软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(3):799−810 [doi: 10.13328/j.cnki.jos.005439]

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

基于循环神经网络的数据库查询开销预测*

毕里缘[1], 伍 赛[1,2], 陈 刚[1,2], 寿黎但[1,3], 陈 珂[1,2], 胡天磊[1,2]

[1](浙江大学 计算机科学与技术学院,浙江 杭州 310027)

[2](浙江省大数据智能计算重点实验室(浙江大学),浙江 杭州 310027)

[3](浙江大学 计算机辅助设计与图形学国家重点实验室,浙江 杭州 310027)

通讯作者: 伍赛, E-mail: wusai@zju.edu.cn

摘 要: 在数据库负载管理、性能调优过程中,开销预测模型是提高其效率的关键技术.首先,由于数据库系统的复杂性和计算机资源的竞争,很难精确地估计不同操作的开销;其次,现有的研究大多没有真正预测查询的执行时间,而是预测了类似查询优化器中开销模型生成的开销;由于查询计划结构的复杂性,现有研究更多地使用了笼统的查询信息,而很少利用查询计划中操作层面的信息,并依据这些信息来获得开销模型.为了减少负载管理的复杂性,提出了基于循环神经网络的精细模型来预测查询开销,以查询计划中的操作行为及其实际运行时间作为特征提取的来源.特别地,考虑到查询计划结构的复杂性,采用一种特殊的循环神经网络——长短期记忆(long-short term memory,简称 LSTM).给一个特定的查询计划,在该计划实际执行之前,模型就能产生其预测的执行时间区间.这会比现有数据库的查询优化器产生的开销预估结果(任意单位)更具有参考性,也优于需要在执行开始之后才能预测的查询进度指示器.所提方法预测查询执行时间,可以解决数据库负载管理中的关键问题.通过实验验证,模型的正确率高于 71%,在一定程度上证明了方法的可行性.

关键词: 数据库负载管理;查询开销预测;查询计划;循环神经网络;长短期记忆

中图法分类号: TP311

中文引用格式: 毕里缘,伍赛,陈刚,寿黎但,陈珂,胡天磊.基于循环神经网络的数据库查询开销预测.软件学报,2018,29(3):799−810. http://www.jos.org.cn/1000-9825/5439.htm

英文引用格式: Bi LY, Wu S, Chen G, Shou LD, Chen K, Hu TL. Database query cost prediction using recurrent neural network. Ruan Jian Xue Bao/Journal of Software, 2018,29(3):799−810 (in Chinese). http://www.jos.org.cn/1000-9825/5439.htm

Database Query Cost Prediction Using Recurrent Neural Network

BI Li-Yuan[1], WU Sai[1,2], CHEN Gang[1,2], SHOU Li-Dan[1,3], CHEN Ke[1,2], HU Tian-Lei[1,2]

[1](College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China)

[2](Key Laboratory of Big Data Intelligent Computing of Zhejiang Province (Zhejiang University), Hangzhou 310027, China)

[3](State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou 310027, China)

Abstract: Query cost models are the key parts of database workload management and performance tuning. Firstly, it is difficult, even impossible, to precisely estimate the costs of different relational operators due to the complexity of database systems and competition of computer resources. Secondly, most existing research work uses general query information without taking advantage of actual operators

* 基金项目: 国家重点基础研究发展计划(973)(2015CB352400); 国家自然科学基金(61661146001, 61472348, 61672455); 浙江省自然科学基金 (LY18F020005)

Foundation item: National Program on Key Basic Research Project (973) (2015CB352400); National Natural Science Foundation of China (61661146001, 61472348, 61672455); Natural Science Foundation of Zhejiang Province of China (LY18F020005)

本文由基于图结构的大数据分析与管理技术专刊特约编辑林学民教授、杜小勇教授、李翠平教授推荐.

收稿时间: 2017-07-31; 修改时间: 2017-09-05; 采用时间: 2017-11-07; jos 在线出版时间: 2017-12-05

CNKI 网络优先出版: 2017-12-06 15:23:06, http://kns.cnki.net/kcms/detail/11.2560.TP.20171206.1522.004.html

because of the complexity of query plans. Thirdly, most previous research work does not address the problem of predicting actual execution time of a query but rather predicts the query performance by the cost the like query optimizers generate. To reduce the complexity of workload management, his paper proposes an elaborate cost prediction model based on recurrent neural network through learning from operator behavior and detailed runtime information. In particular, the model uses a special kind of recurrent neural network, called long-short term memory (LSTM). Given an ad-hoc query, the model is able to predict its running time before it starts to run. It is more meaningful than the state-of-the-art query optimizers of existing database systems which only estimate costs in arbitrary units. It is also better than query progress indicators which cannot predict cost before the query runs. This research provides a novel approach to solve the key problem in database workload management. Verified by the experiments, the accuracy of the model is over 71% which shows the method is feasible to some degree.

Key words: database workload management; query cost prediction; query plan; recurrent neural network; long-short term memory

随着数据库中数据量的与日俱增和查询的日益复杂,数据库负载管理、查询调度面临极大的挑战.负载管理要解决的核心问题就是查询执行时间的预测.对于用户提交的查询任务,管理系统要判断是否执行;若执行就要确定任务执行顺序;如果一个任务执行很久没有结束,需要判断是否继续执行或者强制结束.无法预计时间的长时查询,是计算机资源耗尽的根源所在.如果在查询执行之前能够确定其运行时间,就可以取消执行无法在期望时间内完成的查询,或者在不影响其他查询的情况下执行.但是,由于数据库系统的复杂性和计算机资源的竞争,很难精确地估计不同数据库操作的开销.因此,查询开销预测成为一个重要的研究问题.

除此以外,预测查询执行时间对于数据库管理领域的其他方面也有重要意义[1],如权限控制[2,3]、进度监控[4]和系统规模定制[5].

针对查询开销的预测问题,研究人员提出了许多面向关系型数据库的查询开销预测方案.其中,有些研究没有预测出真实的运行时间,而是估计了查询完成的百分比或者输出了一个任意单位的值来代表查询开销,类似于查询优化器的开销预测;有些研究需要运行时性能统计,而这又需要额外的开销去产生统计数据[6].

虽然上述研究方案在一定程度上解决了查询开销预测的问题,但是都有各自的局限性:第一,开销预估的结果是任意单位,很难映射到时间单位,无法支持负载管理中的一些决策;第二,预测需要查询执行中的信息,无法在查询执行前就给出预测.

为了解决以上问题,本文提出了一种基于循环神经网络的查询开销预测方法,该方法不仅能够预测查询计划的执行时间,而且在查询执行前就可以得到预测结果.模型主要分为编码和预测两个模块:编码模块能够接受一个 JSON 格式的查询计划并产生与它等价的特征向量;预测模块用于预测该查询计划的开销.

本文的主要贡献如下:

(1) 设计了一种编码方法,将树形结构的查询计划转换成特征向量,并尽可能保留其影响开销的信息;

(2) 提出了一种基于循环神经网络的查询开销预测模型.给定一个查询计划,在计划实际执行前,模型就能够产生该查询计划实际运行时间的预测;

(3) 通过实验验证,不管是短时间运行的查询还是长时间运行的查询,模型的预测结果都较为准确.模型的正确率高于 71%,在一定程度上证明了本文模型的可行性.

本文第 1 节介绍相关的研究成果.第 2 节描述模型要解决的问题.第 3 节介绍模型的设计和实现.第 4 节通过分析实验的结果,验证模型的可行性.第 5 节总结全文并提出未来工作的展望.

1 相关工作

1.1 查询开销预测

数据库查询优化器根据开销模型产生的开销预估比较不同计划的好坏.使用 PostgreSQL 提供的 explain 操作,可以输出查询计划的开销预估.但是这个开销模型并不擅长预测开销具体的执行时间,因为它只反映了优化器所关心的部分,而忽略了一些影响开销的重要因素.例如,开销模型没有考虑结果行传递给客户端所花费的时间,优化器忽略了它的原因是对于不同的计划,它的值不会产生变化.但是对于负载管理,这个重要因素不能被

忽略.其次,这个开销模型依赖于直方图或者应用采样技术.由于空间有限,直方图间隔尺寸不能太细,因此开销预测就会不准确.最后,大多数优化器使用的单位是任意单位,无法轻易地映射到时间单位[6].

与查询优化器不同的是,查询进度指示器能够给出时间单位的预测,而且它使用了基于操作行为和运行信息的一个更为精细的模型.本文中的模型也是基于操作层面的信息建立的.但是查询进度指示器不能在查询执行前进行开销预测,它预测的是运行中的查询完成进度[6].

文献[6]提出使用机器学习的方法预测查询开销,与本文最大的区别在于其处理特征向量的方法丢失了查询计划的结构信息.文献[7]提出一种能够对数据库系统工作负载类型分类的模型,试图去解决性能调优问题,但并没有直接给出查询开销的预测.文献[8]使用统计机器学习技术建立了查询计划层面信息和操作层面信息的模型以及混合两者的模型来预测查询性能,但没有直接预测查询执行时间.同样使用统计机器学习技术来预测开销的还有文献[9,10].

文献[11]首次提出了应用循环神经网络(recurrent neural network,简称 RNN)到数据库查询优化领域的思想概括,而本文运用 LSTM 解决了具体问题并实现验证了方法的可行性.现在,越来越多的研究尝试使用深度学习方法来解决数据库领域的问题.例如,文献[12]运用 LSTM 预测一段时间内将要执行查询的数量,解决了负载管理领域的另一个问题.

1.2 神经网络

前馈神经网络(feedforward neural network,简称 FNN)是最基础的神经网络,网络拓扑结构上不存在环或者回路.图 1 展示了 FNN 的结构,包括一个输入层、一个或多个隐藏层和一个输出层.输入层接收一个向量:$x=(a_0, a_1,\ldots,a_{n-1})$.即,输入层有 n 个神经元并且每一个神经元都有特定的值 a_i.u_j 是第 1 个隐藏层的第 j 个神经元.u_j 可以按照公式(1)计算:

$$u_j = f\left(\sum_{i=0}^{n-1} w_{ij}a_i + b_j\right) \tag{1}$$

f 是激活函数,例如典型的激活函数有 ReLU 和 tanh.w_{ij} 是从输入层的第 i 个神经元到隐藏层的第 j 个神经元之间的权重,而 b_j 是偏移量.

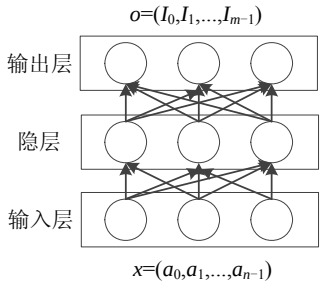

Fig.1 FNN model structure

图 1 FNN 模型结构

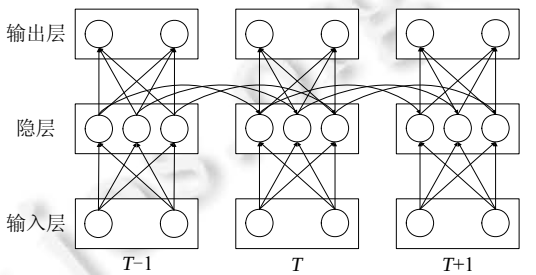

Fig.2 The expanding RNN model structure

图 2 展开的 RNN 模型结构

通常,为了简化表达,可以使用向量 U、向量 B 和矩阵 W 来重写公式(1),见公式(2):

$$U=f(Wx+B) \tag{2}$$

每一个隐藏层都会调用上述的计算过程,该计算过程被称为正向传播.最终,最后一个隐藏层计算得到输出向量 $o=(l_0,l_1,\ldots,l_{m-1})$.输出向量会与真实的结果进行比较,并计算出预测误差,反向传播至隐藏层来调整层与层之间的权重矩阵.反向传播可以使用随机梯度下降方法(stochastic gradient descent,简称 SGD).

FNN 很难对随时间状态变化的过程建模,例如股票价格预测和语音识别.其原因是:FNN 主要利用窗处理来做时序预测,但模型受限于窗的大小(帧数).

而 RNN 比 FNN 多了时间步(time step)的概念,当前时刻的预测同时取决于当前输入和上一时刻的状态.因

此,RNN 特别适用于处理声音、时间序列数据或自然语言等序列数据.图 2 是 RNN 随时间展开后的结构.

通过图 1 和图 2 中 FNN 和 RNN 模型结构的比较不难发现,RNN 本质上是 FNN 在时间维度上的扩展.将图 2 简化一下,RNN 的模型结构如图 3 所示.

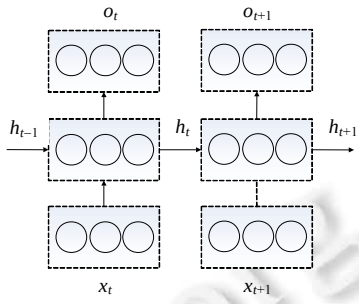

Fig.3 RNN model structure

图 3 RNN 模型结构

在时间状态 t,神经模块收到输入 $x_t=(a_0,a_1,\ldots,a_{n-1})$, 隐藏层按照公式(3)计算:

$$h_t = f(Wx_t + \bar{W}h_{t-1} + B) \tag{3}$$

f,W 和 B 都与之前定义的一致.$\bar{W}$ 是前一个状态和现在状态之间的权重矩阵.因此,在每个时间状态都能获得一个输出向量 o_t.FNN 和 RNN 之间主要的区别在于,RNN 使用了代表之前状态的值 h_{t-1} 来计算新的状态.公式(3)定义了一个简单的 RNN,但由于梯度消失和梯度爆炸,普通 RNN 难以传递相隔较远的信息.因此,考虑到查询计划的结构,在本文中采用了一种特殊的 RNN——长短期记忆(long-short term memory,简称 LSTM)[13].关于 RNN 的更多细节,见文献[13,14].

LSTM 有 3 个门:输入门、遗忘门和输出门.本文采用了 LSTM 的一种变种——peephole connection[14].这个变种在 LSTM 的 3 个门的计算中加入了细胞状态.第 3.2 节将详细阐述这种特殊 LSTM 的结构.

2 问题与方法

图 4 展现了传统查询优化器的基本结构[15],包括两个最重要的模块:查询生成器和搜索算法.查询生成器应用转换规则产生可能的逻辑和物理计划,而搜索算法依据开销模型指导查询生成器产生高质量的候选计划.最终,优化器产生了较好的查询计划并用于执行.计划的实际运行统计会反馈给查询优化器并对其进行调节.

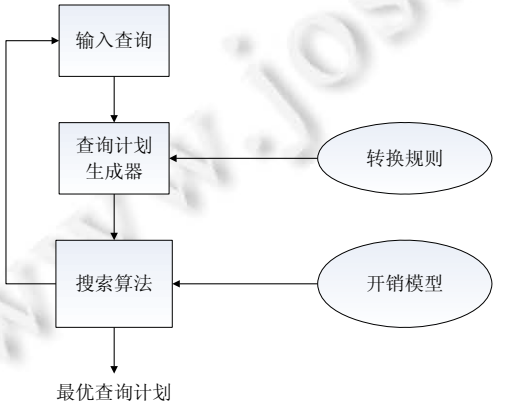

Fig.4 Process of query optimization

图 4 查询优化过程

开销模型预测的准确性直接影响了查询优化器的性能.为了估计不同数据库操作的开销,现有的方法是建立直方图[16,17]或者是应用采样技术[18].因此,开销模型可以估计需要执行多少顺序或随机 I/O.估计的精确程度依赖于数据分布,但是由于空间限制,不能建立过多直方图或者保存过多样本.因此,开销估计常常是不准确的,导致优化器产生了性能极差的查询计划.另一方面,查询计划的搜索空间可能会过大,对于 n 个表连接的查询来说,有$(2n-2)!/(n-1)!$种连接计划.查询优化器不可能遍历所有的计划.

数据库中的负载管理离不开准确的开销模型,而查询优化器中开销模型无法满足需求,第 1.1 节详细阐述了原因.因此,本文提出了基于神经网络的开销模型,功能类似于查询优化器的开销模型,用于负载管理场景,模型框架如图 5 所示:首先,从数据库历史查询记录中抽取出查询计划和运行时间构成原始数据,按照查询计划的运行时间长短将原始数据分类,使得数据集中每类包含的查询计划的数量相等;其次,将查询计划编码成操作向量,获得训练集.最后训练并得到模型.针对待测的查询计划,编码得到其对应的操作向量,再输入神经网络模型中,输出与操作序列对应的运行时间序列,完成数据库查询时间的预测.

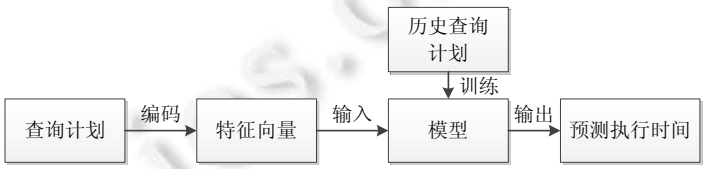

Fig.5 General idea of the model

图 5 模型框架

一个查询语句对应多个查询计划.查询计划的结构是一棵多叉树,直观上看并不是序列数据.但显然:查询计划树的子树开销会影响父节点的开销;同一个节点的子树之间相互独立,互不影响.因此,本文采用后序编码来保留这种结构信息,将查询计划树转换成序列.序列前面的数据会影响后面的数据,抽象来说,就是一个时间序列.但序列中有些数据前后独立,即:同一个节点的子树在序列中虽然有先后关系,但彼此之间却是独立的.这与机器翻译问题非常相似,因此,本文模型遵循了神经网络机器翻译的思路[19−21].

后序编码后的查询计划作为神经网络的输入.网络逐个地学习查询计划中每个操作的开销,那么查询计划的第 1 个操作开销将会帮助网络决定第 2 个操作的开销预测结果.最后,网络输出该查询计划的开销序列.

3 模型建立

3.1 特征提取

应用深度学习算法需要特征向量与预测标签.预测标签,即模型的输出,有两种设计方案.

- 第 1 个方案是试图解决回归问题,也就是直接预测时间.由于不同查询计划可能导致大量不同的开销,不利于模型的训练.本文的相关实验验证了以上的观点.再次分析问题的应用场景,其实预测时间本质目的是区分短时间查询和长时间查询,即并不关心非常精确的时间,重要的是确定查询开销处于哪个数量级;
- 第 2 个方案是解决分类问题,将时间范围划分成 c 个区间,预测开销落在那个区间.实验结果表明,第 2 个设计方案比第 1 种设计方案更合适.

注意:区间的划分并不是数值均匀的,而是保持每个区间内的数据量均匀.假设有 m 条数据和 c 个区间,那么每个区间内的数据量应该是 m/c 条.可以将所有数据排序,根据每个区间内的数据量来划分区间范围.若数据量较大,可采用抽样方法来确定区间范围.

相比预测标签,定义特征向量就比较复杂.图 6 是下面查询语句的一个查询计划树.

SELECT $T_1.k$

FROM T_1,T_2,T_3

WHERE $T_1.k=T_2.fk$ AND $T_1.fk=T_3.k$ AND $T_1.k<100$

GROUP BY $T_1.k$

其中,T_1,T_2,T_3 分别表示了数据库中的源表,即查询计划的各个操作可能用到的源表;$T_1.k$ 表示 T_1 表的主键,同理,$T_3.k$ 表示 T_3 表的主键;$T_1.fk$ 表示了 T_1 表的外键,同理,$T_2.fk$ 表示了 T_2 表的外键.

查询优化器产生一个查询计划只要毫秒到秒级别的时间,因此,这些信息不难获取.查询计划本质上是一棵以各种类型的操作为节点的多叉树,每个节点上都有操作的相关信息.例如,图 6 中左侧 Hash Join 节点代表的是连接类型下的一种操作,其连接的条件是 $T_1.k=T_2.fk$.

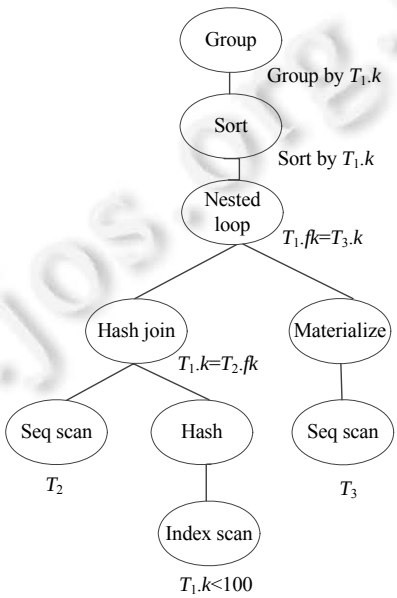

Fig.6 A typical query plan

图 6 查询计划树

在将查询计划编码成 LSTM 能接收的格式时,为了保留查询计划树的结构信息,本文采用后序编码,将其编码成一个操作序列 $S_{op}=\{op_0,op_1,\ldots,op_{m-1}\}$,$op_i$ 是操作序列 S_{op} 中第 i 个操作,m 表示查询计划中操作的个数.图 6 中查询计划树的后续遍历可以表示成:

$$S_{op}=\{T_2,\sigma T_1.k<100,Hash,\overline{T}=T_1\triangleright\triangleleft T_2,T_3,Materialize,\overline{T}\triangleright\triangleleft T_3,Sort,Group\}.$$

σ表示选择操作,$\triangleright\triangleleft$表示表的连接操作,$\overline{T}$ 表示 T_1,T_2 连接后的结果.使用多叉树后序遍历,一棵计划树可以被转换成独一无二的 S_{op}.反之亦然,因为在每一个节点中都保存了源表信息,即树中的叶子节点,所以 S_{op} 也能转换成一棵独一无二的计划树.

对于一个查询计划中的各个操作,经过后序遍历生成操作序列,遍历生成操作序列时,对于查询计划中的各个操作提取关键特征,将每个操作转换成向量 $v.v$ 包含 5 个部分:

(1) n_0 代表操作的类型,例如 Hash Join,Nested Loop 等.PostgreSQL 中共有 34 种操作类型,表 1 给出了操作清单.因此,n_0 是一个 34 位的向量,该操作类型对应的位设置为 1,其他位设置为 0;

(2) n_1 代表操作在数据库中对应的源表.假设数据库有 k 个表,那 n_1 就有 k 位.计划树的叶子节点带有源表信息.例如,图 6 中左边叶子节点的源表是 T_2,因此,该叶子节点操作的 n_1 中 T_2 对应的位设置为 1,其他位设置为 0.子节点的源表信息会传递给父节点.例如,图 6 中 Hash Join 操作的源表是 T_1 和 T_2,分别来自它的两棵子树;

(3) n_2 代表操作在数据库中对应源表中涉及的列.假设数据库中所有表共 m 列,那 n_2 就有 m 位.例如,图 6

中 Hash Join 涉及表 T_1 中主键和表 T_2 中的外键,那么这两列对应的位就会被设置为 1,其余的位设置为 0;

(4) n_3 代表操作对应结果行的平均宽度,即每行所占字节数.假设操作的输出仅包含一列且该列的类型是 integer,那么结果行的平均宽度是 4 字节.将宽度的范围划分成 c 个区间,宽度落在哪个区间内,该区间对应的位就设置为 1,其余位设置为 0;

(5) n_4 代表操作对数据的选择率,即结果数据量占总量的百分比.这部分考虑了数据分布对于开销的影响.由于 TPC-H 的数据集分布是均匀的,因此本文实验部分不包含 n_4,但实际数据库中的数据分布很有可能是不均匀的.可以使用等宽直方图或等高直方图来计算选择率.等高直方图更加准确,因为频度较高的属性值,所处的桶范围就小,因此更为近似;相反,频度较低的属性值,等高直方图的近似相对不准确,但对于估算结果来说,频度高的属性值更重要.多列条件筛选时,选择率可以是多列选择率的乘积,也可以收集多列统计信息.前者的实现方式更为简单,后者估算结果更为准确,但需要维护统计信息.

Table 1 Operation list

表 1 操作清单

编号	类型	编号	类型	编号	类型
1	Result	13	Seq scan	25	Materialize
2	Insert	14	Index scan	26	Sort
3	Update	15	Index only scan	27	Group
4	Delete	16	Bitmap index scan	28	Aggregate
5	Append	17	Bitmap HeapScan	29	WindowAgg
6	Merge append	18	Tid scan	30	Unique
7	Recursive union	19	Subquery scan	31	SetOp
8	BitmapAnd	20	Function scan	32	LockRows
9	BitmapOr	21	Values scan	33	Limit
10	Nested loop	22	Cte scan	34	Hash
11	Merge join	23	Worktable Scan		
12	Hash join	24	Foreign scan		

向量 v 的前 3 个部分描述了查询的结构,后 2 个部分跟踪了查询过程中包含的数据规模.因此,当数据库系统中数据量动态变化时,模型的输入的一部分也会产生变化,因此模型的输出也会受到影响.同时,用户在此期间产生的历史查询会作为新的训练集继续训练模型,因而模型会随着数据库数据量的变化而变化.现在用一个向量来表示一个特定的操作,即 $S_v=\{v_0,v_1,\ldots,v_{m-1}\}$ 表示一个特定的计划,v_i 是后序编码中第 i 个操作.在这节的其余部分里,将展示如何使用 S_v 来训练 LSTM 模型.

3.2 LSTM模型

LSTM 有 3 个门:输入门、遗忘门和输出门,它可以自适应地记住过去的状态来加强它的学习过程.本文应用了 LSTM 的一个变种——peephole connections[14],如图 7 所示.

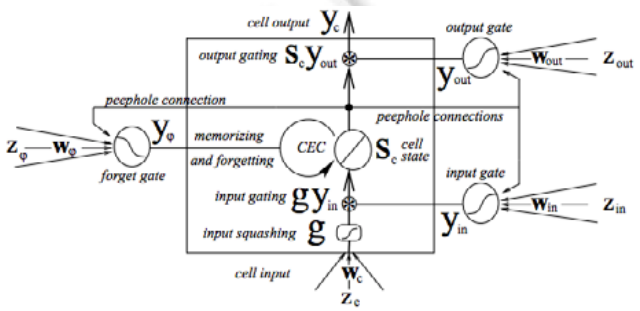

Fig.7 LSTM memory block with peephole connections from the CEC to the gates[14]

图 7 使用 peephole connections 的 LSTM 结构[14]

peephole 连接起了门与 CEC(constant error carousel),可以理解为 3 个门的输入信息增加了 Cell 的状态.这

个变种可以在没有任何短训练样本的帮助下学习间隔 50 个或 49 个时间步的序列之间的细微差别,比普通 LSTM 学习长序列的能力要强[12].而查询计划形成的序列也很长,因而选用了这种网络.

peephole connections 有多个实现的版本,只在细节上有所差异.本文所采用的 peephole connections 是 DL4J 中的版本,见公式(4)~公式(8).

定义 I 是输入的个数,K 是输出的个数,H 是隐藏层中细胞的个数,C 是块(block)中细胞的个数.w_{ij} 是 i 单元~j 单元之间的权重.在 t 时间, j 单元的值经过激励函数输出的值定义为 y_j^t .b_j 表示 j 单元的偏移量.下标 t、ϕ 和ω分别表示输入门、遗忘门和输出门.下标 c 表示 C 中的记忆细胞(memory cell). s_c^t 表示 t 时间细胞 c 的状态.下标 h 代表隐藏层中其他块的细胞输出.f 是门的激励函数,g_{in} 代表细胞输入的激励函数,g_{out} 代表细胞输出的激励函数.

公式(4)、公式(5)、公式(7)表示了 3 个门前向传播的计算方式,公式(6)、公式(8)表示 cell 的状态和 cell 的输出.关于反向传播的公式见文献[23].

$$y_\iota^t = f\left(\sum_{i=1}^{I} w_{i\iota} x_i^t + \sum_{h=1}^{H} w_{h\iota} y_h^{t-1} + \sum_{c=1}^{C} w_{c\iota} s_c^{t-1} + b_\iota\right) \tag{4}$$

$$y_\phi^t = f\left(\sum_{i=1}^{I} w_{i\phi} x_i^t + \sum_{h=1}^{H} w_{h\phi} y_h^{t-1} + \sum_{c=1}^{C} w_{c\phi} s_c^{t-1} + b_\phi\right) \tag{5}$$

$$s_c^t = y_\phi^t s_c^{t-1} + y_\iota^t g\left(\sum_{i=1}^{I} w_{ic} x_i^t + \sum_{h=1}^{H} w_{hc} y_h^{t-1} + b_c\right) \tag{6}$$

$$y_\omega^t = f\left(\sum_{i=1}^{I} w_{i\omega} x_i^t + \sum_{h=1}^{H} w_{h\omega} y_h^{t-1} + \sum_{c=1}^{C} w_{c\omega} s_c^t + b_\omega\right) \tag{7}$$

$$y_c^t = y_\omega^t g_{out}(s_c^t) \tag{8}$$

由公式(4)~公式(8)可推导出 LSTM 隐藏层的参数个数为

$$Num_{parameters} = [(I + H + C + 1) \times 3 + I + H + 1)] \times H \tag{9}$$

使用第 3.1 节介绍的 S_v 作为模型的输入.模型试图去预测查询计划每一步的运行开销,再与真实的结果比较计算出误差来调节网络.

S_t 代表子计划在时间 t 已经被执行了.对于计划 S_v=$\{v_0,v_1,\ldots,v_{m-1}\}$来说,$S_t$=$\{v_0,v_1,\ldots,v_{t-1}\}$($t$<$m$)是它的一个子计划.将时间范围划分为 n 个区间,记为$\{c_0,c_1,\ldots,c_{n-1}\}$.

定义 S_t 的开销为$\delta(S_t)$,$\delta(S_t)$落入某区间 c_i 的概率表示为 $P_{S_t,c_i} = P(\delta(S_t) \in c_i)$,最终的预测结果见公式(10):

$$C(S_t) = MAX(P_{S_t,c_i})(i = 0,1,...,n-1) \tag{10}$$

例 1:对于图 6 中展示的查询计划,在预测了它的子计划 S_2=$\{T_2,\sigma T_1.k$<100,$Hash\}$后,有了扫描表 T_2 和选择 T_1 特定元组的开销预测.假设 S_2 的开销是$\delta(S_2)$.

接下来,模型要预测子计划 $S_3 = \{T_2,\sigma T_1.k < 100, Hash, \overline{T} = T_1 \triangleright\!\triangleleft T_2\}$ 的开销.假设过程 $\{\overline{T} = T_1 \triangleright\!\triangleleft T_2\}$ 产生了开销δ'.S_3 的开销实际上是$\delta(S_2)$+δ',模型通过训练可以掌握该规则.因为 LSTM 可以通过当前的输入和上一时刻的状态来预测当前的结果,所以子计划被作为上一时刻的状态来预测总查询计划的开销.

3.3 网络结构设计

网络设计要确定层数、每个隐藏层的节点数和激活函数、输出层的激活函数和代价函数,以及一些重要参数,如优化算法、初始学习率等.本文设计的网络结构如图 8 所示:第 1 层是输入层,中间两层是隐藏层,最后一层是输出层.隐藏层所用的激活函数是 *sigmoid*,输出层的激活函数是 *soft*max.两个隐藏层都是 100 节点.输入层的节点数取决于特征向量的长度,见表 3、表 4.

优化算法采用了随机梯度下降方法,初始学习率设为 0.05.

代价函数是交叉熵代价函数,见公式(11)(损失越小,表示预测效果越好.训练过程就是不断减小损失的过程):

$$J=-\sum_{k=1}^{n}\sum_{i=1}^{c}y_{ki}\log(\hat{y}_{ki}) \tag{11}$$

公式(11)中,样本数量为 n,类别数量是 c,y_{ki} 是样本 k 属于类别 i 的概率,$\hat{y}_{ki}$ 是模型预测样本 k 属于类别 i 的概率.

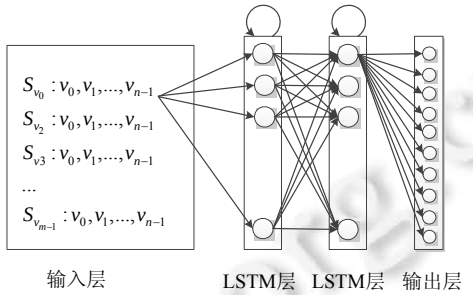

Fig.8 Network structure

图 8 网络结构

4 实 验

4.1 实验环境

为了评估模型的性能,本文使用 DL4J 实现了基于 LSTM 的开销预测模型.训练数据和测试数据来源于 TPC-H 决策支持基准[22],采用了 PostgreSQL 数据库.

本文做了两组实验:一组使用查询优化器产生的 cost 进行训练,一组收集查询真实的运行时间作为训练集.分别称为模拟数据和真实数据.

原型系统采用 Java 语言开发,系统环境为 Ubuntu 14.04.2 LTS,Linux 3.13.0-99-generic (x86_64),2x Intel(R) Xeon(R) CPU E5-2650 v2@2.60GHz,内存 96GB,硬盘 SEAGATE ST9300603SS SAS 10K,300GB.

4.2 训练集收集

大部分 DBMS(如 PostgreSQL 和 SQLServer)都支持 explain 命令,可以得到建议的执行计划和查询优化器估计的 I/O 开销.为了使用实际运行数据作为训练数据集,可以使用 explain analyze 命令去运行和收集计划中每一个操作真实的运行时间.

Table 2 The experimental data range

表 2 实验数据范围

实验组别	开销单位	数量		开销范围		
		训练集	测试集	平均	最小	最大
1	任意单位(arb. unit)	120 000	30 000	4.015E29	3.199E9	6.910E30
2	毫秒(ms)	8 000	2 000	387.122	2.154	6738.856

大部分由 TPC-H 模板产生的查询语句执行开销较小,并不满足均匀覆盖短时间查询和长时间查询的条件.为了产生长时间查询,实验根据数据库中的 8 张表自定义了查询模板.通过改变数据库配置和表的连接顺序,利用模板生成了数万条的查询计划.更改数据库配置,是为了引导查询优化器产生不同的查询计划.实验中随机更改 PostgreSQL 提供的 28 项设置,以引导查询优化器同时产生“好”和“不好”的计划.

特别注意的是:在收集每一条查询计划的运行时间时,都需要在执行该条计划前清理缓存,以确保数据的准确性.在 Linux 上,可以通过关闭 PostgreSQL 数据库,使用 drop_caches 清理系统层面的缓存,再重启 PostgreSQL 的方法清理缓存.另外,训练集在训练前要随机打乱,避免训练数据之间的相关性很大影响训练结果.

4.3 模型评估

本节介绍了两组实验中对模型的评估.表 3、表 4 总结描述了两次实验中网络各层的详细信息.表 3 中输入大小为 120,代表第 3.1 节中被转换成向量 v 的操作长度.实验中的向量 v 有 4 个部分:n_0 有 34 位,代表了 PostgreSQL 数据库中 34 种操作类型;n_1 有 8 位,代表了 TPC-H 中的 8 张表;n_2 有 61 位,代表了表中的所有列;n_3 有 17 位,表示结果行的平均宽度范围被划分成 17 个区间.4 部分总共 120 位.

实验使用 80%的数据作为训练集,20%的数据作为测试集.表 5 中第 1 组实验使用了 12 万条查询计划进行训练,3 万条独立的查询计划测试;第 2 组实验采用了真实运行时间数据,8 000 条查询计划用于训练,2 000 条独立的查询计划用于测试.

公式(12)用于评估模型预测的正确率,Num_{total} 指的是预测的总次数,Num_{hit} 指预测开销所在的区间与真实开销所在区间一致的预测次数.Accuracy 并不代表查询时间值的差异,而是代表了时间开销间隔类别的差异.

$$Accuracy=Num_{hit}/Num_{total} \tag{12}$$

实验结果见表 5:两组实验的正确率都高于 71%,模拟数据训练下模型的正确率高于 78%.分析预测结果发现:不正确的预测结果误差范围较小,对实际应用场景的执行影响很小.模型的训练时间代价见表 6,在本文系统环境下,实验 2 模型的训练代价仅为 1 小时,达到了 71%的正确率.

实验中模型终止迭代评分为 3.78.终止迭代次数见表 6 中,各为 2 000 次.

Table 3 Network layer information of the first experiment

表 3 实验 1 网络层级信息

层级	类型	输入大小	层大小	参数个数	激励函数
1	GravesLSTM	120	100	88 700	*sigmoid*
2	GravesLSTM	100	100	80 700	*sigmoid*
3	RnnOutput	100	10	1 010	*softmax*

Table 4 Network layer information of the second experiment

表 4 实验 2 网络层级信息

层级	类型	输入大小	层大小	参数个数	激励函数
1	GravesLSTM	112	100	85 500	*sigmoid*
2	GravesLSTM	100	100	80 700	*sigmoid*
3	RnnOutput	100	10	1 010	*softmax*

Table 5 Model evaluation results

表 5 模型评估结果

实验组别	数量		正确率(%)
	训练集	测试集	
1	120 000	30 000	78.244
2	8 000	2 000	71.258

Table 6 Model training time cost

表 6 模型训练时间代价

实验组别	迭代次数	时间		
		数据加载	训练时间	测试时间
1	2 000	72.503s	65.035h	458.33s
2	2 000	2.343s	1.001h	6 372ms

图 9 和图 10 是两次实验对于每个子计划预测的命中率,第 1 类的时间区间是(−∞,0],负数的时间实际是没有意义的,因此这个区间只包含 0.为了让每一个计划向量的长度相同,采用最大字长,不足最大长度的计划向量在前面补 0,这样的做法使得训练集和测试集都存在一定数量开销为 0 的操作.例如实验 2 中,2 000 条计划有 2000×6 个操作,其中,开销为 0 的个数为 5 005,命中的数量是 4 977,模型预测的准确率是 99.441%,但由于其数量之大影响了图 10 中其他数据的显示,因此图中没有给出.图 9 同理.

实验还比较了查询优化器与本文模型的预测速度.第 2 组实验中,对于 2 000 条测试数据,查询优化器用时 846ms,本文模型用时 6 372ms.本文模型的应用场景是负载管理,与查询优化器的代价模型有所不同,本文第 2 节给出了详细的说明.因此两者的速度要求并不相同.实验中本文模型用时是查询优化器的 8 倍.但随着深度学习算法的优化,和硬件的提升,未来模型时间性能的提升是必然的.

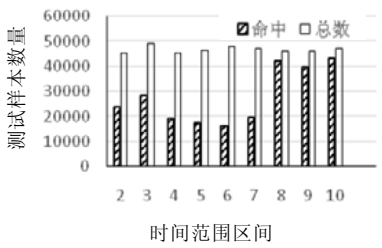

Fig.9 The hit condition of the first experiment

图 9 实验 1 的命中情况

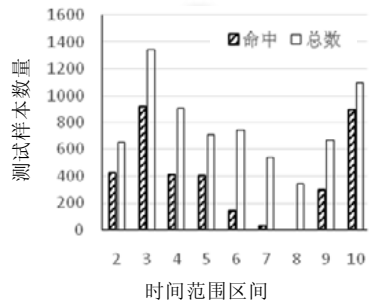

Fig.10 The hit condition of the second experiment

图 10 实验 2 的命中情况

5 结束语

本文提出一种基于循环神经网络的查询开销预测模型.给出一个特定的查询计划,在计划实际执行前,模型就能够产生该查询计划实际运行时间区间的预测,不管是短时间查询还是长时间查询,模型的预测结果都较为准确.特别地,本文设计了一种编码方法,能够将具有复杂结构的查询计划转换成特征向量.通过实验验证,模型的正确率高于 71%,说明了本文模型的可行性.后续工作将继续研究模型在 PostgreSQL 以外的关系型数据库上的性能表现以及如何将模型扩展应用到关系型数据库以外的数据库上,例如 MongoDB、HBase 等.

References:

[1] Wu W, Chi Y, Zhu S, Tatemura J, Hacigümüs H, Naughton JF. Predicting query execution time: Are optimizer cost models really unusable? In: Proc. of the 2013 IEEE 29th Int'l Conf. on Data Engineering (ICDE). New York: IEEE, 2013. 1081−1092. [doi: 10.1109/ICDE.2013.6544899]

[2] Tozer S, Brecht T, Aboulnaga A. Q-Cop: Avoiding bad query mixes to minimize client timeouts under heavy loads. In: Proc. of the 2010 IEEE 26th Int'l Conf. on Data Engineering (ICDE). New York: IEEE, 2010. 397−408. [doi: 10.1109/ICDE.2010.5447850]

[3] Xiong P, Chi Y, Zhu S, Tatemura J, Pu C, HacigümüŞ H. ActiveSLA: A profit-oriented admission control framework for database-as-a-service providers. In: Proc. of the 2nd ACM Symp. on Cloud Computing. New York: ACM Press, 2011. 15. [doi: 10.1145/2038916.2038931]

[4] Mishra C, Koudas N. The design of a query monitoring system. ACM Trans. on Database Systems (TODS), 2009,34(1):1−51. [doi: 10.1145/1508857.1508858]

[5] Wasserman TJ, Martin P, Skillicorn DB, Rizvi H. Developing a characterization of business intelligence workloads for sizing new database systems. In: Proc. of the 7th ACM Int'l Workshop on Data Warehousing and OLAP. New York: ACM Press, 2004. 7−13. [doi: 10.1145/1031763.1031766]

[6] Ganapathi A, Kuno H, Dayal U, Wiener JL, Fox A, Jordan M, Patterson D. Predicting multiple metrics for queries: Better decisions enabled by machine learning. In: Proc. of the 2009 IEEE Int'l Conf. on Data Engineering (ICDE 2009). New York: IEEE, 2009. 592−603. [doi: 10.1109/ICDE.2009.130]

[7] Elnaffar S, Martin P, Horman R. Automatically classifying database workloads. In: Proc. of the 11th Int'l Conf. on Information and Knowledge Management (CIKM 2002). New York: ACM Press, 2002. 622−624. [doi: 10.1145/584792.584898]

[8] Akdere M, Çetintemel U, Riondato M, Upfal E, Zdonik SB. Learning-Based query performance modeling and prediction. In: Proc. of the 2012 IEEE 28th Int'l Conf. on Data Engineering. New York: IEEE, 2012. 390−401. [doi: 10.1109/ICDE.2012.64]

[9] Ahmad M, Aboulnaga A, Babu S, Munagala K. Interaction-Aware scheduling of report-generation workloads. The VLDB Journal—The Int'l Journal on Very Large Data Bases, 2011,20(4):589−615. [doi: 10.1007/s00778-011-0217-y]

[10] Duggan J, Cetintemel U, Papaemmanouil O, Upfal E. Performance prediction for concurrent database workloads. In: Proc. of the 2011 ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2011. 337−348. [doi: 10.1145/1989323.1989359]

[11] Wang W, Zhang M, Chen G, Jagadish HV, Ooi BC, Tan KL. Database meets deep learning: Challenges and opportunities. ACM SIGMOD Record, 2016,45(2):17−22. [doi: 10.1145/3003665.3003669]

[12] Pavlo A, Angulo G, Arulraj J, Lin H, Lin J, Ma L, Santurkar S. Self-Driving database management systems. In: Proc. of the 8th Biennial Conf. on Innovative Data Systems Research (CIDR). 2017.

[13] Gers FA, Schmidhuber J, Cummins F. Learning to forget: Continual prediction with LSTM. Neural Computation, 2000,12(10):2451−2471. [doi: 10.1162/089976600300015015]

[14] Gers FA, Schraudolph NN, Schmidhuber J. Learning precise timing with LSTM recurrent networks. Journal of Machine Learning Research, 2003,3(1):115−143. [doi: 10.1162/153244303768966139]

[15] Ioannidis YE. Query optimization. ACM Computing Surveys (CSUR), 1996,28(1):121−123. [doi: 10.1145/234313.234367]

[16] Scott DW. On optimal and data-based histograms. Biometrika, 1979,66(3):605−610. [doi: 10.1093/biomet/66.3.605]

[17] Ioannidis YE, Poosala V. Balancing histogram optimality and practicality for query result size estimation. ACM SIGMOD Record, 1995,24(2):233−244. [doi: 10.1145/568271.223841]

[18] Haas PJ, Naughton JF, Seshadri S, Stokes L. Sampling-Based estimation of the number of distinct values of an attribute. In: Carey M, ed. Proc. of the 21th Int'l Conf. on Very Large Data Bases (VLDB'95). New York: ACM Press, 1995. 311−322. [doi: 10.1145/223784.223841]

[19] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.

[20] Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.

[21] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. In: Ghahramani Z, ed. Proc. of the 27th Int'l Conf. on Neural Information Processing Systems (NIPS 2014). Massachusetts: MIT Press Cambridge, 2014. 3104−3112.

[22] TPC-H benchmarks. http://www.tpc.org

[23] Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks, 2005,18(5):602−610.

毕里缘(1993－),女,上海人,硕士生,主要研究领域为数据库.

寿黎但(1974－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为数据库.

伍赛(1980－),男,博士,副教授,CCF 专业会员,主要研究领域为数据库.

陈珂(1977－),女,博士,副研究员,CCF 专业会员,主要研究领域为数据库.

陈刚(1973－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为数据库.

胡天磊(1982－),男,博士,副教授,主要研究领域为数据库.