Auto-Differentiation of Relational Computations for Very Large-Scale Machine Learning

Yuxin Tang[1], Zhimin Ding[1], Dimitrije Jankov[1], Binhang Yuan[23], Daniel Bourgeois[1], Chris Jermaine[1]
[1]Rice University, USA [2]ETH Zürich, Switzerland [3]HKUST

Motivation

Auto-Differentiation (auto-diff) has been a key component in modern machine learning systems (JAX, PyTorch, Tensorflow)

- Forward Pass is specified by user. Backward Pass is generated by system
- The process of evaluating gradient can be error-prone and tedious

However, current auto-differentiation library is only based on Linear Algebra
What if differentiating ML computations in Relational Algebra?

Central Questions

(1) How to auto-differentiate arbitrary computation in Relational Algebra?

(2) Database systems (DBMS) are built on top of Relational Algebra. If DBMS are equipped with differentiation ability, what are the benefits?

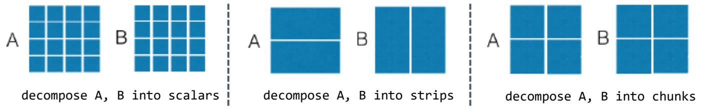
Background: Database for Machine Learning

Relational Algebra is the theoretical foundation for SQL, which is the query language on top of modern relational databases. For example, a distributed matrix multiplication can be specified in Relational Algebra/SQL:

```
A (row,col,value*)
B (row,col,value*)

1 SELECT A.row, B.col, matrix_multiply (A.value, B.value)
2 FROM A, B
3 WHERE A.col = B.row
4 GROUP BY A.row, B.col
```

value can be a scalar, a vector or a multi-dimensional array (tensor)



Benefits:

- Declarative Interface
- Automatic Parallelization, Distribution and Optimization
- No data export/import overhead
- Easy to scale to large-scale datasets and models

Vector Database support ML:

Pinecone milvus zilliz BigQuery amazon REDSHIFT mldb.ai

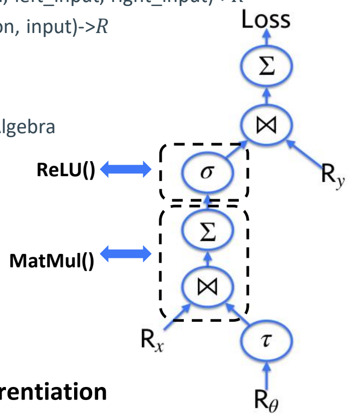
Functional Relational Algebra

∇ denotes the vector differential operator which is a high-order function, requiring functions as input and output. We define relational algebra operations are high-order functions:

- Select (σ):(predicate, projection, kernel_function, input)->R
- Join (⋈): (predicate, projection, kernel_function, left_input, right_input)->R
- Aggregation (Σ): (group_function, kernel_function, input)->R
- TableScan (τ): (key)->R

Example: a simple Logistic Regression in Relational Algebra

- Features are stored in R_x
- Labels are stored in R_y
- Coefficients are stored in R_Θ



Relational Algebra Auto-Differentiation

We derive Relation-Jacobian Products (RJP) for Select (σ), Join (⋈), Aggregation (Σ), TableScan (τ) in relational domain. They are analogous to Vector-Jacobian Product (VJP) in Linear Algebra

- The kernel functions can be differentiated by utilizing JAX/autograd
- RJP rules are implemented efficiently without materialization of Jacobian

RJP for Aggregation (Σ):

- One Join (⋈) or one Select (σ)

RJP for Select (σ):

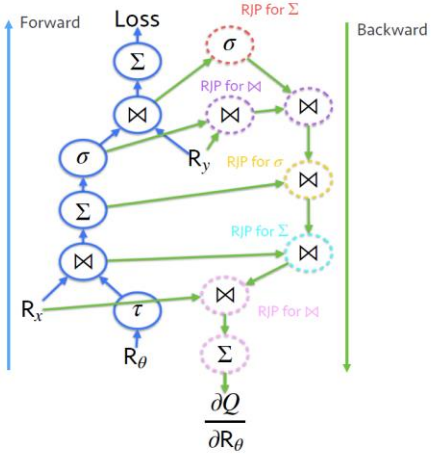
- One Join (⋈)

RJP for Join (⋈):

- Two Joins (⋈) with one aggregation (Σ)

RJP for TableScan (τ):

- Self



For more RJP optimizations and the equivalence to VJP Please refer to our paper.

Main Experiment Results

Experiment settings: AWS m5.4xlarge instances (1 to 16 nodes).
All the Implementation is on top of a relational database engine - plinycompute

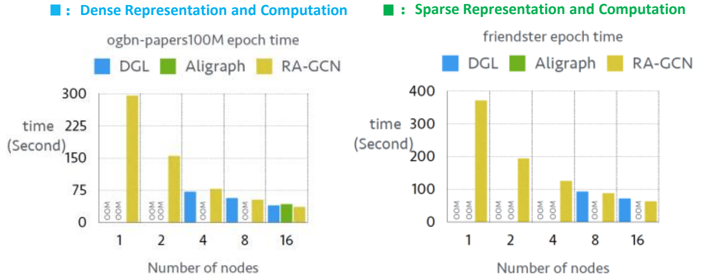
- Non-Negative Matrix Factorization (RA-NNMF)
- **Graph Convolutional Networks (RA-GCN)**
- Large-scale Knowledge Graph Embeddings (RA-KGE)

Baseline systems for GCN: DGL, Aligraph (graph-learn)
Datasets: ogbn-papers100M (N=0.1B, E=1.6B), friendster (N=65.6M, E=3.6B)

Graph Convolution in Relational Algebra/SQL

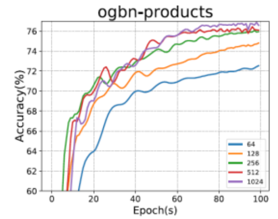
```
Node (ID INT , vec VECTOR [2048])
Edge (sourceID INT , destID INT)

SELECT n1.ID as n.ID , ReLU(MAT_MUL(AVG(Normalize(n2.vec)))) as n.vec
FROM Node as n1 , Edge as e , Node as n2
WHERE n1.ID = e.sourceID and n2.ID = e.destID
GROUP BY n1.ID
```



Boost performance through Large Embedding (64 to 1024 embedding vector)

- 1024 embedding can only be handled by RA-GCN



Key findings:

(1) A relational system, equipped with this auto-diff technology, could show better scalability than other systems, even special purposed ML engines.

(2) RA-GCN is the only one that can handle graph preprocessing, graph loading and training without any OOM error.

Further details and the relevant code:

https://github.com/yuxineverforever/Relational-Algebra-Auto-Differentiation