中国人民大學

RENMIN UNIVERSITY OF CHINA

大数据计算智能大作业

卷积神经网络预测查询规模

学　　院:	信息学院
专　　业:	计算机科学与技术
年　　级:	2018 级
学　　号:	2018202195,2018202181
学生姓名:	邵宁录, 思子华
完成日期:	2020.12.14

目录

1 问题描述

根据 sql 查询语句预测查询规模是本次实验的核心内容。

数据库领域中的查询优化 (*Query optimization*) 是基于查询规模估算 (*cardinality estimation*) 的。为了能够在符合实际应用的需求的前提下，查询优化需要在较短时间内获知查询结果的规模，而处理速度和结果的准确性往往是 *trade-off* 的，尤其是在处理跨表链接的情形时，这是查询规模估算的最大挑战。

查询规模估计问题可以被转化成一个有监督的机器学习问题，只需要将输入的 *query* 抽象出特征，再将估计的规模作为输出即可。因此本次实现综合考虑了本学期所有学习过的机器学习算法和深度学习算法，尤其是着重考虑了神经网络能否在该问题上得到应用，并最终成功构建出了一个基于深度学习的、类似于卷积神经网络的模型来解决该问题，并取得了不错的成绩。

2 思路摘要

在本次实验中, 我们实现了一个基于深度学习的卷积网络，用于查询规模估算。

我们的模型将表示 *query* 的 *sql* 语句分成了三类，分别是 *table*(表),*Join*(连接),*Predicate*(谓词)。我们使用集合的方式来表达 *query* 的特征，例如 $(A \bowtie B) \bowtie C$ 或者 $A \bowtie (B \bowtie C)$ 都可以表示为 $\{A, B, C\}$。这样一来，我们的模型就可以将 $(A \bowtie B) \bowtie C$ 或者 $A \bowtie (B \bowtie C)$ 表示成为同一种连接，事实上它们的效果也确实是一样的，这样以来可以有效地减少内存的浪费，并提高预测准确率。

更直观地讲，我们将 *query* 采用如下的形式表示。

$$q = (T_q, J_q, P_q), where\ q \in Q, T_q \in T, J_q \in J, P_q \in P \tag{1}$$

其中 Q 表示所有 *query* 组成的集合, T 表示所有 *table* 构成的集合, J 表示所有的 *join* 构成的集合, P 表示所有的 *predicate* 构成的集合。我们还对 *table,join,Predicate* 进行了独热编码。

我们从近几年的一篇论文 Deep Sets[1] 中获得了一些启发, 用神经网络模型来对集合进行表示。我们使用了简单的全连接多层神经网络 (*fully-connected multi-layer neural networks*) 来将任意的集合映射成为向量。本模型对于每一个集合单独的学习了一个映射（使用共享的参数），这种方式本质上和 *CNN* 中的 1×1 *convolution* 是类似的。本模型对于上文提到的 query 中的三类特征（*Table, Join,Predicate*）分别使用全连接多层神经网络进行表示，再将表示结果作为输入，使用一个全连接多

层神经网络进行预测。上述的模型的逻辑可以简单表达为:

$$\begin{aligned} Table\ module:\ input\ is\ T_q,\ output\ is\ w_T \\ Join\ module:\ input\ is\ J_q,\ output\ is\ w_J \\ Predicate\ module:\ input\ is\ P_q,\ output\ is\ w_P \\ Merge\ \&\ predict\ module:\ input\ is\ w_T, w_J\ and\ w_P, output\ is\ w_{out} \end{aligned} \tag{2}$$

其具体的实现方式将在后文中详细介绍。

3 思路分析

标准的深度学习的神经网络的框架，比如说 *CNN*、*RNN* 和 *MLP* 等都无法直接读取本次大作业的输入数据，所以需要对输入数据进行转化，将 sql 的查询语句转化为一种有序的序列。因此，我们将数据进行了独热编码 (*one-hot encoding*) ，具体方式在特征工程部分介绍。

我们对于 query 的表达方式包含了集合，这促使我们对于每一个 *set* S，学习了一个神经网络 $MLP_S(v_s)$，其中 MLP_S 是全连接多层神经网络 (*fully-connected multi-layer neural networks*) 的缩写，即对于每一个特征向量 $v_s, where\ s \in S$ 我们都学习了一个神经网络 MLP。然后，我们将每一个 query 中不同 set 的结果进行了线性组合，并将 Table、Join、Predicate 部分的结果作为最终的预测模型输入。具体的实现方式在算法模型部分给出。

我们还对数据进行了归一化处理，我们对训练数据的输出 *cardinalities* 先取对数函数，再使用最大值和最小值正则化为 $[0, 1]$ 之间的值，这使得数据的分布更加地均匀。而且显然这个归一化的过程是可逆的，所以在预测结束以后，只需要逆向处理就可以得到非正则化的预测结果。

我们使用最小化平均 *q-error* 的方式训练模型，*q-error* 是估计值和真实值之间的因数（反之亦然）[2]。更进一步地，我们使用 Adam 来随机梯度下降。

4 特征工程

如前文所说，我们将每一个 $Table$,$Join$,$Predicate$ 都作为集合考虑，并将它们分别进行了独热编码(one-hot)。如下所示，下面的式子是将 $select * from\, title\, t, movie\_inf$ $mi.movie\_id\ AND\ production\_year > 2019\ AND\ t.kind\_id = 5$ 进行集合化的独热编码的一个例子。

$$\begin{aligned} title\ t:&\ table\ set\{[0101...010]\} \\ movie\_info\ mi:&\ table\ set\{[1111...100]\} \\ t.id = \ mi.movie\_id:&\ join\ set\{[0010]\} \\ production\_year > 2010:&\ predicate\ set\{[10000\ 100\ 0.72]\} \\ t.kind\_id = 5:&\ predicate\ set\{[00010\ 010\ 0.14]\} \end{aligned} \tag{3}$$

可以看到除了简单的读热编码，我们还对谓词 (Predicate) 中的数字进行了归一化处理。谓词中最后一项数值，被归一化为了一个 $\in [0, 1]$ 的值。

归一化的方式如下：

$$\begin{gathered} given\ value\ v,\ calculate\ normalized\ v_{norm}: \\ v_{norm} = \frac{v - v_{min}}{v_{max} - v_{min}} \\ where\ v_{min}\ is\ min\ value\ of\ this\ column, \\ v_{max}\ is\ max\ value\ of\ this\ column. \end{gathered} \tag{4}$$

除此以外，为了方便模型调参，我们还将训练集的 cardinalities 进行了归一化

处理，我们首先使用对数函数来使得数据分布得更加均匀，之后再用最大值和最小值将数据归一化到 $[0,1]$ 这个区间上。具体的过程如下：

$$
\begin{aligned}
&given\ value\ t,\ calculate\ normalized\ t_{norm}:\\
&v_{norm} = \frac{log(t) - log(t)_{min}}{log(t)_{max} - log(t)_{min}}\\
&where\ log(t)_{min}\ is\ min\ \log\ value\ of\ this\ column,\\
&log(t)_{max}\ is\ max\ \log\ value\ of\ this\ column.
\end{aligned}
\tag{5}
$$

显而易见，上述的将 cardinality 归一化的过程是可逆的，所以在预测出结果后，逆向处理一下就可以得到真正的预测结果。

5 算法模型

在模型选择的大方向上，我们选择了深度神经网络作为我们的选择方向。

我们参考了许多现有的标准的神经网络框架，如卷积神经网络（CNN）、循环神经网络（RNN）和多层感知机（MLP）等。但很显然，现有的框架并不能很好的解决我们当前的这个问题，因为我们的特征维度比较高、信息分布比较分散，所以理想的模型应该最好能够做到像 CNN 一样有一个类似于卷积层（$ConvolutionalLayer$）的降维操作。并且 SQL 查询语句具有前后关联性，所以理想的模型应该最好也能够做到像 RNN 一样具有记忆功能，做到前后文相关联。这就要求了理想的模型能够学习发现特征向量的结构，比如，它能够学习发现特征向量在不同集合中的的边界，并且这个集合中序列化的元素是任意的。

因此我们选择采用了 $multi-setconvolutionalnetwork$（MSCN）。该模型的结构如下图所示：

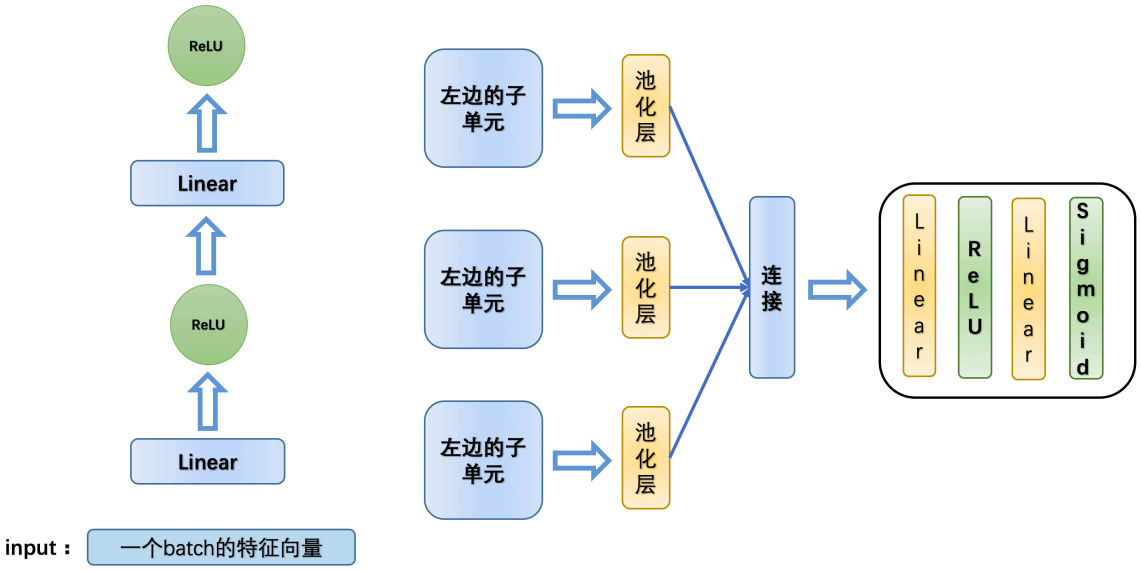

图 1: MSCN 示意图

如上图所示，该模型的输入端接收一个 batch，然后将 batch 依次经过：线性层（Linear）、ReLU、线性层、ReLU。再将所有的 batch 做了一个池化（Pool，这里的池化方式是取平均值）。最后将多个端做一个向量的连接，作为新的特征输入到输出层。输出层分别依次由线性层、ReLU、线性层、Sigmoidz 组成。

由于最后是经过一个 Sigmoid 函数，因此我们的输出会是一个 $[0, 1]$ 的值。由于刚开始输入规模 y 是通过 $Min - Max$ 归一化的，因此需要通过逆向运算来获得实际的预测出的规模。

6 结果分析

6.1 训练过程

训练数据共有 100000 条，测试数据共有 5000 条。训练前，我们把训练数据划分为训练集与测试集。其中，训练集有 90000 条，测试集有 10000 条。

训练时，我们把训练集分为 90 个 batch，每个 btach 的大小为 1000。由于算力与时间有限，因此我们只尝试了以下几组参数，分别为：模型的隐藏层大小为 128，训练时的 epoch 为 50；模型的隐藏层大小为 128，训练时的 epoch 为 100；模型的隐藏层大小为 256，训练时的 epoch 为 100。分别如下表格所示。

表 1: 参数设置

Index	Number of Hidden Units	Number of Epochs
0	128	50
1	128	100
2	256	100

在查阅各类参考文献的时候，我们发现有经验的研究者会在测试完分出一部分测试集后，再将测试集喂回到模型中去，使得模型的拟合效果更好。我们参考了这一方法，在测试完测试集后，将其重新输入到模型中跑了一轮。

6.2 测试结果

为比较各组参数的好坏，我们输出了各组参数每一轮的 $Loss$，并将它们画成折线图比较。几组参数的结果如下所示。为方便比较，去掉了第一次的 $Loss$（因为第一次的 $Loss$ 都会很大），并且每隔 5 个数进行采样。

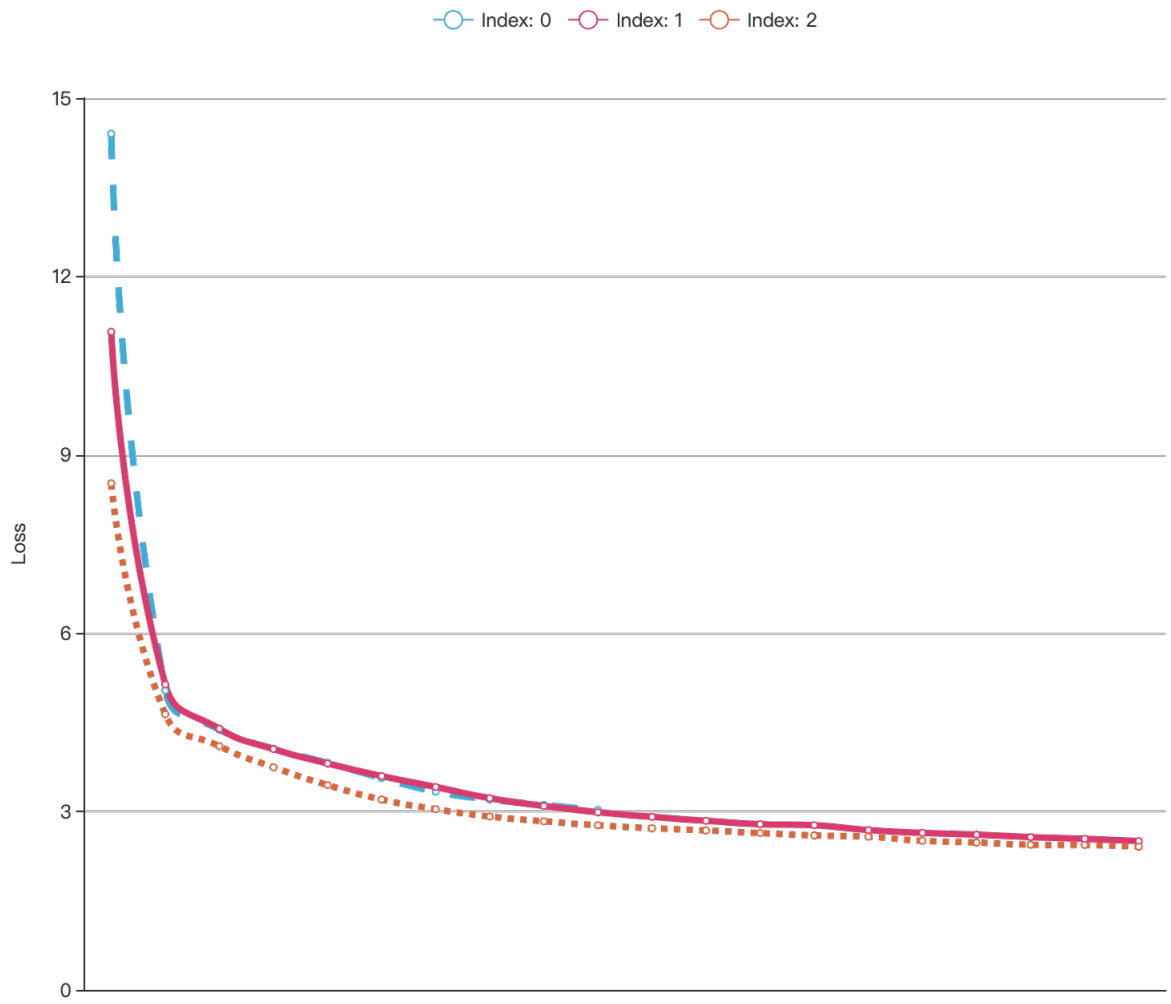

图 2: 各组的 Loss 曲线

其中我们可以看到，各组之间的曲线是很接近的，在最后都几乎收敛到一个非常相近的数值。相对而言，第 2 组的 *Loss* 比其他两组都低一些。

我们又考察了各组的在测试集上的 *q-error*，结果如下。

表 2: q-error (mean)

Index	q-error (mean)
0	3.623
1	2.506
2	2.563

我们可以看到，在 *q-error* 的比较下，除了第 0 组有明显差异外，其他两组的差异其实非常的小。因此，在综合考虑了训练效果、过拟合问题以及训练的时间之后，我们选择了第 1 组作为我们训练参数。

7 References

参考文献

[1] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In Advances in Neural Information Processing Systems, 2017.

[2] G. Moerkotte, T. Neumann, and G. Steidl. Preventing bad plans by bounding the impact of cardinality estimation errors. PVLDB, 2(1):982–993, 2009.