Data Stream Query Optimization Using Deep Reinforcement Learning

Final presentation

Apoorva Tamaskar

Boston University
Metropolitan College

01 February 2021

Outline

Section 1

Background Information

Databases

How do databases store, retrieve and manipulate data?

1. Where is the data stored? Centralised, Distributed, Personal

Databases

How do databases store, retrieve and manipulate data?

1. Where is the data stored? Centralised, Distributed, Personal
2. Is the data format fixed? SQL vs NoSQL

Databases

How do databases store, retrieve and manipulate data?

1. Where is the data stored? Centralised, Distributed, Personal
2. Is the data format fixed? SQL vs NoSQL
3. What are the requirements? Scale, operations to be performed, QoS metrics to meet.

Databases

How do databases store, retrieve and manipulate data?

1. Where is the data stored? Centralised, Distributed, Personal
2. Is the data format fixed? SQL vs NoSQL
3. What are the requirements? Scale, operations to be performed, QoS metrics to meet.
4. What operations are going to be performed? E.g. Updates, Deletes, adding new fields, queries to be answered

Databases

How do databases store, retrieve and manipulate data?

1. Where is the data stored? Centralised, Distributed, Personal
2. Is the data format fixed? SQL vs NoSQL
3. What are the requirements? Scale, operations to be performed, QoS metrics to meet.
4. What operations are going to be performed? E.g. Updates, Deletes, adding new fields, queries to be answered
5. Online system or offline system? Handling static data Vs Data streams.

SQL Vs NoSQL

Some of the differences between these two are:-

1. SQL databases are table based databases whereas NoSQL databases can be document based, key-value pairs, graph databases.

SQL Vs NoSQL

Some of the differences between these two are:-

1. SQL databases are table based databases whereas NoSQL databases can be document based, key-value pairs, graph databases.
2. SQL databases are vertically scalable while NoSQL databases are horizontally scalable.

SQL Vs NoSQL

Some of the differences between these two are:-

1. SQL databases are table based databases whereas NoSQL databases can be document based, key-value pairs, graph databases.
2. SQL databases are vertically scalable while NoSQL databases are horizontally scalable.
3. SQL databases have a predefined schema whereas NoSQL databases use dynamic schema for unstructured data.

SQL Vs NoSQL

Some of the differences between these two are:-

1. SQL databases are table based databases whereas NoSQL databases can be document based, key-value pairs, graph databases.
2. SQL databases are vertically scalable while NoSQL databases are horizontally scalable.
3. SQL databases have a predefined schema whereas NoSQL databases use dynamic schema for unstructured data.
4. SQL requires specialized DB hardware for better performance while NoSQL uses commodity hardware.

SQL Vs NoSQL

Some of the differences between these two are:-

1. SQL databases are table based databases whereas NoSQL databases can be document based, key-value pairs, graph databases.
2. SQL databases are vertically scalable while NoSQL databases are horizontally scalable.
3. SQL databases have a predefined schema whereas NoSQL databases use dynamic schema for unstructured data.
4. SQL requires specialized DB hardware for better performance while NoSQL uses commodity hardware.
5. SQL is an ideal choice for the complex query intensive environment and NoSQL is a best used for solving data availability problems.

SQL

In the thesis we focus on Structured Query Language.
How does a SQL database look like?
An SQL database is a collection of tables of data.

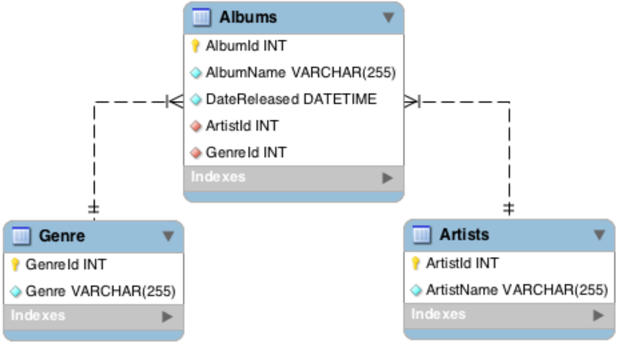

Figure: Album database

SQL

How does a table in SQL look like?
A table can be thought of a matrix, with each row representing a data point and column representing the attribute value for the data point.

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Figure: Customer Database

SQL

What is a SQL query?
An SQL query is a question or a request for answer on a database.

```
        SELECT * FROM Customers
        WHERE Country='Mexico';

```

Listing 1: SQL statement selects all the customers from the country "Mexico" in the "Customers" table

SQL

Hows is a SQL query executed?

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

```

Listing 2: SQL query to convert

SQL: Pipeline

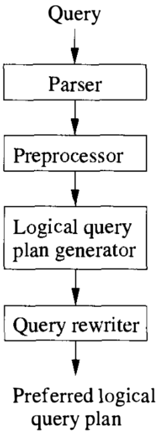

Figure: The pipeline for query processing

SQL: Parser

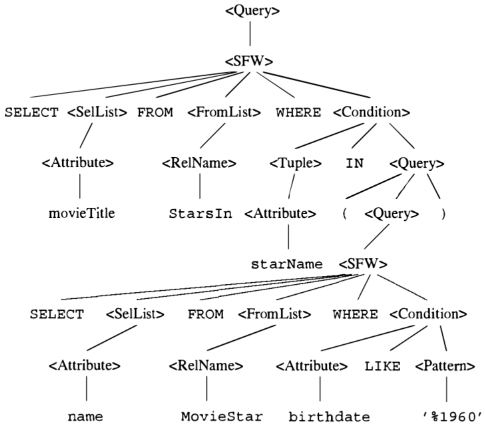

Figure: An example of parse tree

SQL: Relational Algebra: Selection

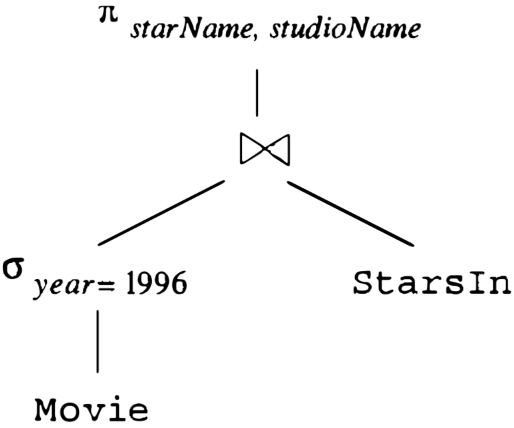

Figure: An example of selection being pushed down for optimization

SQL: Relational Algebra: Selection

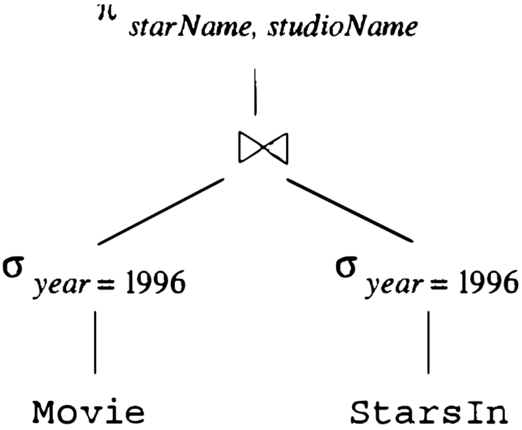

Figure: An example of selection being pushed down for optimization

SQL: Relational Algebra: Projection

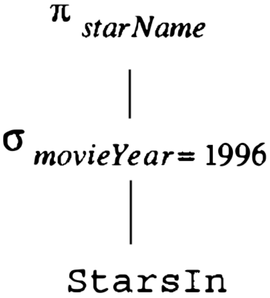

Figure: An example of projection being pushed down for optimization

SQL: Relational Algebra: Projection

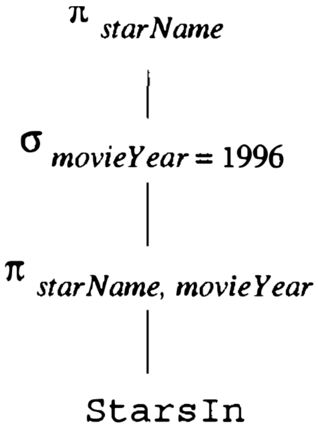

Figure: An example of projection being pushed down for optimization

SQL: Relational Algebra: Join

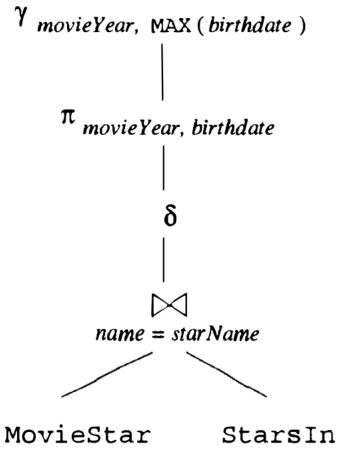

Figure: An example of join being optimized

SQL: Relational Algebra: Join

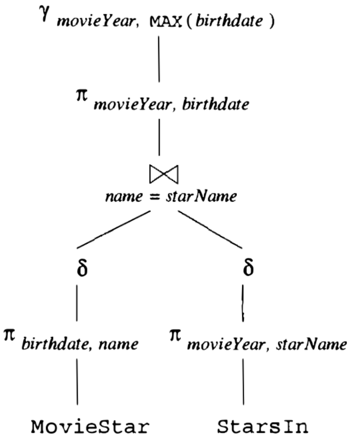

Figure: An example of join being optimized

SQL: Grouping operators

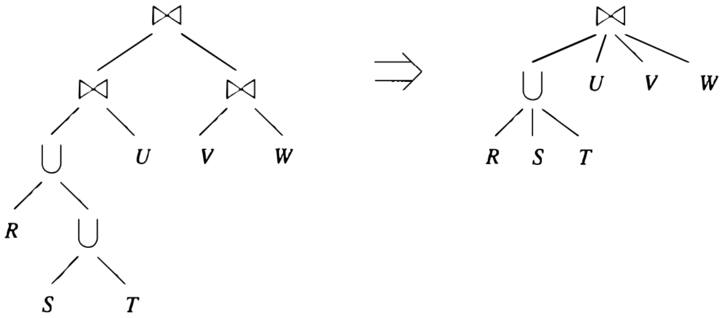

Figure: An example associative operators being grouped

SQL: Cost

$B(R) :=$ is the number of blocks needed to hold all the tuples of a relation R.
$T(R) :=$ is the number of tuples of a relation R.
$V(R, a) :=$ is the value count for an attribute a of relation R, that is, the number of distinct values relation R has in attribute a.
$V(R, [a_1, a_2, ..., a_n]) :=$ is the number of distinct values R has when all of attributes $a_1, a_2, ..., a_n$ are considered together, that is, the number of tuples in $\delta(\pi_{a_1,a_2,...,a_n}(R))$

SQL: Cost

We assume that the "cost" of evaluating an expression is approximated well by the number of disk I/O's performed. The number of disk I/O's, in turn, is influenced by:

- The particular logical operators chosen to implement the query.

SQL: Cost

We assume that the "cost" of evaluating an expression is approximated well by the number of disk I/O's performed. The number of disk I/O's, in turn, is influenced by:

- The particular logical operators chosen to implement the query.
- The sizes of intermediate results, need to pass them to the next function.

SQL: Cost

We assume that the "cost" of evaluating an expression is approximated well by the number of disk I/O's performed. The number of disk I/O's, in turn, is influenced by:

- The particular logical operators chosen to implement the query.
- The sizes of intermediate results, need to pass them to the next function.
- The physical operators used to implement logical operators, e.g., the choice of a one-pass or two-pass join, or the choice to sort or not sort a given relation.

SQL: Cost

We assume that the "cost" of evaluating an expression is approximated well by the number of disk I/O’s performed. The number of disk I/O’s, in turn, is influenced by:

- The particular logical operators chosen to implement the query.
- The sizes of intermediate results, need to pass them to the next function.
- The physical operators used to implement logical operators, e.g., the choice of a one-pass or two-pass join, or the choice to sort or not sort a given relation.
- The ordering of similar operations, especially joins.

SQL: Cost

We assume that the "cost" of evaluating an expression is approximated well by the number of disk I/O's performed. The number of disk I/O's, in turn, is influenced by:

- The particular logical operators chosen to implement the query.
- The sizes of intermediate results, need to pass them to the next function.
- The physical operators used to implement logical operators, e.g., the choice of a one-pass or two-pass join, or the choice to sort or not sort a given relation.
- The ordering of similar operations, especially joins.
- The method of passing arguments from one physical operator to the next.

SQL: Cost: Estimation

1. Give accurate estimates. No matter what method is used for executing query plans.

SQL: Cost: Estimation

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

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; that is, the size estimate for an intermediate relation should not depend on how that relation is computed. For instance, the size estimate for a join of several relations should not depend on the order in which we join the relations.

SQL: Cost: Methods

1. Histogram

SQL: Cost: Methods

1. Histogram
2. Heuristics

SQL: Cost: Methods

1. Histogram
2. Heuristics
3. Top-Down

SQL: Cost: Methods

1. Histogram
2. Heuristics
3. Top-Down
4. Bottom-up

SQL: Cost: Methods

1. Histogram
2. Heuristics
3. Top-Down
4. Bottom-up
5. Dynamic Programming

SQL: Cost: Methods

1. Histogram
2. Heuristics
3. Top-Down
4. Bottom-up
5. Dynamic Programming
6. Branch-and-Bound

SQL: Cost: Methods

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

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: Joins

1. Nested-loop join

SQL: Joins

1. Nested-loop join
2. Index join

SQL: Joins

1. Nested-loop join
2. Index join
3. Dynamic Programming

SQL: Joins

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

SQL: Physical Plan

1. Selection, Index based

SQL: Physical Plan

1. Selection, Index based
2. Selection, Table Scan

SQL: Physical Plan

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

SQL: Physical Plan

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

SQL: Physical Plan

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

Data Streams

Stream query optimization is the process of modifying a stream processing query, often by changing its graph topology and or operators, with the aim of achieving better performance (such as higher throughput, lower latency, or reduced resource usage), while preserving the semantics of the original query.

Stream query optimizations are best understood with respect to stream graphs. A stream graph is a directed graph whose edges are streams and whose nodes are operators. Root and leaf nodes are called sources and sinks, respectively.

Possible Optimizations

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

We focus on Operator Reordering.

Deep Neural Networks

To understand a neural network we should first look at a neuron.

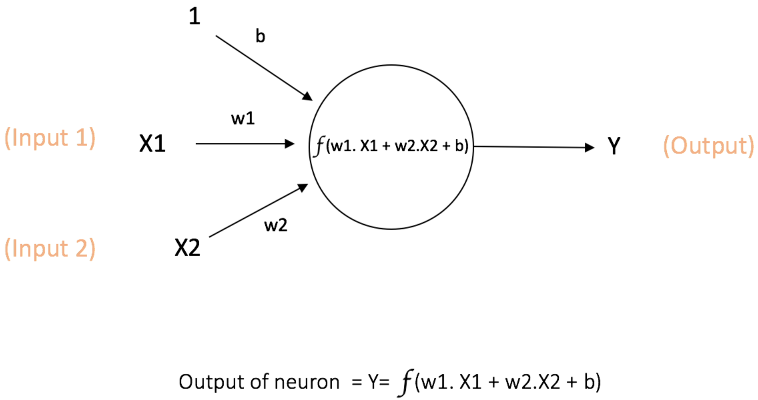

Figure: Example of a neuron

f is generally taken to be a non linear function. This non linearity grants neural networks additional flexibility.

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

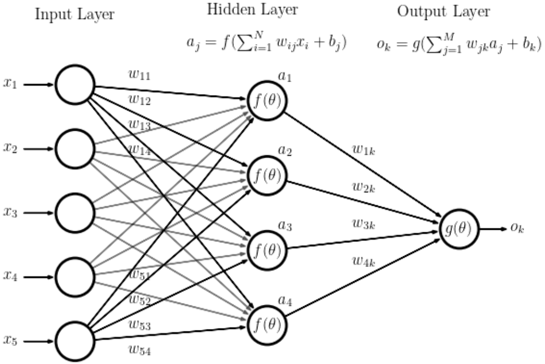

Figure: Example of a deep neural network

Backpropogation

How does a neural network train on these parameters? First look at how a single node back propogates.

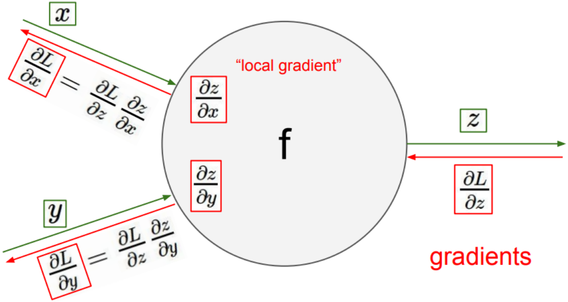

Figure: Example of backpropogation on a node

Backpropogation

By doing backpropogation on each node, we finish the process.

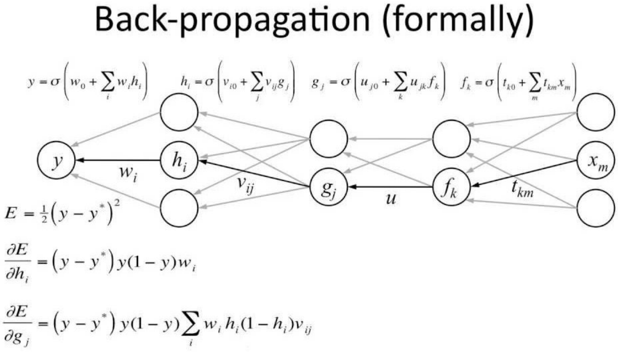

Figure: Example of backpropogation

Reinforcement Learning

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

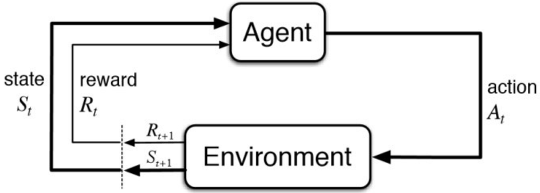

Figure: Example of a framework of a reinforcement learning agent

Value Iteration

```
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 3: value iteration algorithm

Policy Iteration

```
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 4: Policy iteration algorithm

Deep Reinforcement Learning

Deep reinforcement learning combines Deep learning and reinforcement learning

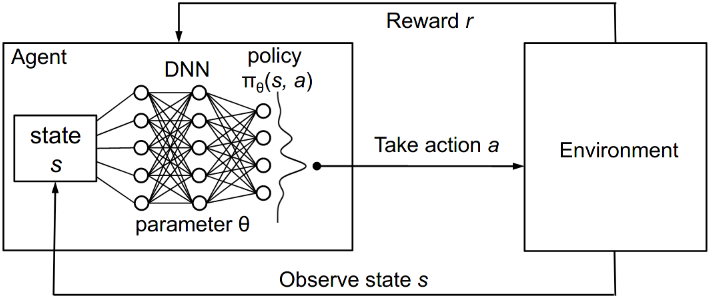

Figure: Deep Reinforcement learning(DQN) framework

Section 2

Linear Road, implementation and Justification

Linear Road

Linear Road is inspired by the increasing prevalence of variable tolling on highway systems in cities throughout the world. Linear Road specifies a variable tolling system for a fictional urban expressway system where tolls are determined based on changing factors such as congestion and accident proximity. Each car on the expressway is equipped with a transponder or sensor that emits a position report that identifies the vehicle's exact location (coordinates) every 30 seconds. These position reports are used to generate statistics about traffic conditions on every segment of every expressway for every minute.

Linear Road

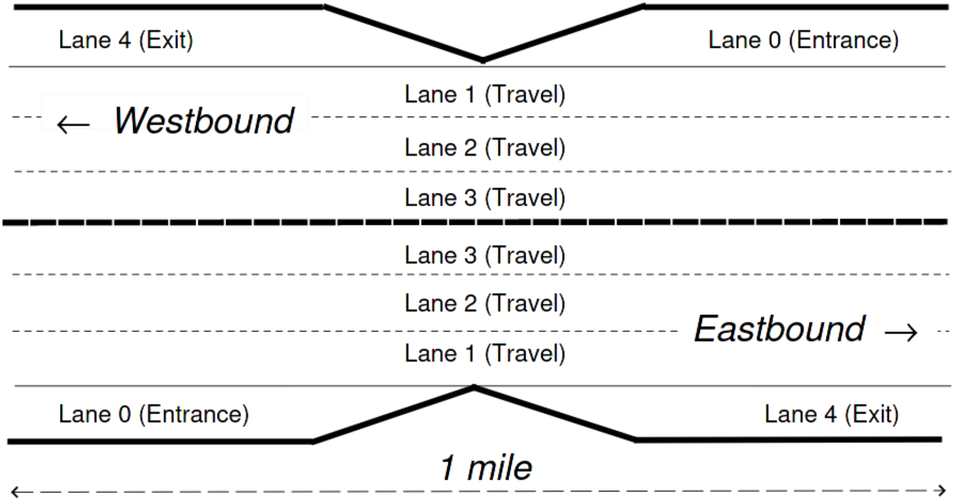

Figure: An Example Expressway Segment

Queries

The query to execute is *SegToll*, but in order to calculate it, we need to calculate various other relations first.

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

Listing 5: CurCarSeg linear road query

Queries

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

```

Listing 6: SEGAVGSPEED linear road query

Queries

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

Listing 7: SEGVOL linear road query

Queries

```
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 8: SEGTOLL linear road query

Implementation: Data Generator

Used the walmart Linear road data generator.

Implementation: Query Execution

Mimicked the execution of query in C++.
Given that there are 24 possible orderings, executed all of them and recorded how much time and the number of operations they took.
For each window of data extracted the entropy of columns, size of tables and stored these too.

Implementation: DQN

The DQN took in input the column wise entropy, the size of tables and the ordering of operations, the reward for the training is taken to be the number of operations required.
For prediction purposes
For each entropy vector, we calculate the reward for each ordering.
We check which ordering has the least numbre of operations required. and determine the category.

Justification

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

- To achieve a value as close as possible to the global minima for the optimization function (Adam optimzer)

Justification

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

- To achieve a value as close as possible to the global minima for the optimization function (Adam optimzer)
- The time required to predict the optimal move should not exceed the time saved by using it.

Justification

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

- To achieve a value as close as possible to the global minima for the optimization function (Adam optimzer)
- The time required to predict the optimal move should not exceed the time saved by using it.
- The time and resources required for training should not exceed the capacity of the system while it is running the query processing in the background.

Justification

What we found was:-

- Adding additional layers improves the prediction of the optimal moves but not by significant margin.

But note, these 2 are only query and data specific findings.

Justification

What we found was:-

- Adding additional layers improves the prediction of the optimal moves but not by significant margin.
- Adding additional layers resulted in the time spent predicting the answer overshadowing the time saved by executing the optimal move.

But note, these 2 are only query and data specific findings.

Section 3

Results, Conclusion and Further Work

Data Distribution

The data obtained by executing all the orderings for the query plan on Linear road data, resulted in the following frequencies for optimal orderings.

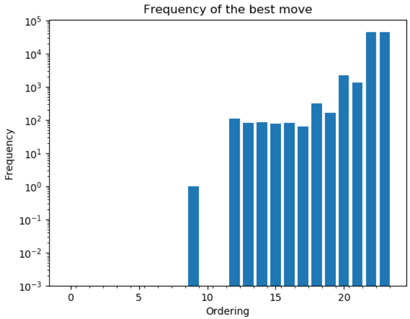

Figure: This figure shows the frequency distribution of the number of cases where each move is optimal

Data Distribution

While training and testing the DQN model, we divided the total data into a 70 : 30 for training and testing purposes respectively.

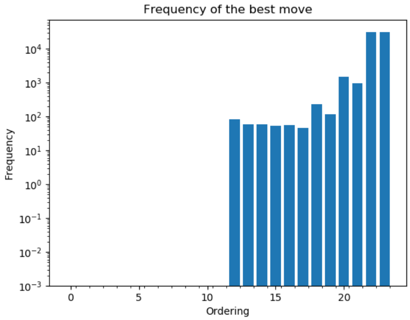

Figure: This figure shows the frequency distribution of cases where each move is optimal in the training dataset

Data Distribution

While training and testing the DQN model, we divided the total data into a 70 : 30 for training and testing purposes respectively.

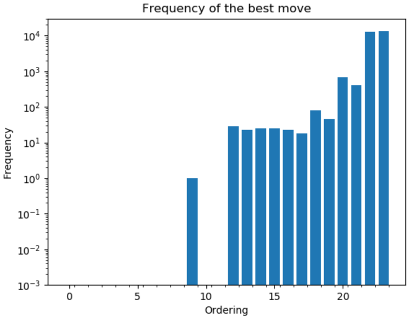

Figure: This figure shows the frequency distribution of cases where each move is optimal in the testing dataset

Confusion Matrix

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

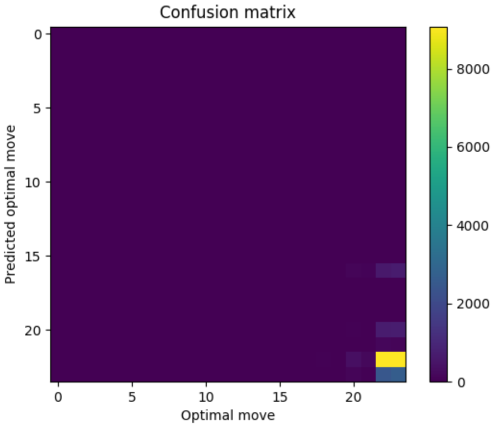

Figure: DQN Run 1 confusion matrix

Confusion Matrix

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

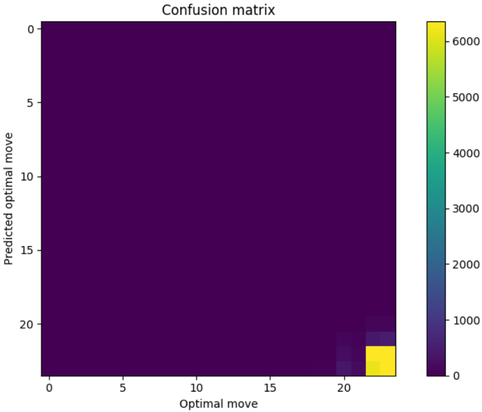

Figure: DQN Run 2 confusion matrix

Predictions

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

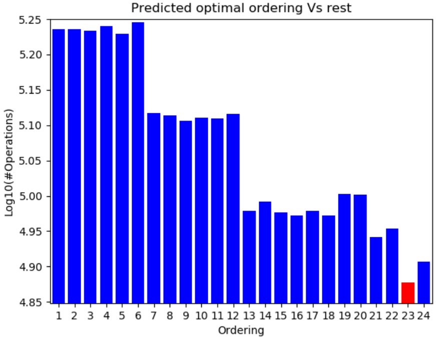

Figure: The figure shows the $\log_{10}$(number of operations) required to execute the query depending on the ordering of the selection operators chosen. The predicted optimal ordering is shown in red.

Predictions

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

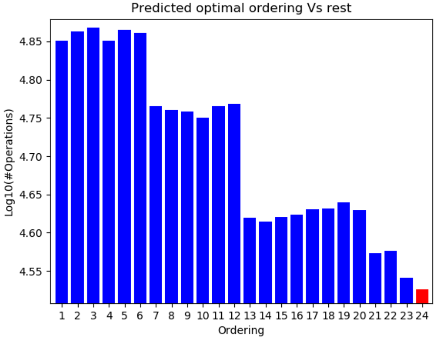

Figure: The figure shows the $\log_{10}$(number of operations) required to execute the query depending on the ordering of the selection operators chosen. The predicted optimal ordering is shown in red.

Overall Performance

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

Overall Performance

1. The sum of operations required by the optimal ordering 890640075.0

Overall Performance

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

Overall Performance

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

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

- The neural network is unable to achieve low loss for the test data, this perhaps speak the lack of features extracted during the querying time.

Further Work

- The neural network is unable to achieve low loss for the test data, this perhaps speak the lack of features extracted during the querying time.
- The data sample we have is highly biased, leading to rather baised DQNs.

Further Work

- The neural network is unable to achieve low loss for the test data, this perhaps speak the lack of features extracted during the querying time.
- The data sample we have is highly biased, leading to rather baised DQNs.
- The entire method needs to be parallelized and set up on a scalable infrastructure to see the actual effects of DQN.

Further Work

- The neural network is unable to achieve low loss for the test data, this perhaps speak the lack of features extracted during the querying time.
- The data sample we have is highly biased, leading to rather baised DQNs.
- The entire method needs to be parallelized and set up on a scalable infrastructure to see the actual effects of DQN.
- The DQN used is simulating a single move game rather than a multi move game.

Further Work

- The neural network is unable to achieve low loss for the test data, this perhaps speak the lack of features extracted during the querying time.
- The data sample we have is highly biased, leading to rather baised DQNs.
- The entire method needs to be parallelized and set up on a scalable infrastructure to see the actual effects of DQN.
- The DQN used is simulating a single move game rather than a multi move game.
- Not looked at how DQNs can be trained online.

Questions?