Data Stream Query Optimization Using Deep Reinforcement Learning

Final presentation

Apoorva Tamaskar

Boston University
Metropolitan College

01 February 2021

Outline

Section 1

Introduction and Related Work

SQL: Introduction

Given a SQL query, how is it executed?

```sql
        SELECT MovieTitle
        FROM StarsIn
        WHERE StarName IN(
            SELECT name
            FROM MovieStar
            WHERE birthdate LIKE '%1960'
        );

```

Listing 1: SQL query to execute.

SQL: Pipeline

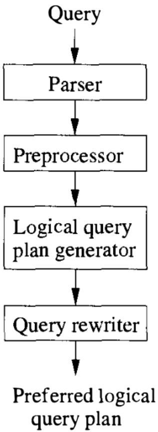

Figure: The pipeline for query processing

SQL: Parser

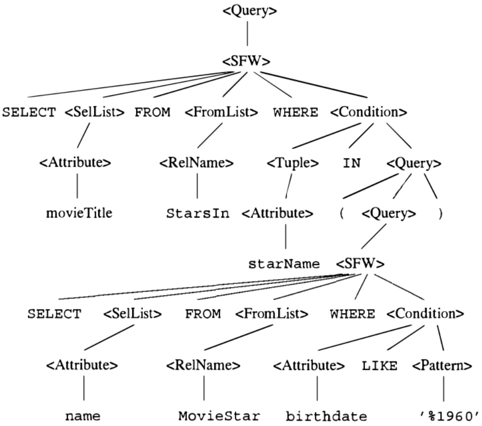

Figure: An example of parse tree

SQL: Preprocessing

Sanity checking

1. Check relation uses
2. Check and resolve attribute uses.
3. Check types.

SQL: Relational Algebra

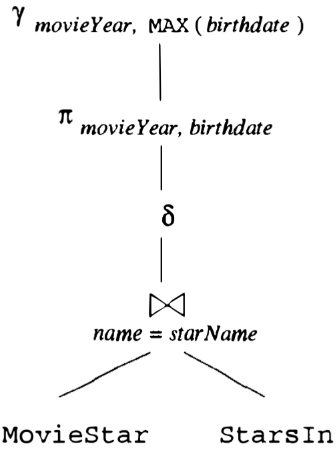

Figure: An example of join being optimized

SQL: Relational Algebra

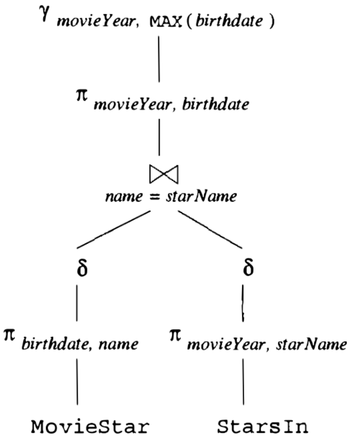

Figure: An example of Relational Algebra being used for optimization

SQL: Rewrite

Need to convert the query plan using Relational Algebra into a query plan which requires has lowest cost according to estimates.

SQL: Cost: Estimation

1. Give accurate estimates. No matter what method is used for executing query plans.
2. Are easy to compute.
3. Are logically consistent.

SQL: Cost: Methods

1. Histogram
2. Heuristics
3. Top-Down
4. Bottom-up
5. Dynamic Programming
6. Branch-and-Bound
7. Hill Climbing
8. Selinger-Style Optimization

SQL: Cost: Considerations

1. An order and grouping for associative-and-commutative operations
2. An algorithm for each operator
3. Additional operators - scanning, sorting
4. The way in which arguments are passed

SQL: Joins

1. Algorithm Nested-loop join
2. Algorithm Index join
3. Order Dynamic Programming
4. Order Greedy

SQL: Physical Plan

1. Selection, Index based
2. Selection, Table Scan
3. Join, One-pass join
4. Join, Hash join
5. Join, Index join

Stream Query Optimization

Modifying queries by changing graph topology and/ or operators to get better performance, as measured by

1. Throughput
2. Latency
3. Resource Usage.

While preserving semantice of the original query.

Stream Graph

Queries on Data streams can be thought of as directed graphs,

1. Edges represent streams and nodes represent operators.
2. Root and leaf nodes are called sources and sinks, respectively.

Whereas for traditional Databases, we have parse trees.

Possible Optimizations

- Batching
- Placement
- State sharing
- Load Balancing
- Algorithm selection
- Load Shedding
- Fusion
- Operator Separation
- Operator Reordering
- Redundancy elimination
- Fission

Operator Reordering

A reordering operator moves more selective operators, which reduce the data volume upstream. This has the benefit of reducing the amount of data flowing into downstream compuration. Thus eliminatin unnecessary work. We focus on finding the optimal method of executing associative opeators.

Section 2

Research Question

Research Question

Is it possible to apply deep reinforcement learning to find the optimal ordering of executing associative operations?

Section 3

Concept and Implementation

Deep Neural Networks

Deep neural networks is a layer wise combinations of neurons
Building up on the neuron seen in the last slide. We have

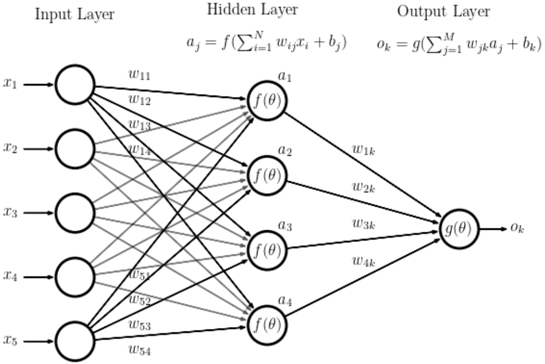

Figure: Example of a deep neural network

Reinforcement Learning

What is reinforcement learning? How is it different from supervised and unsupervised learning?

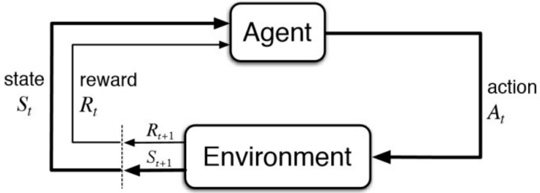

Figure: Example of a framework of a reinforcement learning agent

Optimal Policy

```
for s in S:
    V(s)=0
while(not converged):
    for s in S:
        V(s)=R(s)+max over all action[gamma*(sum(P(s,a,s')V(s')))]

```

Listing 2: value iteration algorithm

```
initialize random pi
while(not converged):
    V=V(pi)
    for s in S:
        pi(s)=max over all actions[sum(P(s,a,s')V(S'))]

```

Listing 3: Policy iteration algorithm

Deep Reinforcement Learning

Deep reinforcement learning combines Deep learning and reinforcement learning

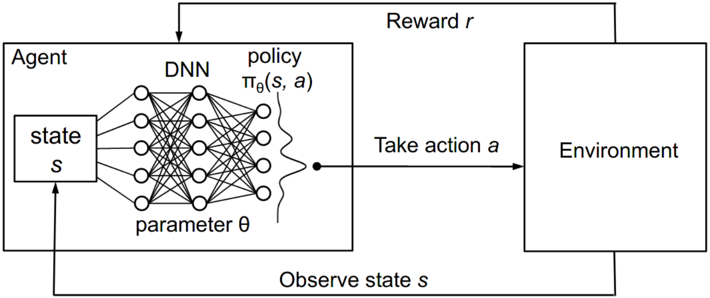

Figure: Deep Reinforcement learning(DQN) framework

Linear Road

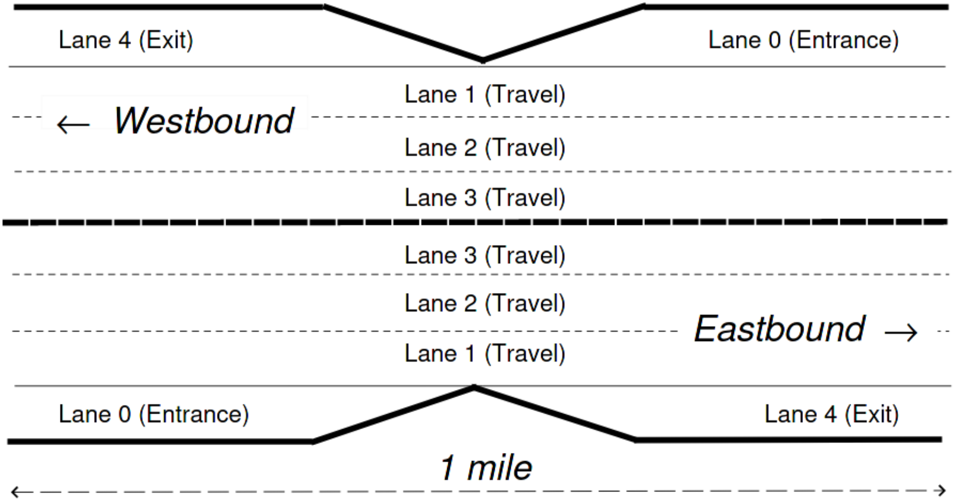

Figure: An Example Expressway Segment

Linear Road: Query to execute

The goal is to execute the SegToll Query.

```
SELECT car_id, exp_way, dir, seg
FROM CarSegStr [PARTITION BY car_id ROWS 1], CurActiveCars
WHERE CarSegStr.car_id = CurActiveCars.car_id;

SELECT exp_way, dir, seg, AVG(speed) as speed,
FROM CarSegStr [RANGE 5 MINUTES]
GROUP BY exp_way, dir, seg;

SELECT exp_way, dir, seg, COUNT(*) as volume
FROM CurCarSeg
GROUP BY exp_way, dir, seg;

SELECT S.exp_way, S.dir, S.seg, basetoll*(V.volume-150)*(V.volume-150)
FROM SegAvgSpeed as S, SegVol as V
WHERE S.exp_way = V.exp_way and S.dir = V.dir and S.seg = V.seg
      and S.speed <= 40;
```

Listing 4: SEGTOLL linear road query

Implementation: Example of Query Plan

$C_1 = (S.exp\_way = V.exp\_way)$
$C_2 = (S.dir = V.dir)$
$C_3 = (S.seg = V.seg)$
$C_4 = (S.speed <= 40)$
There are total of $4! = 24$ possible orderings.
Few different query plans are

$$\pi_{\text{S.exp\_way, S.dir, S.seg, S.toll}}(\sigma_{C_1}(\sigma_{C_2}(\sigma_{C_3}(\sigma_{C_4}(\text{S,V})))))$$

$$\pi_{\text{S.exp\_way, S.dir, S.seg, S.toll}}(\sigma_{C_1}(\sigma_{C_2}(\sigma_{C_4}(\sigma_{C_3}(\text{S,V})))))$$

$$\pi_{\text{S.exp\_way, S.dir, S.seg, S.toll}}(\sigma_{C_1}(\sigma_{C_3}(\sigma_{C_2}(\sigma_{C_4}(\text{S,V})))))$$

Implementation: Query Execution

1. Mimicked the execution of query in C++.
2. Given there are 24 possible orderings, executed all of them and recorded how much time and the number of operations they took.
3. For each window of data store the column wise entropy and size of tables.

Implementation: DQN

Training

1. Input the column wise entropy, the size of tables and ordering of operations.
2. The reward for the training is taken to be the number of operations required.

Prediction

1. Predict the reward for each ordering, for given entropy vector+ table size.
2. The rewards represent number of operations required.
3. Check which ordering has least number of operations, this ordering is optimal.

Testing

1. Check if prediction for a test data point is same as the actual optimal ordering.

Justification

The things considered while determining the neural network to use for training the DQN are :-

- Value of loss function
- Time for prediction/ Complexity of model
- Resources for training

Justification

What we found was adding layers improves the accuracy of prediction of the optimal moves but not by significant margin.
These are only query and data specific findings.

Section 4

Evaluation

Linear Road Data

Query Type, Time stamp, vehicle ID, speed, expressway, lane, direction, segment, position, query ID, start segment, end segment, day of week, minute of day, day in past 10 weeks

```
0,0,13,10,8,0,0,89,469920,-1,-1,-1,-1,-1,-1
0,0,17,10,8,0,1,65,348479,-1,-1,-1,-1,-1,-1
0,0,22,10,8,0,0,12,63360,-1,-1,-1,-1,-1,-1
0,0,33,10,8,0,1,94,501599,-1,-1,-1,-1,-1,-1
0,0,42,10,8,0,0,14,73920,-1,-1,-1,-1,-1,-1
0,0,4,10,7,0,0,61,322080,-1,-1,-1,-1,-1,-1
0,0,85,10,8,0,1,30,163679,-1,-1,-1,-1,-1,-1
0,0,11,10,6,0,1,41,221759,-1,-1,-1,-1,-1,-1
0,0,23,10,7,0,1,81,432959,-1,-1,-1,-1,-1,-1
0,0,15,10,6,0,0,5,26400,-1,-1,-1,-1,-1,-1
```

Listing 5: Example of Linear Road Data

Data Distribution

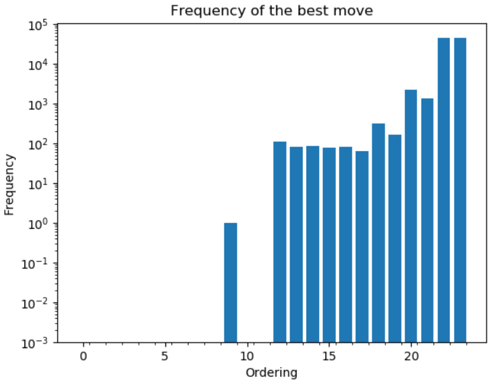

Figure: Shows for each ordering of operations, the number of data windows for which it was optimal, the data is baised towards the last 2 orderings.

Data Distribution

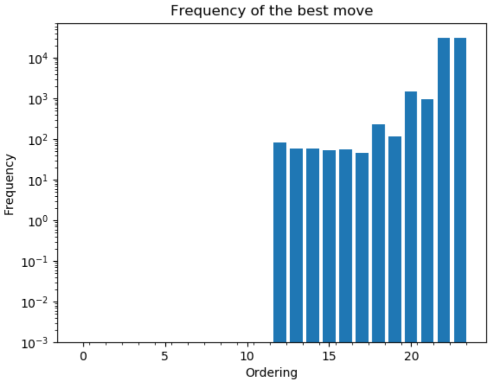

Figure: Shows for each ordering, the number of data windows in trainging data for which it was optimal, the data was divided into 70%:30% for training and testing

Data Distribution

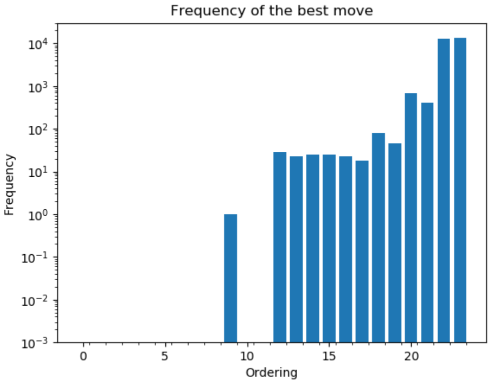

Figure: Shows for each ordering, the number of data windows in testing data for which it was optimal, the data was divided into 70%:30% for training and testing

Confusion Matrix

```
[[   0    0    0    0    0    0    0    0    0    0    0    1    0]
 [   0    0    0    0    0    0    0    0    0    0    0   16   13]
 [   0    0    0    0    0    0    0    0    0    0    0   18    5]
 [   0    0    0    0    0    0    0    0    0    0    0   13   12]
 [   0    0    0    0    0    0    0    0    0    0    0   12   13]
 [   0    0    0    0    0    0    0    0    0    0    0   17    6]
 [   0    0    0    0    0    0    0    0    0    0    0    9    9]
 [   0    0    0    0    0    0    0    0    0    0   10   21   49]
 [   0    0    0    0    0    0    0    0    0    3    4   13   26]
 [   0    0    0    0    0    0    0    1    0   18   77  213  374]
 [   0    0    0    0    0    0    0    0    0   19   69  118  195]
 [   0    0    2    0    0    0    0    0    0  116  406 6347 6167]
 [   0    0    1    0    0    0    0    6    0  113  476 6338 6355]]
```

Listing 6: Confusion matrix for DQN classificaiton

Confusion Matrix

The predictions on trained model lead to the following confusion matrix.

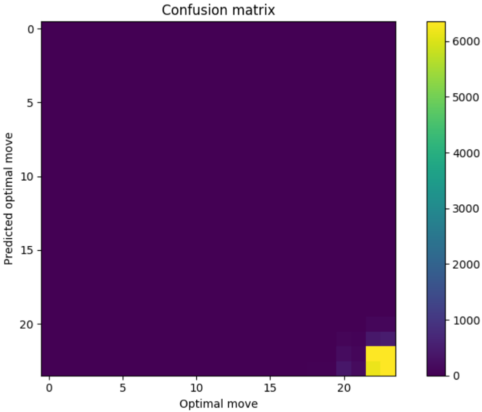

Figure: DQN predictions visualized as confusion matrix

Predictions

Some of the cases we were able to predict the correct answers.

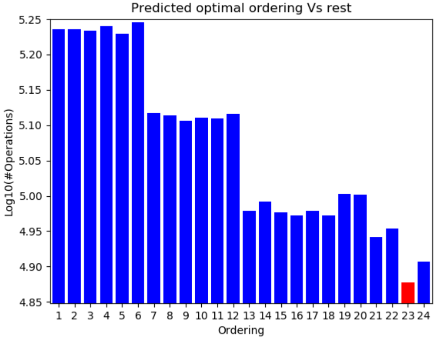

Figure: The figure shows the $\log_{10}$(number of operations) required to execute the query depending on the ordering of the selection operators chosen. The predicted optimal ordering is shown in red.

Predictions

Some of the cases we were able to predict the correct answers.

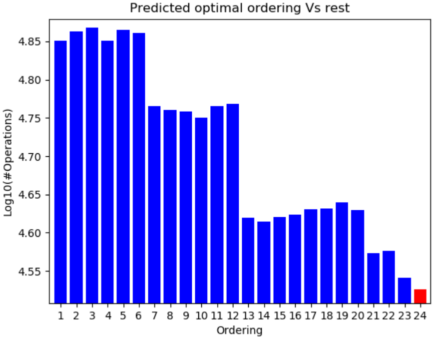

Figure: The figure shows the $\log_{10}$(number of operations) required to execute the query depending on the ordering of the selection operators chosen. The predicted optimal ordering is shown in red.

Section 5

Conclusion and Future work

Overall accuracy

The model predicted optimal move correctly for 12978 data points, out of 27681, i.e. 47%.

Comparison

1. The sum of number of operations required by the optimal ordering 890640075.0
2. The sum of number of operations required by the predicted ordering 900269878.0
3. The sum of number of operations required by the optimal ordering 2310227856.0

Performance

Our model resulted in requiring 39% of operations as the query execution would have required if it executed the worst ordering every time.
Our model resulted in requiring 102% of operations as the query execution would have required if it executed the optimal ordering every time.

Further Work

- Online training of DQN.
- Integration of DQN into stream processing systems.

Questions?