数据集, 例如 MNIST 数据集、Fashion MNIST 数据集等. 近几年来, 已经有研究者成功将卷积神经网络应用在了表格数据集和文本数据集, 而且效果不错, 因此对于这两类数据集, CNN 也是一个不错的选择.

(3) 循环神经网络: 循环神经网络 (recurrent neural network, RNN) 主要在多层感知机的基础之上, 对隐藏层进行了一定的修改[17]. 图 3 是一个循环神经网络实例.

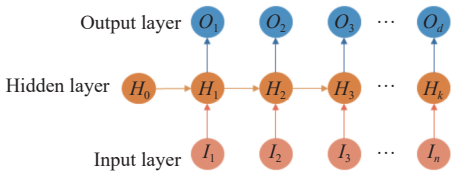

图 3 RNN 示意图

循环神经网络主要用于处理含有时序信息的数据, 这类数据是多层感知机和卷积神经网络无法处理的, 因为网络各个输入之间独立, 无法保存时序信息, 而循环神经网络通过引入循环层, 解决了这一问题.

循环神经网络中的循环层中每一个神经元, 都保存了之前输入的信息. 例如图 3 中的 H_2 保存了 I_1 的信息、H_3 保存了 I_1 和 I_2 的信息等; 也就是在处理输入 I_i 时, 可以利用 I_{i-1} 及之前所有输入数据的信息, 循环神经网络正是通过这样一种方式, 实现了对于含有时序信息的数据的处理.

循环神经网络最常用于含有时序信息的文本数据集.

(4) 残差网络: 残差网络 (residual network, ResNet)[18]是一个经典的 CNN 模型, 它的提出主要用于解决网络退化的问题. 网络退化是指: 随着网络深度的加深, 模型的训练效果不但没有变好, 反而变得越来越差. 按照常理来说, 如果浅层模型的训练效果不好的话, 可以通过适当加深网络来使得模型能获取的信息更多从而优化模型训练效果. 但是网络退化的现象很明显不符合常理, 其原因是, 如果在浅层模型已经能够很好地拟合现有问题的情况下, 还去一味地加深网络, 虽然理论上来说, 只要后续加深的层都是恒等变化, 那么加深之后的模型效果就会等同于原有浅模型的模型效果, 但是这只是理论上的结果, 要想将后续加深的层训练到都变为恒等变化, 其难度非常大, 甚至基本做不到, 所以才会导致网络退化的现象.

而 ResNet 通过引入残差块 (residual) 解决了网络退化的问题, 残差块示意图如图 4 所示, 残差块在一个或多个卷积层之间加上了一条“短接线 (shortcut connection)”, 在添加了短接线之后, 模型就产生了很大的变化: 原本在没有短接线的情况下, 由输入 x, 网络需要直接训练拟合预期输出 $H(x)$; 而在有了短接线之后, 网络只需要训练拟合残差函数 $F(x)=H(x)-x$, 然后再间接得到预期输出 $H(x)=F(x)+x$. 假设上一层的输出也就是这一层的输入 x 已经达到最优状态, 那么在没有短接线的情况下, 需要训练拟合的是 $H(x)=x$, 这个的训练难度会很大, 而添加了短接线之后, 需要训练拟合的就变为了 $F(x)=H(x)-x=0$, 这个的训练难度就降低了很多, 因为相比于训练成恒等变化, 训练成 0 的难度要简单很多.

残差块除了可以解决网络退化的问题, 还可以有效缓解梯度消失和梯度爆炸的问题, 因为它将层与层之间梯度的传递方式由累积变为了累加. 例如, 原本的层与层之间的连接方式可以描述为 $H(x)=G(F(x))$, 现在的层与层之间的连接方式可以描述为 $H(x)=G(x)+F(x)$, 在梯度传播的过程中, 减少梯度之间的乘积可以有效缓解梯度消失和梯度爆炸的问题.

ResNet 根据其层数 (卷积层和全连接层的层数之和) 的不同, 又可以分为 ResNet18[19]、ResNet34、ResNet50、ResNet101 以及 ResNet152 这 5 种常用模型, 这些模型名字中包含的数字代表它们的层数.

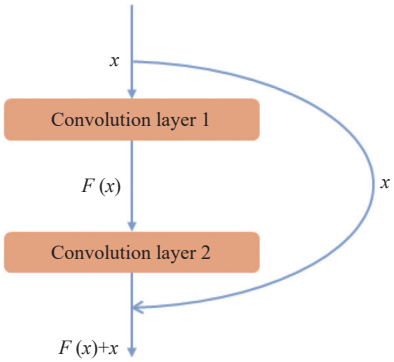

图 4 残差块示意图

2.2 反事实解释方法

(1) 公式化表达: 假设模型预测函数为 f, 其输出为 n 维向量, 每一维代表对应分类的预测概率, 原数据为 x, 在没有其他约束的条件下, 反事实 x' 的定义为:

$$\min_{x'} dist(x,x') \quad \text{s.t.} \quad \arg\max(f(x)) \neq \arg\max(f(x')) \tag{1}$$

其中, $dist$ 表示距离函数; $\arg\max$ 表示最大值对应的下标.

(2) 方法描述: 反事实解释方法旨在生成一条数据, 其尽量接近于原数据而预测结果又不同于原数据, 换句话说, 就是对原数据进行尽量少的修改, 使得其预测结果发生改变; 尽量接近于原数据体现在公式 (1) 中的 $dist$ 函数取 $\min$, 预测结果不同体现在公式 (1) 中的两个预测取 $\arg\max$ 不等. 与其他解释方法不同, 反事实解释并不能明确回答决策的“为什么”部分, 它是通过提供建议, 来实现解释的目的. 反事实解释也适用于黑盒模型 (因为在反事实解释方法中, 不需要知道模型的内部结构和原理, 只需要模型具有预测的能力即可), 因此不限制模型的复杂性, 也不要求披露模型.

这里用一个简单的例子来对反事实解释方法进行进一步说明: 假设 Alice 走进一家银行, 寻求住房抵押贷款, 而决定是否同意 Alice 的请求在很大程度上取决于深度学习分类器, 该分类器考量了 Alice 的特征向量, 即{收入, 信用值, 学历, 年龄}, 选择了拒绝 Alice 申请的贷款, 但是 Alice 对此感到不解, 她不禁想知道: ① 为什么贷款被拒绝了? ② 她可以采取哪些不同的措施, 以便贷款在未来获得批准? 前一个问题可以用“信用值太低”这样的解释来回答, 这与大多数传统的解释方法类似. 后一个问题则无法用传统的解释方法来解决, 这也构成了反事实解释的基础: 如何对 Alice 的特征向量进行尽量小的修改, 以便使得分类器的判断发生改变. 这里假设该修改是增加 1 万美元的收入, 或者是获得一个新的硕士学位, 又或者是两者兼而有之. 这个修改就是生成的反事实解释, 修改过后的特征向量也就是生成的反事实. 现在再回过头来看看以上两个问题, 问题①的答案并没有告诉 Alice 该采取什么行动, 而问题②的答案 (反事实解释) 则明确地帮助了她, 在此, 人们可以很实际地感受到反事实解释的意义[20,21].

反事实解释方法的类似于建议的作用, 也非常适合于应用在数据库的配置优化上. 我们假设对数据库的性能影响比较大的参数有 n 个, 那么数据库的配置参数数据就为 n 维向量 x_1, 每一维都代表一个参数; 工作负载数据为向量 x_2; 数据库的性能指标为标量 y (假设 y 为二分类标签, 负标签表示性能低下, 正标签表示性能优越); 通过配置参数数据 x_1 和工作负载数据 x_2 来预测数据库的性能指标 y 的模型为 f. 当 f 预测出来的 y 为负标签时, 我们可以对 x_1 求反事实 x_1' (由于是要对数据库的配置参数进行修改以优化性能指标, 所以应该对 x_1 求解反事实, x_2 仅用于辅助模型预测), 从而使得 f 的预测结果由负标签改变为正标签, 之后再根据 x_1' 的相应参数值对数据库的对应参数进行配置.

这里再用一个简单例子来对反事实解释方法如何应用于数据库配置优化做进一步说明: 假设待优化的数据库为 MySQL 数据库, 待优化的数据库配置参数为 innodb_buffer_pool_size、key_buffer_size、max_connections 等, 其中, innodb_buffer_pool_size 表示 InnoDB 类型的表和索引的最大缓存, 这个值越大, 查询的速度就会越快, 但是

当这个值太大时会影响操作系统的性能. key_buffer_size 表示索引缓冲区的大小, 这个值越大, 得到的索引就会越好处理, 但是当这个值太大时就会导致操作系统频繁换页, 降低系统性能. max_connections 表示允许连接到 MySQL 数据库的最大数量, 这个值越大, 可以支持的连接请求就越多, 但是当这个值过大时, 就会浪费内存资源甚至导致 MySQL 服务器僵死. 如果此时的配置参数数据向量为{128M, 128M, 151,…} (省略号表示其他配置参数), 预测模型的预测结果为负标签, 那么我们可以基于该向量求解反事实, 假设求得的反事实结果为{256M, 128M, 151,…}, 预测模型的标签由负标签变为了正标签, 也就是说, 可以通过将 innodb_buffer_pool_size 的值增大 128M, 以使得数据库的性能得到比较大的提高, 此时, 反事实建议的作用在数据库配置优化中得到了充分体现.

此外, 反事实在数据库配置优化中, 还有另一条优势: 在取得相似优化效果的前提下, 它可以尽可能地节省资源消耗. 在上述例子中, 我们可以发现, innodb_buffer_pool_size、key_buffer_size 以及 max_connections 的取值设置得越大, 占用的系统资源就会越多, 从而降低操作系统的性能. 而在第 1 节介绍的数据库配置优化方法里, 都没有考虑到这个问题, 它们都是追求数据库本身的性能, 而忽略了提高数据库性能的同时, 可能会对系统带来的影响. 而反事实解释方法则在优化数据库性能的同时, 兼顾了该问题, 因为反事实是在原数据的基础之上进行尽可能少的修改以达到优化数据库性能的效果, 假设原数据向量为{128M, 128M, 151,…}, 优化数据 1 为{256M, 128M, 151,…}, 优化数据 2 为{512M, 256M, 151,…}, 当优化数据 1 和优化数据 2 都能使预测结果由负标签变为正标签时, 最终的反事实会是优化数据 1, 因为它相较于优化数据 2, 对原数据的修改更少, 对系统资源的消耗也更少.

近几年来, 研究反事实解释方法的研究者们提出了很多评估反事实的指标, 比较通用的几种指标包括有效性、接近度、稀疏性、耗时性以及实际性[20,21]等. 它们的具体含义如下.

● 有效性: 有效性指的是, 确实能将预测结果改变为期望结果的反事实数目占总反事实数目的比重; 比重越高, 有效性越好.

● 接近度: 接近度指的是, 生成的反事实与原数据距离是否相近; 越接近, 接近度越好. 通常衡量反事实与原数据之间距离的方式有 L1 范数、L2 范数、马氏距离以及欧氏距离等.

● 稀疏性: 稀疏性指的是, 生成的反事实更改的特征总数要尽可能地少, 因为人们总是觉得反事实解释越简短越好, 而且简短的反事实解释也便于人们理解.

● 耗时性: 耗时性指的是, 生成反事实所需要耗费的时间; 耗费时间越短, 耗时性越好. 耗时性这一指标很便于人们理解, 因为人们总是希望能够更快地得到反事实解释.

● 实际性: 实际性指的是, 生成的反事实是否符合现实生活中的情况; 越符合现实, 实际性越好. 这一指标很重要, 例如生成的反事实中, 将年龄从 45 岁变为了 25 岁, 这显然没有任何意义, 因为人们不可能减小自己的年龄, 这一条反事实解释对于人们来说根本起不到建议的作用.

以上几个指标是目前比较通用的反事实评估指标, 其实除了上述这些, 还有其他一些指标, 它们是在特定场景中使用的, 或者和上述指标在意义上重叠度比较高, 所以在此没有进行罗列.

3 反事实解释方法的研究

本文从算法的角度来进行反事实解释方法的研究, 提出了两种反事实解释方法, 其一是基于梯度排序的反事实解释方法, 其二是基于特征重要性排序的反事实解释方法.

3.1 基于梯度排序的反事实解释方法

本文从目标函数、工作流程以及复杂度分析 3 个方面对基于梯度排序的反事实解释方法进行说明.

(1) 目标函数: 结合研究背景, 算法最终生成的反事实需要满足以下几点要求.

● 要能够确实地跨越模型预测边界, 即能够使得预测改变, 这是探求反事实最根本的目的.

● 生成的反事实与原数据之间的距离要尽可能地近, 这是因为反事实解释方法有着接近性这个属性, 反事实与原数据比较接近才有更加实际的意义.

● 不能修改不可操作特征, 这是因为反事实解释方法有着可操作性这个属性, 该属性要求在对特征进行扰动

之前, 需要事先区分出可操作特征和不可操作特征, 在整个算法流程中只能修改可操作特征.

● 生成的反事实要符合各个特征的特征域, 这是因为反事实解释方法有着实际性这个属性, 如果生成的反事实中的某个特征不处于相应的特征域中, 那么这个反事实将没有任何实际意义.

为了满足以上要求, 目标函数的设计如下.

为了达到生成的反事实能够跨越预测边界的目的, 最终的目标函数中需要包含以下部分:

$$\arg\max(f(x)) \neq \arg\max(f(x')) \tag{2}$$

其中, x 表示原数据; x' 表示生成的反事实; $\arg\max$ 表示最大值对应的下标; f 表示模型的预测函数.

为了使得生成的反事实与原数据之间的距离尽可能地近, 最终的目标函数中需要包含以下部分:

$$\min_{x'} dist(x,x') \tag{3}$$

其中, x 表示原数据; x' 表示生成的反事实; min 表示函数取最小值时, x' 的取值; $dist$ 表示距离函数, 常见的距离函数包括 L1 范数、L2 范数、马氏距离以及欧氏距离等, 在此选择的距离函数为欧氏距离, 它的计算公式如公式 (4) 所示.

$$dist(x,y) = \sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\ldots+(x_n-y_n)^2} \tag{4}$$

其中, x 与 y 表示需要进行欧氏距离衡量的两个向量, 它们的向量表示分别为 $(x_1,x_2,\ldots,x_n)$ 以及 $(y_1,y_2,\ldots,y_n)$.

为了保证生成的反事实没有修改不可操作特征, 最终的目标函数中需要包含以下部分:

$$changed(x,x') \in changeAble(X) \tag{5}$$

其中, x 表示原数据; x' 表示生成的反事实; X 表示所有数据集合; $changed$ 表示 x 与 x' 中, 发生改变的特征; $changeAble$ 表示 X 中可以修改的特征.

为了保证生成的反事实中各个特征都符合各自的特征域, 最终的目标函数中需要包含以下部分:

$$x' \in dom(X) \tag{6}$$

其中, x' 表示生成的反事实; X 表示所有数据集合; dom 表示 X 中各个特征的特征域.

综上, 结合反事实需要满足的 4 点要求, 本文最终确定的目标函数如下:

$$\begin{cases} \min\limits_{x'} dist(x,x') \quad \text{s.t. } \arg\max(f(x)) \neq \arg\max(f(x')) \\ changed(x,x') \in changeAble(X), \ x' \in dom(X) \end{cases} \tag{7}$$

其中, 整个算法的目标就是在满足预测发生改变、各个特征符合对应特征域以及只改变可修改特征 3 个条件下, 求解与原数据距离最近的反事实; 其中, s.t. 为 subject to 的缩写, 表示约束条件的意思.

(2) 工作流程: 首先确定需要进行反事实求解的数据, 这些数据需要满足一个前提条件: 原预测为负标签. 因为本文的目的是进行数据库性能优化, 也就是将吞吐量变高或者时延变低, 如果将原本预测为正标签的数据进行反事实求解, 那么生成的反事实就会让吞吐量从高变低或者是时延从低变高, 这显然不是我们所期望的.

接下来就是使用基于梯度排序的反事实解释方法对选定数据进行反事实求解, 其算法流程如算法 1.

算法 1. 基于梯度排序的反事实解释算法.

输入: $x, f, \alpha, \beta, eps, times, domain$;

输出: x'.

1. **Initialize**: $x' \leftarrow x, cnt \leftarrow 0$ #初始化反事实以及循环次数
2. **while** $cnt < times$ **do**
3. $\quad loss1, loss2 \leftarrow Loss(x, x', f)$
4. $\quad loss \leftarrow \alpha \times loss1 + \beta \times loss2$ #损失函数
5. $\quad gradient \leftarrow Backward(loss)$ #反向传播得到特征梯度
6. $\quad gradient' \leftarrow AdjustGradient(x', gradient, domain)$ #调整梯度

7. **if** $Max(Abs(gradient')) < eps$ **do** #损失函数到达极小值, 退出循环
8. **break**
9. $gradient' \leftarrow Sort(gradient')$ #基于梯度绝对值大小进行特征排序
10. $x' \leftarrow GD(x', gradient')$ #进行梯度下降
11. $x' \leftarrow AdjustFeature(x', domain)$ #将特征映射回特征域 (限制范围)
12. $cnt \leftarrow cnt + 1$
13. **return** x'

算法 1 中, 输入中的 x 表示原数据, f 表示模型的预测函数, α 表示设置在 $loss1$ 前的超参, β 表示设置在 $loss2$ 前的超参, eps 表示循环终止梯度值, $times$ 表示最大循环次数, $domain$ 表示特征域; 输出中的 x' 表示求解出的反事实; 整个算法流程中的 $Loss()$ 表示损失计算函数, $Backward()$ 表示反向传播函数, $AdjustGradient()$ 表示梯度调整函数, $Abs()$ 表示绝对值函数, $Max()$ 表示最大值函数, $Sort()$ 表示排序函数, $GD()$ 表示梯度下降函数, $AdjustFeature()$ 表示特征调整函数.

从算法 1 所示的流程图中可以看出整个算法的核心思想是梯度下降: 设置损失函数、反向传播求梯度、梯度下降等, 但是本文根据研究背景和目的对其进行了修改, 部分算法细节如下.

步骤 4 中的损失函数由两部分组成.

第 1 部分为:

$$loss1 = p_1 - p_2 \tag{8}$$

其中, p_1 为原始标签的预测概率, p_2 为目标标签的预测概率. $loss1$ 的作用是降低预测为原始标签的概率、增加预测为期望标签的概率.

第 2 部分为:

$$loss2 = dist(x, x') \tag{9}$$

其中, $dist$ 为欧氏距离函数, x 为原数据, x' 为反事实. $loss2$ 的作用是减小原数据和反事实之间的距离.

完整损失函数如公式 (10) 所示:

$$loss = \alpha \times loss1 + \beta \times loss2 \tag{10}$$

其中, α 和 β 是实验设置的超参, 用来把握 $loss1$ 和 $loss2$ 的重要性. 如果 α 比较大、β 比较小, 说明 $loss1$ 占比较大, 即倾向于改变预测; 如果 α 比较小、β 比较大, 说明 $loss2$ 占比较大, 即倾向于拉近原数据和反事实之间的距离.

整个算法的循环结束条件为损失函数到达极小值 (或到达时间步上限, 即 $cnt \geqslant times$), 这样既可以使得反事实能够跨越预测边界, 又可以尽可能缩短原数据与反事实之间的距离. 如果将损失函数单纯地设置为 $loss1$, 循环结束条件设置为反事实的预测发生改变, 其实更能达到以上目的, 但是这种做法有一个很大的弊端, 那就是分类预测概率相差过小, 比如, 生成的反事实的模型预测概率可能为 0.49 和 0.51 (0.49 为原始标签的预测概率, 0.51 为期望标签的预测概率), 这样虽然使得预测发生了改变, 但是两个预测概率之间相差过小, 以致可以说无论这条数据被预测为哪个分类都无可厚非, 显然, 这样的结果并不符合研究目的, 本文希望能够以较大的把握预测该数据属于期望标签. 如果损失函数设置为 $loss1$ 不变, 将循环结束条件修改为损失函数到达极小值, 那么又会造成新的问题: 整个梯度下降过程没有对于原数据和反事实之间的距离的控制. 损失函数设置为公式 (10), 循环结束条件设置为损失函数到达极小值 (或到达时间步上限), 不管是从理论分析的角度还是从实验验证的角度来看, 都能完美达到本文研究目的.

步骤 6 进行梯度调整的目的是: 避免死循环以及避免修改不可操作特征. 具体来说是将已经到达最大值且还将继续增大 (梯度小于 0) 的特征、已经到达最小值且还将继续减小 (梯度大于 0) 的特征以及不可操作特征 (以“workload-”开头的工作负载特征), 这 3 类特征的梯度置为 0, 将前两者梯度置为 0 的原因是避免死循环, 而将不可操作特征的梯度置为 0 的原因是避免修改不可操作特征.

步骤 9 基于梯度绝对值大小进行特征排序的目的是: 为了保证对原数据的修改尽可能少, 在循环过程中, 每次

只选择一个特征进行修改, 为此需要对特征进行排序, 选择序列第 1 个特征进行梯度下降. 而基于梯度排序的反事实解释方法是以梯度绝对值大小来对特征进行排序的, 因为扰动梯度绝对值最大的特征, 能够以较大的幅度降低损失.

(3) 复杂度分析: 假设需要为 n 条数据生成反事实, 每一条数据的特征数为 m; 在计算损失函数时, $loss2$ 需要计算原数据与反事实之间的距离, 时间复杂度为 $O(m)$; 假设反向传播和求导的时间复杂度为 $O(V)$; 梯度调整的时间复杂度为 $O(m)$; 按照梯度绝对值对特征进行排序的时间复杂度为 $O(m\log_2 m)$, 但是, 实际上在整个方法流程中只使用了排序之后的第一个元素, 也就是梯度绝对值最大的元素, 所以可以直接在特征向量中筛选出该特征, 时间复杂度为 $O(m)$; 将反事实映射回特征域的时间复杂度为 $O(m)$; 假设求得反事实平均需要执行的循环次数为 T.

综合以上分析, 整个方法的时间复杂度为 $O(nT(m+V+m+m+m)) = O(nT(V+4m)) = O(nTV+4nTm)$, 即 $O(nTV+nTm)$.

3.2 基于特征重要性排序的反事实解释方法

本文从相关技术、工作流程以及复杂度分析 3 个方面对基于特征重要性排序的反事实解释方法进行说明 (目标函数的设置与基于梯度排序的反事实解释方法相同, 这里不再进行赘述).

(1) 相关技术: 模型可解释方法可以分为: ① 直接使用可解释的、透明的模型; ② 为模型提供事后解释这两种, 其中事后解释方法又可以分为模型特定方法以及模型不可知方法. 模型特定方法的主要代表包括特征重要性方法和模型简化方法; 模型不可知方法的主要代表包括视觉解释方法、局部解释方法、特征重要性方法以及模型简化方法[20]. 在基于特征重要性的方法中所用到的 LIME (local interpretable model-agnostic explanations, 局部可理解的与模型无关的解释技术)[22]方法就是属于模型不可知方法中的局部解释方法 (或者可以说是局部解释方法和模型简化方法的综合).

这里首先可以从名字的角度对 LIME 方法进行分析.

- Local interpretable: 它表明整个方法是从局部的角度来对数据点进行解释.
- Model-agnostic explanations: 它表明整个方法与模型无关, 可以用于处理黑盒模型.

LIME 方法的整体算法流程如下.

首先, 对待解释模型进行数据的训练, 这里假设经过充分拟合之后的模型的数据图如图 5 所示, 蓝色区域和橙色区域的边界就是模型的决策边界 (二分类问题, 蓝色区域和橙色区域分别代表一个预测分类, 假设蓝色区域代表的分类为类 1, 橙色区域代表的分类为类 2).

然后确定待解释的数据点, 这里假设确定的数据点为图 6 中的红点, 那么接下来将要进行的工作, 就是对这个红点被预测为类 2 (橙色区域) 而不是类 1 (蓝色区域) 进行解释. LIME 方法采取的解释手段为, 给出一段文本或一个可视化的数据分析图, 上面显示了, 在预测过程中, 各个特征起到的作用, 这个作用包括: 对预测起正向作用还是反向作用、具体作用是大还是小等.

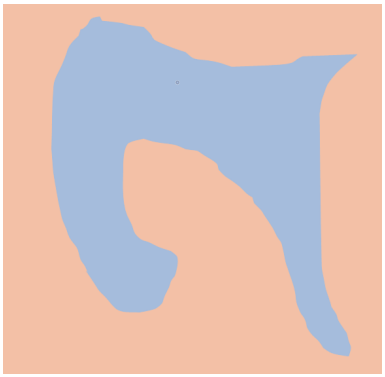

图 5 训练待解释模型

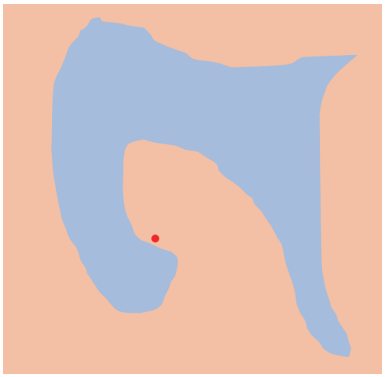

图 6 确定待解释点

从图 6 中可以看到, 模型的决策边界非常复杂, 因此要从全局的角度来进行解释是相当困难的, 但是如果将边界分成一个个小的局部来看待的话, 复杂性就会小很多. 因此, LIME 方法会在待解释数据点周边进行采样, 得到多个数据点 (如图 7 中的黑色点所示), 为了保证采样点是在待解释数据点的附近, LIME 方法采取对待解释数据点进行扰动来获取采样点的方式, 至于采样点的数量, 可以视具体情况而定.

接下来用待解释模型对采样点进行预测分类, 如图 8 所示, 深棕色点是被预测为类 2 的点, 而深蓝色点是被预测为类 1 的点. 然后根据这些采样点与待解释数据点之间的距离 (距离公式的选择可以是 L1 范数、L2 范数、马氏距离以及欧氏距离等), 赋予不同的权重, 与待解释数据点的距离越近, 权重越大; 与待解释数据点的距离越远, 权重越小 (体现在图 8 中就是, 距离越近, 点的大小越大; 距离越远, 点的大小越小).

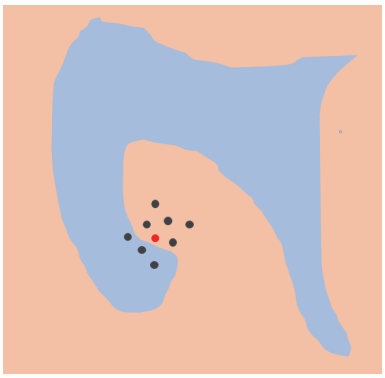

图 7　采样数据点

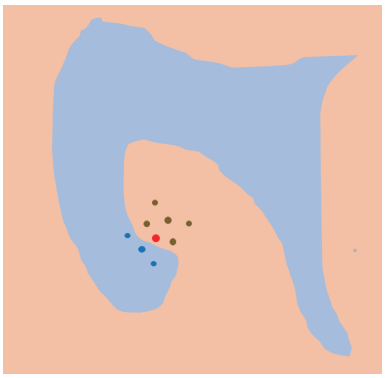

图 8　预测数据点

最后使用一个简单的、可解释的模型, 对所有点 (包括采样点与待解释数据点) 进行拟合, 同时, 在拟合过程中, 会更加看重权重更高, 即距离更近的点. 拟合完成之后, 通过该简单模型的可解释性, 就可以得到各个特征的作用和重要性, 因为 LIME 方法只是在待解释数据点的周边进行采样, 所以这是从局部的角度来对数据点进行解释的方法. LIME 方法中使用的简单模型为线性模型 (如图 9 中的虚线所示), 最后拟合完成的线性模型中, 各个特征前的权重, 就可以描述为相应特征的重要性, 通过这些重要性, LIME 方法就能够得出模型在预测的过程中, 更加看重哪些特征. 实际上, 除了线性模型, 还有很多简单的、带有良好解释性的模型, 例如决策树等, 研究者可以根据实际情况来进行模型的选择.

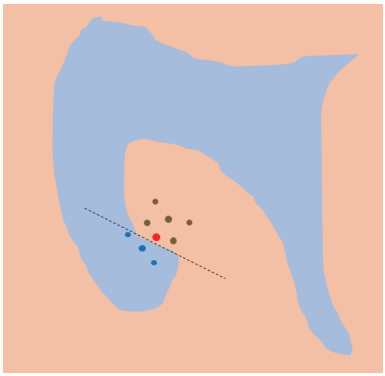

图 9　拟合可解释模型

LIME 方法可以用于各种类型的数据, 例如, 如果是表格数据的话, LIME 方法可以得出是哪些特征在决策过程中起主要作用; 如果是图像数据的话, LIME 方法可以得出是哪几个像素块在决策过程中起主要作用; 如果是文本数据的话, LIME 方法可以得出是哪几个关键词在决策过程中起主要作用.

LIME 方法虽然不属于反事实解释方法, 但是却能很好地应用于基于特征重要性排序的反事实解释方法.

(2) 工作流程: 首先确定需要进行反事实求解的数据, 这些数据同样需要满足原预测为负标签的前提条件. 接下来就是使用基于特征重要性排序的反事实解释方法对选定数据进行反事实求解, 其算法流程如算法 2.

算法 2. 基于特征重要性排序的反事实解释算法.

输入: $x, f, \alpha, \beta, eps, times, domain, tur\_dis, tur\_con, explainer$;
输出: x'.

1. **Initialize**: $x' \leftarrow x, cnt \leftarrow 0, map\_ \leftarrow \{\}$ #初始化反事实、循环次数以及特征记录字典
2. **while** $cnt < times$ **do**
3. $loss1, loss2 \leftarrow Loss(x, x', f)$
4. **if** $loss1 < eps$ **do** #损失 1 达到循环结束条件, 退出循环
5. **break**
6. $loss \leftarrow \alpha \times loss1 + \beta \times loss2$ #设置损失函数 (仅作为衡量算法优劣的指标)
7. $f' \leftarrow Set(f)$ #设置预测函数
8. $explanation \leftarrow LIMEExplain(x', f', explainer)$ #得到解释文本
9. $features \leftarrow Filter(explanation, map\_)$ #进行特征筛选
10. $x' \leftarrow Disturb(x', features, tur\_dis, tur\_con)$ #选取重要性最大的特征进行扰动
11. $x' \leftarrow AdjustFeature(x', domain)$ #将特征映射回特征域 (限制范围)
12. $map\_ \leftarrow Update(map\_, features)$ #更新特征记录字典
13. $cnt \leftarrow cnt + 1$
14. **return** x'

算法 2 中, 输入中的 x 表示原数据, f 表示模型的预测函数, α 表示设置在 $loss1$ 前的超参, β 表示设置在 $loss2$ 前的超参, eps 表示循环终止 $loss1$ 值, $times$ 表示最大循环次数, $domain$ 表示特征域, tur_dis 表示离散特征扰动值, tur_con 表示连续特征扰动值, $explainer$ 表示解释器; 输出中的 x' 表示求解出的反事实; 整个算法流程中的 $Loss()$ 表示损失计算函数, $Set()$ 表示预测设置函数, $LIMEExplain()$ 表示解释函数, $Filter()$ 表示特征筛选函数, $Disturb()$ 表示扰动函数, $AdjustFeature()$ 表示特征调整函数, $Update()$ 表示更新函数.

从算法 2 所示的流程图中可以看出整个算法的核心思想是不断对特征进行扰动, 直到达到循环结束条件, 部分算法细节如下.

步骤 6 中的损失函数设置与基于梯度排序的反事实解释方法相同, 也为公式 (10) $loss = \alpha \times loss1 + \beta \times loss2$, 不过此处的损失函数并不用于反向传播、求导, 而是仅作为一个衡量算法优劣的指标.

步骤 7 是设置 LIME 方法的预测函数. 因为 LIME 方法在对一个实例进行局部解释时, 需要采集多个采样点, 然后用预测函数对这些采样点进行预测分类, 因此这里需要对其进行设置. 这里的预测函数, 即 ResNet18 的预测函数, 不过, 因为数据类型不同, 需要对该函数进行包装, 在包装中进行数据类型转换.

步骤 9 特征筛选的目的与基于梯度方法的步骤 6 相同, 都是为了避免死循环以及避免修改不可操作特征.

步骤 10 是从经过筛选的 LIME 解释文本中, 选取特征重要性最大的特征进行扰动, 与基于梯度的方法不同, 此处是从另一个角度来选取扰动特征, 从而达到以较大幅度降低损失的目的.

整个算法流程的循环结束条件为: $loss1$ 小于 eps (或循环次数达到上限). eps 设置得较小的话, 会导致反事实与原数据的距离较远; eps 设置得较大的话, 又会导致预测分类概率相差过小的问题. 因此, 后续经过反复实验, 选取了一个能较好地兼顾两方面的 eps.

(3) 复杂度分析: 假设需要为 n 条数据生成反事实, 每一条数据中的特征数为 m; 在计算损失函数中的 $loss2$ 时, 需要计算原数据与反事实之间的距离, 时间复杂度为 $O(m)$; 在进行 LIME 解释时, 需要采集多个采样点, 然后拟合

线性模型, 采样点的个数对 LIME 解释的复杂度影响非常大, 这里假设在 5 000 个采样点 (LIME 方法默认采样点的个数, 也是实验过程中设置的采样点的个数) 的情况下, LIME 解释的时间复杂度为 $O(X)$; 由于 LIME 方法生成的解释文本本身就是按照特征重要性从大到小排序的, 所以这里不需要再对特征进行排序; 筛选特征的时间复杂度为 $O(m)$; 将反事实映射回特征域的时间复杂度为 $O(m)$; 假设求得反事实平均需要执行的循环次数为 T.

综合以上分析, 整个方法的时间复杂度为 $O(nT(m+X+m+m))=O(nT(X+3m))=O(nTX+3nTm)$, 即 $O(nTX+nTm)$.

3.3 两种方法的对比

● 两种方法的联系和区别: 基于梯度的方法和基于特征重要性的方法, 二者的核心思想是一样的, 本质上都是选取对预测结果影响最大的特征进行扰动, 以达到改变预测的目的. 它们之间的不同点在于: 选取标准不同. 基于梯度的方法是对损失函数进行反向传播之后, 选取梯度绝对值最大的特征作为影响最大的特征; 而基于特征重要性的方法利用 LIME 方法, 求解出每一次迭代过程中各个特征的重要性, 选取特征重要性最大的特征作为影响最大的特征.

● 两种方法的优劣: 基于特征重要性的方法和基于梯度的方法, 前者时间复杂度中的 $O(X)$ 是进行 LIME 解释的时间复杂度, 在此过程中, 需要采样数据点以及拟合线性模型, 花费时间较长; 而后者复杂度中的 $O(T)$ 是反向传播、求导的时间复杂度, 它会明显小于 $O(X)$. 因此基于特征重要性的方法要比基于梯度的方法更加耗时; 但是后续得到的实验结果显示, 基于特征重要性的方法求解得到的反事实的有效性要优于基于梯度的方法, 即前者求解的反事实成功改变预测的比例更高.

4 实 验

本文实验包含了 4 部分内容, 其一是实验所用数据集的采集, 分别从使用工具、待采集数据以及采集流程 3 个方面进行说明; 其二是实验所用模型的搭建、训练和测试, 此部分对 MLP 和 ResNet18 两个模型进行了实验比较; 其三是反事实评估实验, 它从多种对比方法、多种评价指标的角度出发, 对本文提出的反事实解释方法所生成的反事实进行了评估; 其四是测试反事实优化效果实验, 它将反事实应用到数据库配置中, 测试了反事实对于数据库性能优化的效果.

4.1 数据采集

由于目前并没有数据库配置优化相关的开源数据集, 之前的相关工作也没有把实验过程中所使用到的数据进行公开, 而本文的实验需要使用大量的数据库配置优化相关的数据来进行模型的训练以及反事实的求解, 所以为了能够顺利完成后续实验, 我们决定自己进行数据的采集, 接下来将会从使用工具、待采集数据以及采集流程 3 方面对数据采集进行说明.

4.1.1 使用工具

本文主要使用 MySQL 数据库和 YCSB 性能测试工具来进行数据集的采集和生成.

MySQL 是当前最热门的关系型数据库系统, 在 Web 应用方面它也是目前最佳的关系数据库管理系统 (relational database management system, RDBMS) 应用之一. MySQL 由瑞典 MySQL AB 公司开发, 目前属于 Oracle 集团. MySQL 作为一个关系型数据库系统, 它使用不同的表来实现数据的存储, 而不是直接将全部数据都存储到大仓库里, 这既增加了速度也提高了灵活性[23].

本文使用 MySQL 数据库作为采集和生成数据集的基础工具, 最主要的原因在于其配置参数相比于其他数据库来说较少, 这可以大大简化采集过程 (MySQL 有几百个配置参数, Oracle 则有上千个配置参数)[1].

YCSB (Yahoo! cloud serving benchmark) 是 Yahoo 公司的一个用来对云服务进行基础测试的工具. YCSB 本身相当于客户端, 不断向服务器发送请求, 同时会记录下数据库完成这些请求时的性能指标.

YCSB 支持自己设计工作负载, 工作负载的一些常见参数如下.

● recordcount: 数据表中的记录数目.

● operationcount: 工作负载的操作数.

● readproportion: 工作负载中, 读操作数所占总操作数的比重.

● updateproportion: 工作负载中, 更新操作数所占总操作数的比重.

● insertproportion: 工作负载中, 插入操作数所占总操作数的比重.

● requestdistribution: 工作负载中, 请求在键空间中的分布, requestdistribution 有 3 种离散取值, 分别是 zipfian、uniform 以及 latest.

YCSB 对数据库进行测试时, 分为两个阶段: load 和 run[24].

load 阶段是初始化测试数据阶段, 实际上就是给数据库提前装载一些数据, 以方便后续 run 阶段的执行. 例如, 如果 run 阶段需要执行读操作和更新操作的话, 数据库中没有记录, 那么这两个操作将会没有意义, 所以 YCSB 会在 load 阶段提前给数据库装载一些数据.

run 阶段是执行测试阶段, 它会根据工作负载的设置, 对数据库不断发送请求, 并记录下数据库的性能. 例如, 如果工作负载的设置是: recordcount 为 1 000, operationcount 为 1 000, readproportion 为 0.5, updateproportion 为 0.25, insertproportion 为 0.25, requestdistribution 为 zipfian; 那么 run 阶段将会执行 500 条左右的读操作, 250 条左右的更新操作, 250 条左右的插入操作, 这些请求在键空间中的分布为 zipfian.

4.1.2 待采集数据

整个实验需要采集 3 类数据, 分别是数据库配置数据、工作负载数据以及性能数据.

MySQL 数据库中的配置参数多达上百个, 如果全部采集的话, 将会对之后模型的建立以及反事实的求解造成很大的困难, 因为采集的配置参数过多会导致神经网络模型的输入特征增多, 进而使得整个模型变得更复杂、占用内存变得更多以及训练和测试时间变得更长; 除此之外, 特征数增多还会导致在之后求解反事实的过程中, 需要探究的反事实空间变大, 求解反事实的时间变长. 为了避免以上问题, 本文挑选了数据库中比较重要的 26 个配置参数进行采集[25].

另外, 在尝试对数据库配置参数进行修改的过程中发现, 部分参数的取值是非连续的 (例如 innodb_buffer_pool_size, 它可以被赋值为 8 388 608、16 777 216, 但是却不可被赋值为 8 388 609, 8 388 610, 16 777 217, 16 777 218 等), 这意味着这类参数在实验过程中, 不能被视为连续参数, 而应该被视为离散参数.

部分数据库配置参数及其类型和取值如表 1 所示.

表 1 部分配置参数信息表

序号	参数名称	类型	取值范围
00	innodb_buffer_pool_size	离散	8 388 608, 16 777 216, 33 554 432, 67 108 864, 134 217 728, 268 435 456, 536 870 912, 1 073 741 824, 2 147 483 648, 4 294 967 296, 8 589 934 592
01	innodb_thread_sleep_delay	连续	0–1 000之间的随机整数
02	innodb_thread_concurrency	连续	0–950之间的随机整数
03	innodb_flush_log_at_trx_commit	离散	0, 1, 2
04	binlog_cache_size	离散	32 768, 16 384, 8 192, 4 096, 65 536, 131 072, 262 144, 524 288, 1 048 576, 2 097 152, 4 194 304, 8 388 608, 16 777 216, 33 554 432

本文针对 YCSB 中常用的 6 种工作负载参数, 即 recordcount、operationcount、readproportion、updateproportion、insertproportion 以及 requestdistribution, 设计了 16 种用于测试 MySQL 数据库的工作负载, 部分工作负载设置如表 2 所示.

为了能够达到优化数据库性能的目的, 数据集中还需要包含能够反映数据库性能的指标, 而 YCSB 在测试数据库时, 会返回相应的数据库性能指标, 本文将这些指标采集到数据集中作为性能数据.

● Throughput: 吞吐量; 吞吐量越大, 性能越好.

● RunTime: 总运行时间; 总运行时间越短, 性能越好.

- ReadLatency: 读操作的平均时延; 读操作的平均时延越短, 性能越好.
- UpdateLatency: 更新操作的平均时延; 更新操作的平均时延越短, 性能越好.
- InsertLatency: 插入操作的平均时延; 插入操作的平均时延越短, 性能越好.

表 2　部分工作负载信息表

name	recordcount	operationcount	readproportion	updateproportion	insertproportion	requestdistribution
workload000	1 000	1 000	0.5	0.5	0	zipfian
workload001	1 000	10 000	0.4	0.3	0.3	latest
workload002	10 000	1 000	0.5	0.5	0	zipfian
workload003	10 000	10 000	0	0.5	0.5	latest
workload004	10 000	10 000	0	0.95	0.05	uniform

4.1.3　采集流程

整个数据集采集流程为: (1) 根据数据库配置参数及其取值设置数据库; (2) 运行 YCSB 中的各个工作负载对数据库进行测试; (3) 将数据库配置参数数据、工作负载数据以及 YCSB 记录的性能数据收集到数据集中; (4) 清除数据库中的记录, 并重复步骤 (1).

在步骤 (1) 中, 不会把所有参数组合全部设置一遍, 因为这样做会导致数据量急剧增加. 例如, 如果每一个配置参数有 3 个可选值, 要列举出所有参数组合的情况的话, 7 个配置参数构成的组合就会达到 2 000 以上, 然后再考虑负载, 数据集将会非常大, 而且在这种情况下, 参数的个数和可选值都不是很充分 (只有 7 个配置参数, 每个配置参数只有 3 种值的选择). 因此, 为了使参数数目和取值更加丰富, 同时保证数据集不会过大, 本文决定采取随机组合的方式, 即每一次进行数据库配置时, 给每一个配置参数分配一个取值范围内的随机值, 这样既可以保证数据集的充分性, 也能保证数据集不至于过大.

执行步骤 (4) 的原因是, YCSB 的 load 阶段会装载一些数据进来, 如果在 run 阶段结束后, 不把这些数据清除掉的话, 那么后续进行新一轮 YCSB 测试时, load 阶段装载的数据可能会与之前的数据发生主键冲突.

我们将最终生成的数据集命名为 databaseConfiguration 数据集, 其包含 13 088 条数据, 每一条数据由 26 个数据库配置参数特征、7 个工作负载特征以及 5 个性能指标标签构成.

4.2　模型搭建、训练和测试

基于第 2.1 节中对于神经网络模型的介绍, 同时考虑到实验数据集, 即 databaseConfiguration 数据集为表格数据集, 本文决定分别搭建、训练和测试 MLP 和 CNN, 并对两者进行分析和比较.

对于 MLP, 本文使用 PyTorch 框架, 搭建了如图 10 所示的模型.

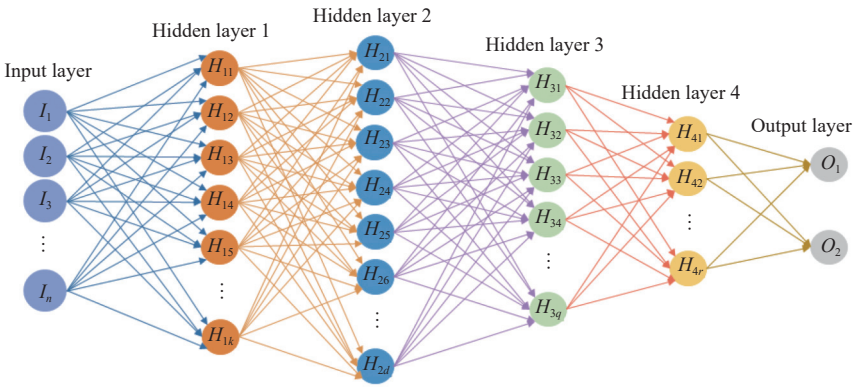

图 10　MLP 结构图

由图 10 可得, 该 MLP 一共有 1 个输入层, 4 个隐藏层以及 1 个输出层. 其中, 输入层神经元数为 169; 隐藏层神经元数分别为 500、1 000、400 以及 50, 将各隐藏层之间的神经元数目变化设置得比较平缓, 是为了能有更好

的训练效果[26]; 输出层神经元数为 2.

MLP 的训练和测试过程, 大致可以分为 3 个步骤: 特征预处理、标签预处理以及实际的训练和测试.

(1) 特征预处理: databaseConfiguration 数据集包含 33 个特征 (26 个以“parm-”开头的数据库配置参数特征以及 7 个以“workload-”开头的工作负载特征) 和 5 个标签 (以“metric-”开头的性能指标标签), 其中, 特征又分为连续特征和离散特征, 对于这两类特征的处理需要分开进行.

对于离散特征, 本文使用 one-hot 编码处理, 即使用只有一个位置为 1, 其他位置均为 0 的向量来表示该离散值, one-hot 编码会让特征数增加, 这使得 MLP 的输入特征数不是 33 而是 169 (进行 one-hot 处理过后的特征数); 而对于连续特征, 由于连续特征之间的数量级差距较大, 为了防止预测结果被数量级大的特征主导, 所以会对连续特征进行归一化操作.

(2) 标签预处理: 由于 5 个性能指标标签, 即 Throughput、RunTime、ReadLatency、UpdateLatency 以及 InsertLatency, 取值都为连续值, 而反事实解释方法主要针对分类任务, 所以需要把连续标签处理成类别标签.

本文采取的处理方法为: 记录数据集中不同负载下性能指标的中位数, 然后将相应负载下的指标小于等于中位数的标签置为分类 0, 大于中位数的标签置为分类 1.

(3) 训练和测试: 将经过数据预处理的数据按照 3∶1 的比例划分训练集和测试集, 然后将训练集和测试集装载进批量大小设置为 256 的 DataLoader 容器中, 然后使用小批量梯度下降算法 (MBGD) 进行 MLP 的训练.

针对标签 Throughput, MLP 的训练准确率为 0.85 左右, 测试准确率为 0.81 左右, 而其在训练过程中的损失-时间图如图 11 所示.

如图 11 所示, MLP 的整体损失是随着时间步的推移而下降的, 不过在最后阶段有上升趋势.

对于 CNN, 本文决定搭建 ResNet, 更加具体地说是 ResNet18, 其结构图如图 12 所示.

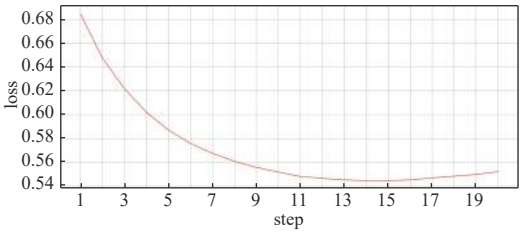

图 11 MLP 损失-时间图

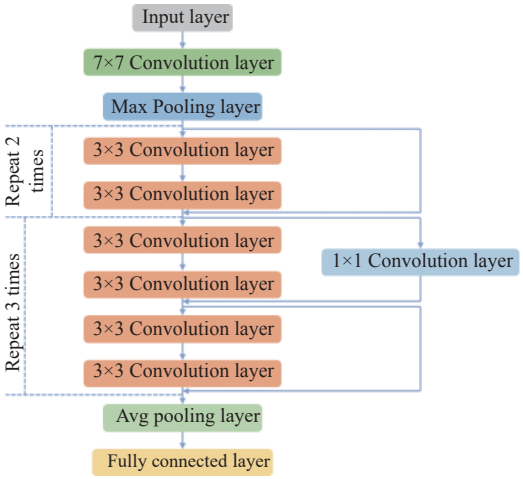

图 12 ResNet18 结构图

从图 12 中可以清晰地看到, ResNet18 一共包含 17 个 (1+2×2+3×4) 卷积层、2 个池化层、1 个全连接层以及一些在图中没有显示的激活层和批归一化 (batch normalization, BN) 层.

ResNet18 的训练和测试过程与 MLP 类似, 也分为特征预处理、标签预处理以及实际训练和测试 3 个步骤, 但是由于 ResNet18 的标签预处理部分与 MLP 基本相同, 所以此处只对特征预处理和实际训练和测试两个步骤进行说明.

● 特征预处理: ResNet18 用于处理图像数据, 它的输入维度为 4 维, 分别是批量维、通道维、长度维以及宽度维, 因为批量维可以在训练和测试中通过设置 batch_size 得到, 通道维初始可以设置为 1 维, 所以在特征预处理阶段需要将数据转换到 2 维 (长度维以及宽度维), 本文采取的处理方法是, 将一维数据直接 reshape 成二维数据, 这

种处理方式在后续实验过程中被证明是有效的.

● 训练和测试: 与 MLP 类似, ResNet18 的训练集和测试集的比例划分为 3:1, DataLoader 容器的批量大小设置为 256, 梯度下降算法选择 MBGD.

针对标签 throughput, ResNet18 的训练准确率为 0.83 左右, 测试准确率为 0.81 左右, 而其在训练过程中的损失-时间图如图 13 所示.

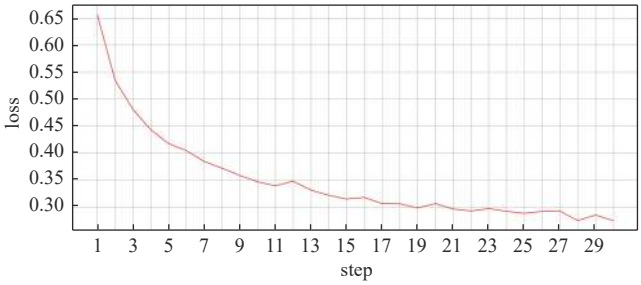

图 13 ResNet18 损失-时间图

如图 13 所示, ResNet18 的整体损失是随着时间步的推移而下降的, 与 MLP 不同的是, 它在最后阶段并没有上升趋势.

对 MLP 和 ResNet18 进行对比: 从准确率的角度来看, 它们的准确率相差并不大, 对于标签 Throughput, MLP 的训练集准确率为 0.85 左右, 测试集准确率为 0.81 左右; ResNet18 的训练集准确率为 0.83 左右, 测试集准确率为 0.81 左右. 从训练和测试的速度的角度来看, 它们的差距较大, ResNet18 的训练速度是 MLP 训练速度的 3 倍到 4 倍, 如果选择 ResNet18 作为后续的模型, 能够为实验节约大量时间. 从训练的稳定性的角度来看, 它们也有所差距, MLP 在训练过程的最后阶段中, 损失有上升趋势, 而 ResNet18 并没有出现类似不良情况. 经过上述 3 方面的比较, 本文最终决定使用 ResNet18 作为预测模型.

4.3 反事实评估

本文从实验方法介绍、评估指标以及对比结果 3 方面对反事实评估实验进行说明.

(1) 实验方法介绍: 本文选择了 4 种反事实解释方法作为实验对比方法, 其中 3 种是近几年来提出的非常具有创新性的方法, 并且在各自的论文和实验数据集中, 都有着十分优异的表现. 另外一种是本文在研究基于梯度排序的反事实解释方法时, 探究的变更损失函数和循环终止条件的情况. 本文决定以这 4 种方法在 databaseConfiguration 数据集上生成的反事实作为对比, 来对我们提出的方法生成的反事实进行评估.

4 种实验对比方法再加上我们的方法, 本文一共需要对以下 6 种方法进行实验评估.

● CEM[27]: 此方法通过找到待解释数据点的 PP (至少足以证明最终分类的合理性的因素, 换言之, 待解释数据点正是因为有了 PP 才会被模型预测为相应分类) 和 PN (在确定最终分类时, 必须缺少的因素, 换言之, 待解释数据点正是因为没有 PN, 才会被预测为相应分类), 来对模型进行解释. 在实验过程中, 只会使用到 PN, 因为原数据点加上 CEM 找到的 PN, 就是相应的反事实, 这里使用该反事实作为对比基准.

● Proto[28]: 此方法针对待解释数据点, 使用自动编码器 (autoencoder) 或者 K-D 树 (K-D trees) 为每个类构建原型, 然后使用与最初预测不同的最近的类来指导反事实的求解过程, 从而加速得到相应的反事实.

● GRACE[9]: 此方法在反事实求解过程中, 针对待解释数据点的特征向量进行排序, 选择最前面的 k 个 (这里的 k 会逐渐增加, 该方法旨在通过修改最少的特征数来求得反事实) 特征进行扰动, 以改变模型预测.

● Ours_Gradient_Contrast: 基于梯度排序反事实解释方法的对比版本, 该方法单纯地将损失函数设置为 $loss1$, 循环结束条件设置为反事实的预测发生改变.

● Ours_Gradient: 基于梯度排序的反事实解释方法.

● Ours_Importance: 基于特征重要性排序的反事实解释方法.

以上方法中, CEM、Proto、GRACE 以及 Ours_Gradient_Contrast 是实验对比方法; Ours_Gradient 以及 Ours_Importance 是本文提出的方法.

(2) 评估指标: 结合本文的研究背景, 我们使用如下 5 个指标作为实验评估反事实的标准.

- 可靠性: 反事实改变模型预测后, 预测分类和其他分类之间的概率差值是否较大.
- 有效性: 能真正将预测结果改变为期望结果的反事实, 占所有数据的比重是否较大.
- 接近度: 生成的反事实与原数据之间的距离是否相近.
- 耗时性: 生成反事实所需要耗费的时间是否较少.
- 实际性: 生成的反事实是否符合实际情况.

以上指标中, 第 1 个指标, 即可靠性, 是本文根据反事实的实际应用提出的评估指标. 在实际情况中, 有些反事实虽然能够改变模型预测, 但是改变之后, 分类之间的预测概率之差太小, 实际意义不大, 以二分类问题为例, 将预测从 [0.51, 0.49] 改变为 [0.49, 0.51], 虽然模型预测分类从 1 号分类变为了 2 号分类, 但是 1、2 号分类预测差距太小, 以致这条数据无论被预测为哪个分类都无可厚非, 即预测不够可靠. 基于以上考虑, 决定引入可靠性这一指标来对此进行衡量.

而后面 4 个指标, 即有效性、接近度、耗时性以及实际性, 是本文根据研究实际情况以及数据库配置优化的相关背景, 从评估反事实的常用指标中挑选出来的.

(3) 对比结果: 实验的待优化指标分为 Throughput (吞吐量)、RunTime (运行总时长)、ReadLatency (读取操作时延)、UpdateLatency (更新操作时延) 以及 InsertLatency (插入操作时延) 等 5 类. 针对每一类优化指标, 本文会随机抽取 30 条数据进行反事实的求解, 实验结果如下 (为了能更快地求解出反事实, 基于梯度排序的反事实解释方法的循环终止条件在实际实验中, 设置为损失函数到达极小值或小于某定值).

在表 3–表 7 这 5 个实验结果的表格中, Method_name 表示实验方法的名称; Avg_fidelity 表示使预测成功发生改变的反事实所占比例; Avg_dif 表示在预测成功改变的前提下, *loss*1 的平均值; Avg_distance 表示成功改变预测的反事实与原始数据之间的平均欧氏距离, 即 *loss*2 的平均值; Avg_time_cost 表示在生成的反事实成功改变模型预测的前提下, 其生成所消耗的平均时间 (注: 00 min 00 s 表示该 Avg_time_cost 不足 1 s).

- Throughput: 针对 Throughput 指标的对比实验结果如表 3 所示.

表 3 Throughput 反事实对比实验结果

Method_name	Avg_fidelity	Avg_dif	Avg_distance	Avg_time_cost
Ours_Gradient_Contrast	1.0	−0.001 995 025	1.959 150 452	00 min 22 s
Ours_Gradient	0.8	−0.381 859 413	2.185 275 010	00 min 42 s
Ours_Importance	0.933 333 333	−0.395 551 437	2.095 657 835	06 min 51 s
CEM	1.0	−0.221 581 434	2.347 991 063	00 min 35 s
Proto	1.0	−0.078 611 674	2.136 516 186	03 min 01 s
GRACE	0.466 666 666	−0.336 858 919	1.417 664 253	00 min 00 s

- RunTime: 针对 RunTime 指标的对比实验结果如表 4 所示.

表 4 RunTime 反事实对比实验结果

Method_name	Avg_fidelity	Avg_dif	Avg_distance	Avg_time_cost
Ours_Gradient_Contrast	0.966 666 666	−0.002 305 337	1.569 737 253	00 min 26 s
Ours_Gradient	0.833 333 333	−0.372 228 517	1.818 955 804	00 min 35 s
Ours_Importance	0.9	−0.363 759 469	2.087 755 354	07 min 39 s
CEM	1.0	−0.225 119 666	2.251 033 532	00 min 35 s
Proto	1.0	−0.026 153 548	1.773 651 919	02 min 54 s
GRACE	0.433 333 333	−0.310 911 325	0.838 141 195	00 min 00 s

- ReadLatency: 针对 ReadLatency 指标的对比实验结果如表 5 所示.

表 5 ReadLatency 反事实对比实验结果

Method_name	Avg_fidelity	Avg_dif	Avg_distance	Avg_time_cost
Ours_Gradient_Contrast	0.966 666 666	−0.003 486 569	1.776 159 210	00 min 24 s
Ours_Gradient	0.8	−0.365 956 849	1.874 996 736	01 min 05 s
Ours_Importance	0.9	−0.427 493 422	1.516 062 605	02 min 57 s
CEM	1.0	−0.247 041 918	1.819 862 623	00 min 46 s
Proto	1.0	−0.051 833 037	1.361 167 536	02 min 04 s
GRACE	0.7	−0.060 104 960	1.694 645 555	00 min 00 s

● UpdateLatency: 针对 UpdateLatency 指标的对比实验结果如表 6 所示.

表 6 UpdateLatency 反事实对比实验结果

Method_name	Avg_fidelity	Avg_dif	Avg_distance	Avg_time_cost
Ours_Gradient_Contrast	0.966 666 666	−0.003 801 067	1.476 540 842	00 min 19 s
Ours_Gradient	0.833 333 333	−0.397 313 281	1.657 900 735	00 min 31 s
Ours_Importance	0.9	−0.397 028 084	1.412 871 501	01 min 49 s
CEM	1.0	−0.223 959 004	2.626 857 125	00 min 32 s
Proto	1.0	−0.044 401 870	2.073 417 938	02 min 25 s
GRACE	0.366 666 666	−0.356 390 335	1.407 531 528	00 min 00 s

● InsertLatency: 针对 InsertLatency 指标的对比实验结果如表 7 所示.

表 7 InsertLatency 反事实对比实验结果

Method_name	Avg_fidelity	Avg_dif	Avg_distance	Avg_time_cost
Ours_Gradient_Contrast	1.0	−0.002 727 061	1.729 411 528	00 min 38 s
Ours_Gradient	0.866 666 666	−0.336 224 151	1.815 373 890	01 min 09 s
Ours_Importance	0.966 666 666	−0.348 820 760	2.011 119 552	05 min 56 s
CEM	1.0	−0.228 429 800	2.358 683 969	00 min 45 s
Proto	1.0	−0.044 245 669	1.943 359 036	02 min 13 s
GRACE	0.466 666 666	−0.455 742 501	1.345 421 793	00 min 00 s

结合以上实验结果, 本文分别从可靠性、有效性、接近度、耗时性以及实际性 5 个评估指标的角度来对各个方法进行评估对比.

● Ours_Gradient_Contrast: 从 5 组实验数据可以分析得到, 虽然它的平均欧氏距离比 Ours_Gradient 小, 有效性也非常趋近于 1. 但是它的可靠性非常低 (其 Avg_dif 是所有方法中最大的), 也就是说它生成的反事实的预测结果均类似于 [0.5001, 0.4999]、[0.5002, 0.4998], 这些反事实在实际应用中基本没有意义.

● Ours_Gradient: 从 5 组实验数据可以分析得到, 它的可靠性 (Avg_dif 很小) 很高, 绝大多数情况下其可靠性与 Ours_Importance 位于前 2 位, 更重要的是, 它能在保证可靠性如此之高, 即能大幅度改变预测的情况下, 还能保证平均欧氏距离与 CEM 方法以及 Proto 方法相近甚至低于此两种方法; 平均消耗时间略高于 CEM 方法、远低于 Proto 方法. 而且此方法考虑实际性, 生成反事实的各个特征都符合各自特征域. 它的不足之处在于有效性略低, 但其平均有效性高于 80% 也与本实验的预期相符.

● Ours_Importance: 从 5 组实验数据可以分析得到, 它的可靠性与 Ours_Gradient 类似, 同样很高, 而且也能在保证可靠性的情况下, 同时保证平均欧氏距离与 CEM 方法以及 Proto 方法相近甚至低于此两种方法; 而且 Ours_Importance 同样也考虑到了实际性. 更值得一提的是, 它的有效性相比于 Ours_Gradient, 明显得到了改善, 平均有效性达到了 90% 以上. 其唯一的缺陷在于求解反事实所消耗的平均时间较长.

● CEM: 从 5 组实验数据可以分析得到, 它最大的优势在于有效性, 能够达到 100%, 而且所消耗的平均时间较短且比较稳定. 其缺陷在于可靠性不足, 它的 Avg_dif 与 Ours_Gradient、Ours_Importance 以及 GRACE 方法相比

不够小, 最终得到的反事实的预测结果类似于 [0.61, 0.39], 在实际应用中也有所局限; 另外该方法也没有考虑到实际性, 它并没有对特征的取值加以限制.

● Proto: 从 5 组实验数据可以分析得到, 它最大的优势与 CEM 方法类似, 也是在于其 100% 的有效性. 其缺陷有很多, 首先是可靠性, 它的 Avg_dif 仅小于 Ours_Gradient_Contrast, 最终得到的反事实预测结果类似于 [0.52, 0.48], 在实际应用中有很大的局限性; 其次, 该方法的生成反事实的平均消耗时间很长, 其平均耗时仅低于 Ours_Importance; 最后, 该方法同样没有考虑到实际性.

● GRACE: 从 5 组实验数据可以分析得到, 它的有效性很低, 大部分情况下, 它的有效性都低于 50%, 虽然它的可靠性、接近度、耗时性都非常出色, 并且对于特征的特征域限制也有考虑, 但是, 其出色的可靠性、接近度以及耗时性很有可能是建立在有效性不高的基础之上, 例如耗时性, 有超过一半的数据都没有成功求解出反事实, 其反事实求解过程便已经结束, 其平均计算时间自然会低. 所以其出色的可靠性、接近度以及耗时性对于本文方法而言, 并没有太多的对比和参考意义.

综合来看, Ours_Gradient、Ours_Importance 以及 CEM 的实验效果最好, 三者求出的反事实比较优质, 另外 3 种方法求解出的反事实都存在着较大的问题. 而 Ours_Gradient 和 Ours_Importance 分别代表着基于梯度排序的方法和基于特征重要性排序的方法, 这从实践的角度说明了本文方法的优异性.

4.4 反事实优化效果

在对反事实进行了对比评估之后, 需要将其应用到数据库中测试其优化效果.

我们会将原数据配置到数据库中, 然后重复进行 3 次负载测试并记录数据库的平均性能指标; 然后再将反事实数据配置到数据库中, 然后重复进行 3 次负载测试并记录数据库的平均性能指标; 最后通过比较两个指标进行反事实优化效果评估.

我们分别针对 Throughput、RunTime、ReadLatency、UpdateLatency 以及 InsertLatency 这 5 类优化指标, 使用能求解出有效反事实的数据进行数据库性能测试. 由于该过程耗时比较长, 所以这里相应地减少了测试数据的数目, 决定针对每一个指标使用 5 条数据进行测试, 实验结果如下 (为了能更快地求解出反事实, 基于梯度排序的反事实解释方法的循环终止条件在实际实验中, 设置为损失函数到达极小值或小于某定值).

在以下 5 组实验中, Index 是数据的下标; *loss*1 的具体含义在第 3.1 节中进行了说明, 这里不再进行赘述; 优化效果是指, 将反事实应用在数据库配置中, 性能相对于原配置提升的百分比, 以 Throughput 为例, 假设在原数据配置下的数据库 Throughput 为 100 ops, 在反事实配置下的数据库 Throughput 为 120 ops, 那么提升的百分比则为 20%.

● Throughput: 针对 Throughput 指标的实验结果如表 8 所示.

● RunTime: 针对 RunTime 指标的实验结果如表 9 所示.

● ReadLatency: 针对 ReadLatency 指标的实验结果如表 10 所示.

● UpdateLatency: 针对 UpdateLatency 指标的实验结果如表 11 所示.

● InsertLatency: 针对 InsertLatency 指标的实验结果如表 12 所示.

将实验结果中的 *loss*1 和优化效果作为 x 轴和 y 轴, 绘制成如图 14 所示散点图, 对图进行分析发现, *loss*1 和优化效果之间并不存在负相关关系, 即 *loss*1 越小, 优化效果越好; 这与实验预期结果相悖, 我们的预想是, *loss*1 越小, 即反事实的可靠性越高, 那么优化效果理应越好, 但是实验证明该想法是错误的. 经过进一步分析, 得出了以下解释: *loss*1 越小, 反事实的可靠性确实会更高, 但是该可靠性是指, 反事实能把坏的预测结果改变为好的可靠性, 而在第 2.3 节中进行标签预处理的时候, 只是简单地把性能指标小于等于中位数的数据置为负标签, 性能指标大于中位数的数据置为正标签, 所以反事实将预测结果从负标签改变到正标签, 改变的幅度是无法预知的, 这就导致了 *loss*1 和优化效果之间的无关性.

将实验结果中的优化效果, 绘制成如图 15 所示直方图, 对图进行分析发现, 大部分优化效果集中在 0–60% 之间, 部分优化效果在 60%–100% 之间, 甚至有少数优化效果超过了 100%. 这说明实验生成的反事实确实能够起到优化数据库性能的作用, 并且优化效果明显.

表 8 Throughput 优化实验结果

Index	基于梯度排序的方法		基于特征重要性排序的方法	
	*loss*1	优化效果	*loss*1	优化效果
7777	−0.400 122 076	0.126 329 787	−0.322 029 620	0.880 924 855
2307	−0.400 043 874	0.644 456 347	−0.495 669 305	0.173 376 623
12962	−0.128 905 177	0.811 584 800	−0.354 268 343	0.624 917 439
860	−0.401 102 900	0.556 818 181	−0.307 937 800	1.454 918 032
10515	−0.270 163 714	0.259 673 258	−0.378 724 306	0.254 693 877

表 9 RunTime 优化实验结果

Index	基于梯度排序的方法		基于特征重要性排序的方法	
	*loss*1	优化效果	*loss*1	优化效果
10666	−0.406 378 186	0.221 881 838	−0.359 302 401	0.874 733 096
10247	−0.414 033 949	0.295 483 870	−0.463 038 802	0.296 796 319
781	−0.400 023 698	0.237 314 279	−0.320 414 781	0.421 840 159
2192	−0.303 177 505	0.473 002 159	−0.338 426 828	0.138 220 694
7233	−0.315 031 230	0.206 377 102	−0.335 196 971	0.319 809 069

表 10 ReadLatency 优化实验结果

Index	基于梯度排序的方法		基于特征重要性排序的方法	
	*loss*1	优化效果	*loss*1	优化效果
7468	−0.402 473 300	0.194 160 583	−0.371 997 505	0.156 186 612
8173	−0.340 446 472	1.331 746 090	−0.607 533 931	0.290 184 049
5134	−0.401 119 828	0.925 777 331	−0.304 987 788	0.157 190 635
9054	−0.405 579 411	0.828 981 233	−0.355 726 003	0.253 521 126
5986	−0.400 072 157	0.132 530 120	−0.337 020 218	0.364 116 094

表 11 UpdateLatency 优化实验结果

Index	基于梯度排序的方法		基于特征重要性排序的方法	
	*loss*1	优化效果	*loss*1	优化效果
7403	−0.400 559 276	0.629 058 839	−0.319 986 134	0.551 268 386
2586	−0.281 378 954	0.485 623 302	−0.305 473 685	0.618 708 477
4818	−0.400 020 301	0.121 991 701	−0.371 130 734	0.666 066 232
1792	−0.401 402 384	0.084 411 384	−0.309 923 380	0.327 360 355
6049	−0.251 310 497	0.145 746 296	−0.390 573 203	0.156 270 847

表 12 InsertLatency 优化实验结果

Index	基于梯度排序的方法		基于特征重要性排序的方法	
	*loss*1	优化效果	*loss*1	优化效果
6360	−0.402 474 552	0.396 219 060	−0.436 734 437	0.505 225 893
7108	−0.400 243 461	0.421 673 606	−0.573 916 137	0.282 921 543
1839	−0.400 008 916	0.380 540 349	−0.512 798 309	0.541 462 590
5032	−0.402 076 750	0.133 856 722	−0.376 900 732	0.152 998 622
983	−0.401 136 219	0.604 202 988	−0.418 213 367	0.710 371 747

综合以上实验结果及分析, 由基于梯度排序的反事实解释方法和基于特征重要性排序的反事实解释方法求解出的反事实, 优化数据库性能的效果显著, 与实验预期相符.

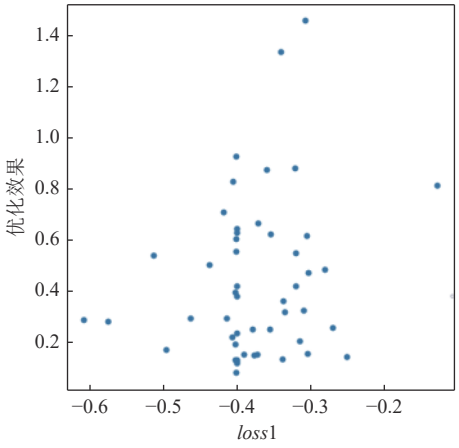

图 14 优化效果-*loss*1 散点图

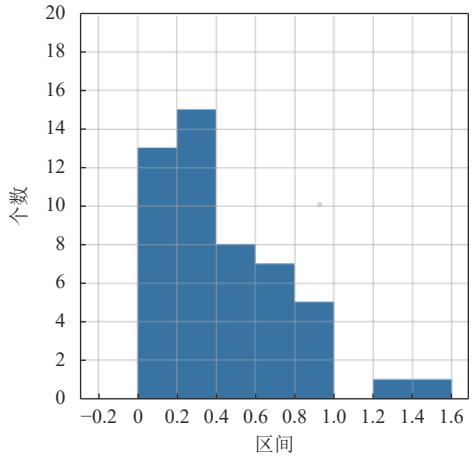

图 15 个数-区间直方图

5 总 结

本文针对数据库配置优化问题, 首先利用 YCSB 性能测试工具以及 MySQL 数据库, 采集并生成了 databaseConfiguration 数据集; 然后分别进行了 MLP 和 ResNet18 的搭建、训练和测试, 并在对两个模型从准确率、训练和测试的速度以及训练的稳定性 3 个方面进行了对比分析之后, 选择了 ResNet18 作为后续使用的预测模型; 接着基于预测模型研究和设计了两种反事实解释方法, 分别是基于梯度排序的反事实解释方法和基于特征重要性排序的反事实解释方法; 最后通过实验证明, 这两种方法能够利用深度学习模型来提取历史调参数据中的抽象特征, 求解相应数据的反事实, 进而找到在当前负载条件下更好的数据库参数组合, 以优化数据性能. 我们未来的工作包括进一步优化基于特征重要性排序的反事实解释方法的时间复杂度以及尝试将研究问题从二分类问题转化为多分类问题进行进一步研究.

References:

[1] Chen L. Automatic database tuning research based on machine learning. Software Guide, 2021, 20(11): 148–151 (in Chinese with English abstract). [doi: 10.11907/rjdk.211707]

[2] Zeng CY, Yan K, Wang ZF, Yu Y, Ji CM. Survey of interpretability research on deep learning models. Computer Engineering and Applications, 2021, 57(8): 1–9 (in Chinese with English abstract). [doi: 10.3778/j.issn.1002-8331.2012-0357]

[3] Lei X, Luo XL. Review on interpretability of deep learning. Journal of Computer Applications, 2022, 42(11): 3588–3602 (in Chinese with English abstract). [doi: 10.11772/j.issn.1001-9081.2021122118]

[4] Li GL, Zhou XH, Sun J, Yu X, Yuan HT, Liu JB, Han Y. A survey of machine learning based database techniques. Chinese Journal of Computers, 2020, 43(11): 2019–2049 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2020.02019]

[5] Wei Z, Ding ZH, Hu JL. Self-tuning performance of database systems based on fuzzy rules. In: Proc. of the 11th Int'l Conf. on Fuzzy Systems and Knowledge Discovery. Xiamen: IEEE, 2014. 194–198. [doi: 10.1109/FSKD.2014.6980831]

[6] Zhu YQ, Liu JX, Guo MY, Bao YG, Ma WL, Liu ZY, Song KP, Yang YC. BestConfig: Tapping the performance potential of systems via automatic configuration tuning. In: Proc. of the 2017 Symp. on Cloud Computing. Santa Clara: ACM, 2017. 338–350. [doi: 10.1145/3127479.3128605]

[7] Zhang J, Liu Y, Zhou K, Li GL, Xiao ZL, Cheng B, Xing JS, Wang YT, Cheng TH, Liu L, Ran MW, Li ZK. An end-to-end automatic cloud database tuning system using deep reinforcement learning. In: Proc. of the 2019 Int'l Conf. on Management of Data. Amsterdam: ACM, 2019. 415–432. [doi: 10.1145/3299869.3300085]

[8] Li GL, Zhou XH, Li SF, Gao B. QTune: A query-aware database tuning system with deep reinforcement learning. Proc. of the VLDB Endowment, 2019, 12(12): 2118–2130. [doi: 10.14778/3352063.3352129]

[9] Le T, Wang SH, Lee D. GRACE: Generating concise and informative contrastive sample to explain neural network model's prediction. In: Proc. of the 26th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining. ACM, 2020. 238–248. [doi: 10.1145/3394486.

3403066]
[10] White A, d'Avila Garcez A. Measurable counterfactual local explanations for any classifier. arXiv:1908.03020, 2019.
[11] Sharma S, Henderson J, Ghosh J. CERTIFAI: Counterfactual explanations for robustness, transparency, interpretability, and fairness of artificial intelligence models. arXiv:1905.07857, 2019.
[12] Guidotti R, Monreale A, Ruggieri S, Pedreschi D, Turini F, Giannotti F. Local rule-based explanations of black box decision systems. arXiv:1805.10820, 2018.
[13] Mothilal RK, Sharma A, Tan CH. Explaining machine learning classifiers through diverse counterfactual explanations. In: Proc. of the 2020 Conf. on Fairness, Accountability, and Transparency. Barcelona: ACM, 2020. 607–617. [doi: 10.1145/3351095.3372850]
[14] Yang F, Alva SS, Chen JH, Hu X. Model-based counterfactual synthesizer for interpretation. In: Proc. of the 27th ACM SIGKDD Conf. on Knowledge Discovery & Data Mining. Singapore: ACM, 2021. 1964–1974. [doi: 10.1145/3447548.3467333]
[15] Panchal G, Ganatra A, Kosta YP, Panchal D. Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers. Int'l Journal of Computer Theory and Engineering, 2011, 3(2): 332–337. [doi: 10.7763/IJCTE.2011.V3.328]
[16] Li ZW, Liu F, Yang WJ, Peng SH, Zhou J. A survey of convolutional neural networks: Analysis, applications, and prospects. IEEE Trans. on Neural Networks and Learning Systems, 2022, 33(12): 6999–7019. [doi: 10.1109/TNNLS.2021.3084827]
[17] Wang F, Tax DMJ. Survey on the attention based RNN model and its applications in computer vision. arXiv:1601.06823, 2016.
[18] He KM, Zhang XY, Ren SQ, Sun J. Deep residual learning for image recognition. In: Proc. of the 2016 IEEE Conf. on Computer Vision and Pattern Recognition. Las Vegas: IEEE, 2016. 770–778. [doi: 10.1109/CVPR.2016.90]
[19] Schaetti N. Character-based convolutional neural network and ResNet18 for twitter author profiling: Notebook for PAN at CLEF 2018. In: Proc. of the 2018 Working Notes of CLEF－Conf. and Labs of the Evaluation Forum. Avignon: CEUR-WS.org, 2018.
[20] Verma S, Boonsanong V, Hoang M, Hines KE, Dickerson JP, Shah C. Counterfactual explanations and algorithmic recourses for machine learning: A review. arXiv:2010.10596, 2020.
[21] Stepin I, Alonso JM, Catala A, Pereira-Fariña M. A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence. IEEE Access, 2021, 9: 11974–12001. [doi: 10.1109/ACCESS.2021.3051315]
[22] Ribeiro MT, Singh S, Guestrin C. "Why should I trust you?": Explaining the predictions of any classifier. In: Proc. of the 22nd ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. San Francisco: ACM, 2016. 1135–1144. [doi: 10.1145/2939672.2939778]
[23] Damodaran BD, Salim S, Vargese SM. Performance evaluation of MySQL and MongoDB databases. Int'l Journal on Cybernetics & Informatics, 2016, 5(2): 387–394. [doi: 10.5121/ijci.2016.5241]
[24] Pandey R. Performance benchmarking and comparison of cloud-based databases MongoDB (NoSQL) vs. Technical Report. MySQL (Relational) using YCSB. Berlin: National College of Ireland, 2020. 8 [doi: 10.13140/RG.2.2.10789.32484]
[25] van Aken D, Pavlo A, Gordon GJ, Zhang BH. Automatic database management system tuning through large-scale machine learning. In: Proc. of the 2017 ACM Int'l Conf. on Management of Data. Chicago: ACM, 2017. 1009–1024. [doi: 10.1145/3035918.3064029]
[26] Hu ZB, Tereykovskiy IA, Tereykovska LO, Pogorelov VV. Determination of structural parameters of multilayer perceptron designed to estimate parameters of technical systems. Int'l Journal of Intelligent Systems and Applications, 2017, 9(10): 57–62. [doi: 10.5815/ijisa.2017.10.07]
[27] Dhurandhar A, Chen PY, Luss R, Tu CC, Ting PS. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In: Proc. of the 32nd Int'l Conf. on Neural Information Processing Systems. Montréal: Curran Associates Inc., 2018. 590–601.
[28] van Looveren A, Klaise J. Interpretable counterfactual explanations guided by prototypes. In: Proc. of the 2021 Joint European Conf. on Machine Learning and Knowledge Discovery in Databases. Bilbao: Springer, 2021. 650–665. [doi: 10.1007/978-3-030-86520-7_40]

附中文参考文献:

[1] 陈镭. 基于机器学习的数据库系统自动调参研究. 软件导刊, 2021, 20(11): 148–151. [doi: 10.11907/rjdk.211707]
[2] 曾春艳, 严康, 王志锋, 余琰, 纪纯妹. 深度学习模型可解释性研究综述. 计算机工程与应用, 2021, 57(8): 1–9. [doi: 10.3778/j.issn.1002-8331.2012-0357]
[3] 雷霞, 罗雄麟. 深度学习可解释性研究综述. 计算机应用, 2022, 42(11): 3588–3602. [doi: 10.11772/j.issn.1001-9081.2021122118]
[4] 李国良, 周煊赫, 孙佶, 余翔, 袁海涛, 刘佳斌, 韩越. 基于机器学习的数据库技术综述. 计算机学报, 2020, 43(11): 2019–2049. [doi: 10.11897/SP.J.1016.2020.02019]

朱霄(2000－), 男, 硕士生, 主要研究领域为机器学习可解释性, 反事实解释, 数据库智能化.

张岩(1965－), 男, 副教授, CCF 高级会员, 主要研究领域为数据库, 信息可用性管理, 算法理论.

邵心玥(1996－), 女, 博士生, 主要研究领域为黑盒算法可解释性, 反事实解释.

王宏志(1978－), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为数据库管理系统, 大数据分析与治理.