Data Stream Query Optimization Using Deep Reinforcement Learning

Final presentation

Apoorva Tamaskar

Boston University
Metropolitan College

01 February 2021

Outline

1 Introduction and Related Work

2 Research Question

3 Concept and Implementation

4 Evaluation

5 Conclusion and Future work

Section 1

Introduction and Related Work

SQL: Introduction

Given a SQL query, how is it executed?

```
1       SELECT MovieTitle
2       FROM StarsIn
3       WHERE StarName IN(
4           SELECT name
5           FROM MovieStar
6           WHERE birthdate LIKE '%1960'
7       );
8
```

Listing 1: SQL query to execute.

SQL: Pipeline

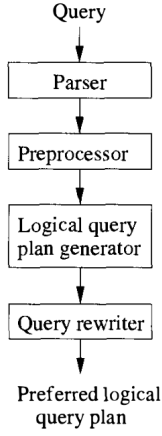


Figure: The pipeline for query processing

SQL: Parser

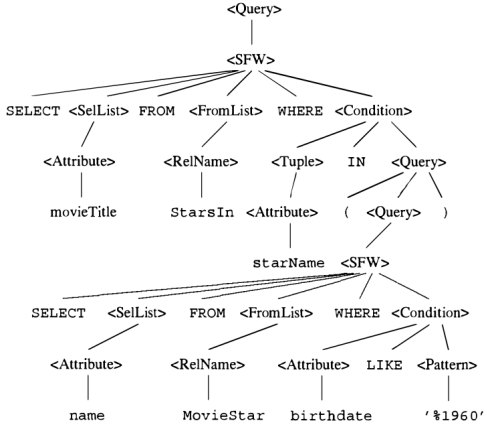
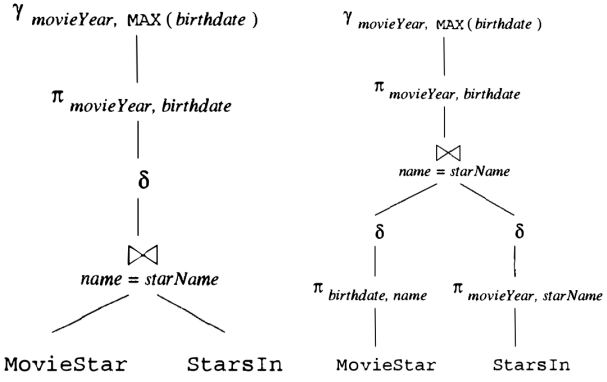


Figure: An example of parse tree

SQL: Rewrite

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

SQL: Relational Algebra




```sql
SELECT movieYear, MAX(birthdate)
FROM MovieStar, StarIn
WHERE name=starName
GROUP BY movieYear;

```

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 Calculation 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: Branch and Bound

```
1    cost_upper_limit= +inf
2    plan=[]
3    while(can explore):
4        if node has unexplored neighbours:
5            explore node neighbour
6            calculate cost of current plan
7            if cost of plan >=cost_upper_limit:
8                backtrack
9            else:
10               update plan
11           if(complete plan and cost of plan < cost_upper_limit):
12               optimal_plan=current_plan
13               cost_upper_limit = cost of plan
14       else:
15           mark node as explored completely
16           backtrack
17   return optimal_plan
18
```

SQL: Physical Plan

1. Algorithm selection(e.g. index scan, table scan, join method)
2. Materialized or pipelined
3. How relations are accessed

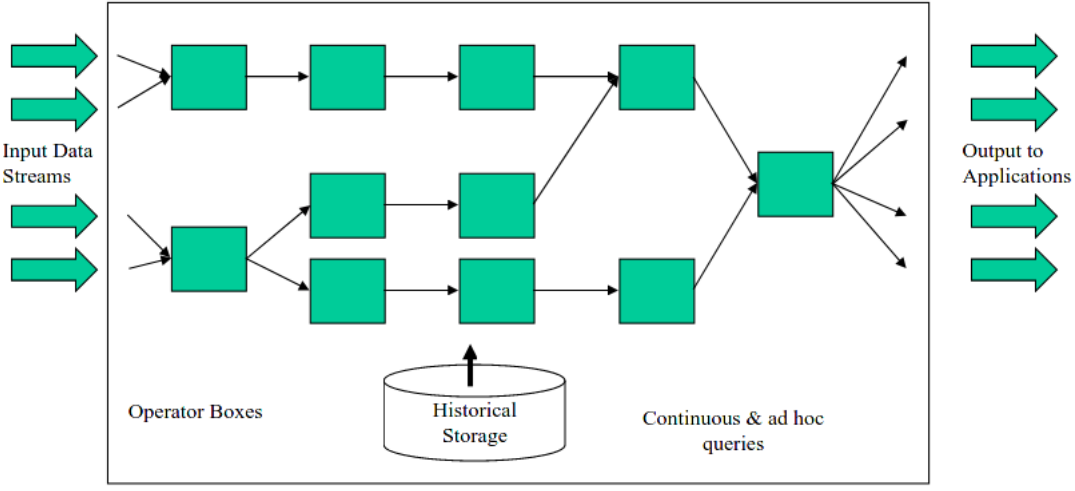
Data Stream Processing

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



Possible Optimization for Stream Query Processing

- 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 learn a model that predicts the optimal ordering, defined by a metric of choice, of operators for a given query on data streams using a deep reinforcement learning model trained on historical data?

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

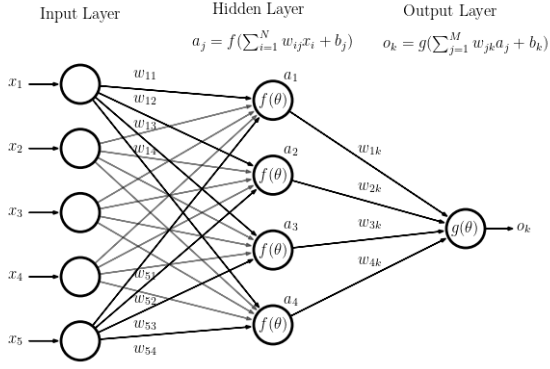


Figure: Example of a deep neural network

Reinforcement Learning

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

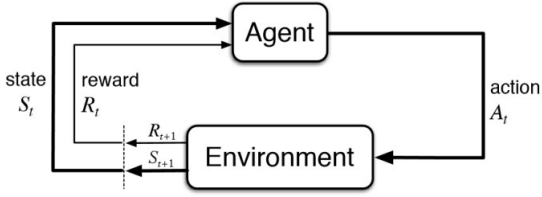


Figure: Example of a framework of a reinforcement learning agent

Optimal Policy

```
1    initialize random pi
2    while(not converged):
3        V=V(pi)
4        for s in S:
5            pi(s)=max over all actions[sum(P(s,a,s')V(S'))]
6
```

Listing 2: Policy iteration algorithm

Deep Reinforcement Learning

Deep reinforcement learning combines Deep learning and reinforcement learning

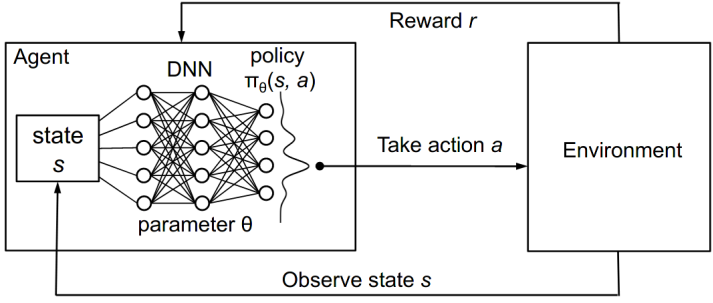


Figure: Deep Reinforcement learning(DQN) framework

Linear Road

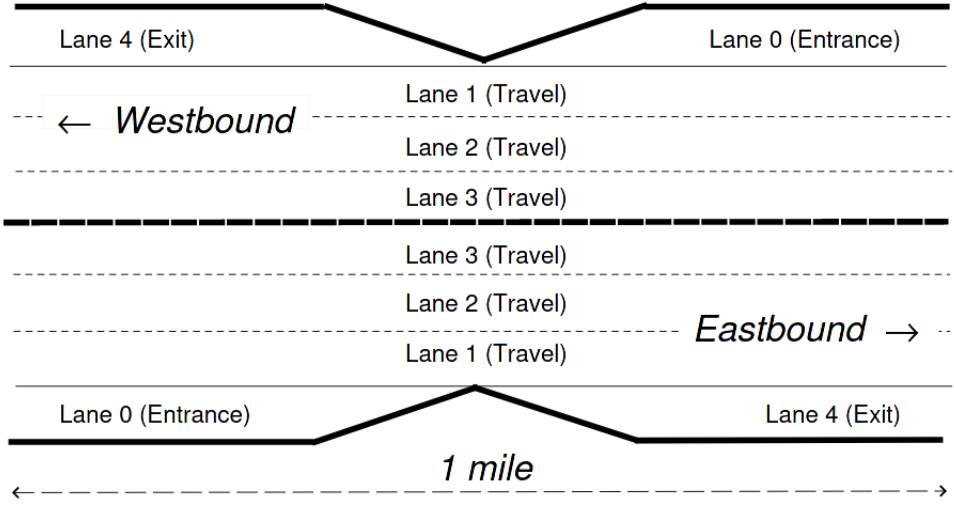


Figure: An Example Expressway Segment

Linear Road: Query to execute

```
1   SEGAVGSPEED
2       SELECT exp_way, dir, seg, AVG(speed) as speed,
3       FROM CarSegStr [RANGE 5 MINUTES]
4       GROUP BY exp_way, dir, seg;
5   SEGVOL
6       SELECT exp_way, dir, seg, COUNT(*) as volume
7       FROM CurCarSeg
8       GROUP BY exp_way, dir, seg;
9   SEGTOLL
10      SELECT S.exp_way, S.dir, S.seg, basetoll*(V.volume-150)*(V.volume-150)
11      FROM SegAvgSpeed as S, SegVol as V
12      WHERE S.exp_way = V.exp_way and S.dir = V.dir and S.seg = V.seg
13            and S.speed <= 40;
14
```

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 for SegVol and SegAvgSpeed.

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.

Model Design

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

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 3: Example of Linear Road Data

Data Distribution

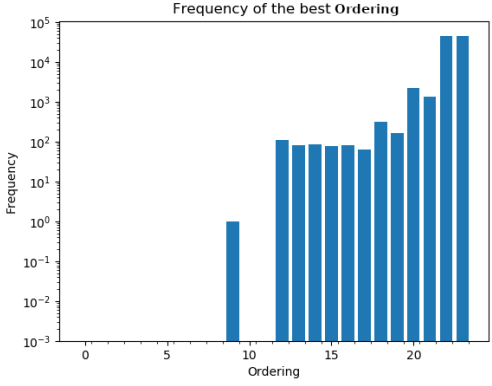


Figure: Shows for each ordering of operations, the number of data windows for which is was optimal ordering.

Data Distribution

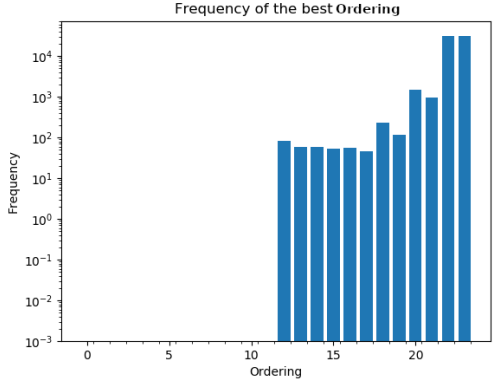


Figure: Training data. Did a 70% 30% divide on data for training and testing respectively.

Data Distribution

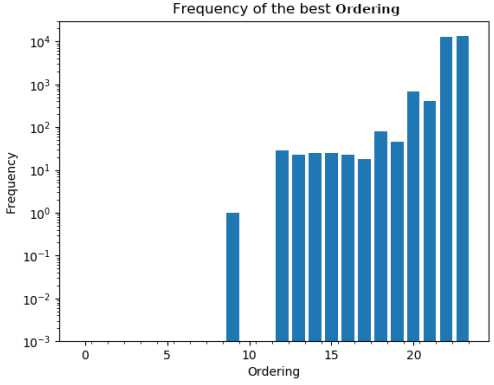


Figure: Testing data. Did a 70% 30% divide on data for training and testing respectively.

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    0    3    4   13   26]
 [   0    0    0    0    0    0    0    0    1    0   18   77  213  374]
 [   0    0    0    0    0    0    0    0    0    0   19   69  118  195]
 [   0    0    2    0    0    0    0    0    0    0  116  406 6347 6167]
 [   0    0    1    0    0    0    0    6    0  113  476 6338 6355]]
```

Listing 4: Confusion matrix for DQN classificaiton

Confusion Matrix

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

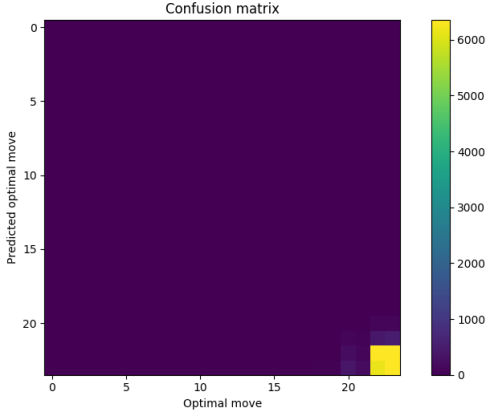


Figure: DQN predictions visualized as confusion matrix

Prediction 1

For first data window, we have the following number of operations required.

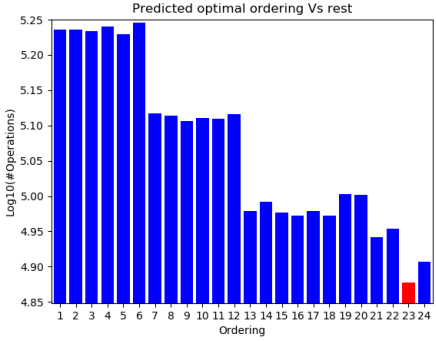


Figure: The figure shows the $\log_{10}$(number of operations) required to execute the query depending on the ordering of the selection operators chosen. The predicted optimal ordering is shown in red.

Prediction 2

For second data window, we have the following number of operations required.

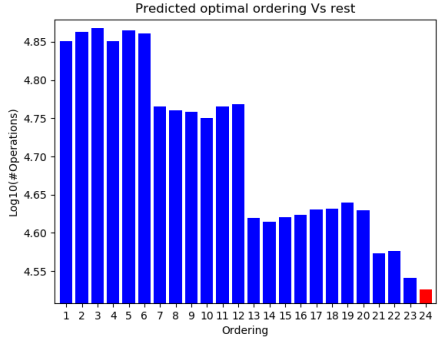


Figure: The figure shows the $\log_{10}$(number of operations) required to execute the query depending on the ordering of the selection operators chosen. The predicted optimal ordering is shown in red.

Overall accuracy

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

Comparison

If the DSMS executed

1. The TRUE optimal ordering, for each data window, on the given test data set, it will require 890M operations overall.
2. The PREDICTED optimal ordering, for each data window, on the given test data, it will required 900M operations overall.
3. The WORST ordering, for each data window, on the given test data, it will require 2310M operations overall.

Section 5

Conclusion and Future work

It is possible to use Deep Reinforcement Learning for Query Optimization in Stream Processing.
While the accuracy can be better, we have provided a proof of concept.

Further Work

- Compare this method to existing method.
- User Defined Function

Thank You

Questions?