Adding Robustness to HYDRA

Handling unseen queries

By Kapil Khurana & Vishal Goel

M.Tech (Coursework) CSA

IISc Bangalore

Problem Statement

How HYDRA works?

- INPUT : AQPs from client
- OUTPUT : Synthetic data to test vendor's database engine on
- ISSUE : Because HYDRA uses only linear programming to learn the properties of the data, it works well
  - only on AQPs similar to the ones in the input.
  - for small number of AQPs (i.e. unscalable)
- Basically HYDRA is not smart (yet!)
- ***So the task is to integrate a learning based cardinality estimation model into HYDRA so that the summary generated is consistent with the model (can handle unseen queries as well).***

THE PLAN

Machine learning to the rescue:

Input : AQPs

Output : A model (that has learnt client’s database properties) to be integrated with HYDRA to make it smart and robust

The Neural Network Approach to Cardinality Estimation

- Data Distribution is complex. (It's not linear , or quadratic or cubic or exponential and one can add many more functions…..)
- Neural Networks can have a large number of free parameters (the weights and biases between interconnected units) and this gives them the flexibility to fit highly complex data (when trained correctly) that other models are too simple to fit.
- It learns all by itself, just like a brain! The neural network can learn its own parameters.
- Complexity increases with the data distribution complexity and the volume of the data.
- Selection of activation, regularization and error function is difficult to decide.

The Neural Network Approach to Cardinality Estimation

Training Data

DATABASE : IMDB job set

Query generation with output cardinalities for each query

Note : Currently, only filter predicates on a single table considered. For join predicates, the neural network can be made multi-level, essentially capturing inter-table correlations.

The Neural Network Approach to Cardinality Estimation

Neural Network Architecture

- Model built on a single table
- One input layer, one hidden layer, one output layer
- INPUT LAYER : Two nodes for each attribute, one for <= and one for >=

(a,b) ≡ [a+1, <=b-1] >a ≡ [a+1, attr_max_val]

<b ≡ [attr_min_val, b-1] =a ≡ [a, a] !=a ≡ total_card - [a,a]

Model assumes AND operations only. To handle OR, we can use DeMorgan's Law. To handle NOT, we can compute complement

The Neural Network Approach to Cardinality Estimation

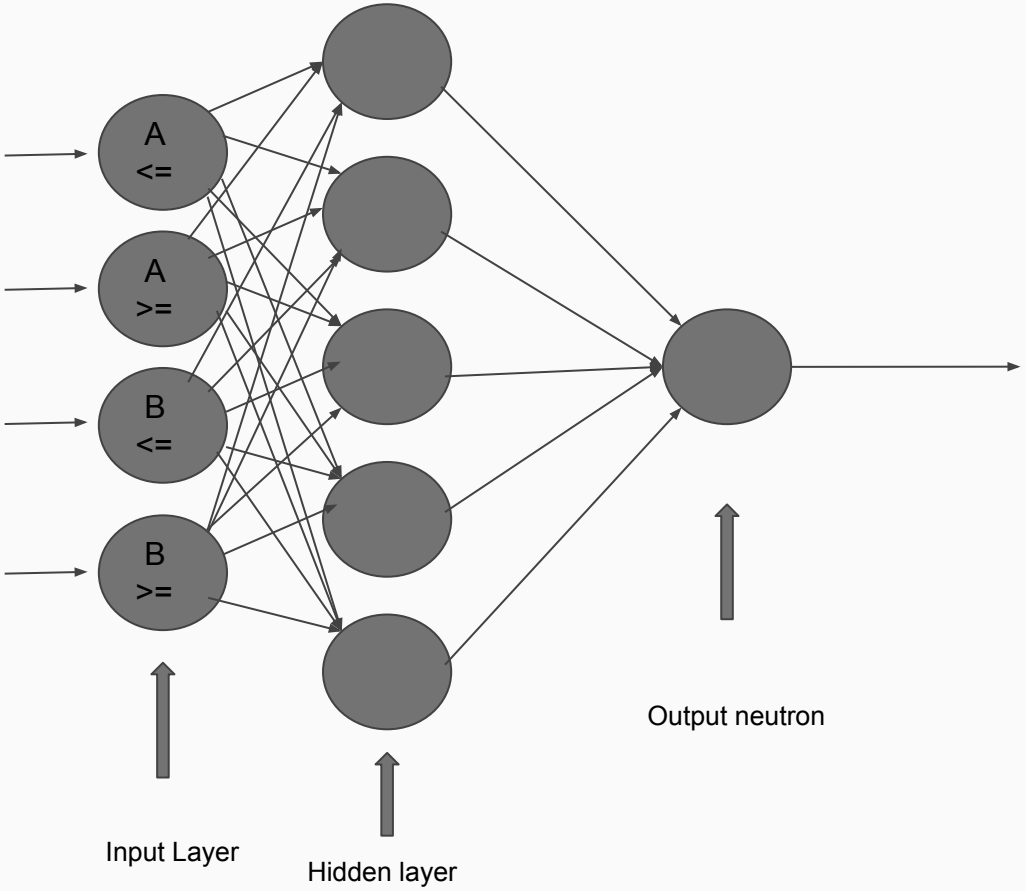
Neural Network Architecture

- HIDDEN LAYER : Number of nodes learned while running diagnostic tests
- OUTPUT LAYER : Only one node - representing result cardinality of the query.
- Sigmoid function used as activation function.

Toy Example

Schema R(A,B)

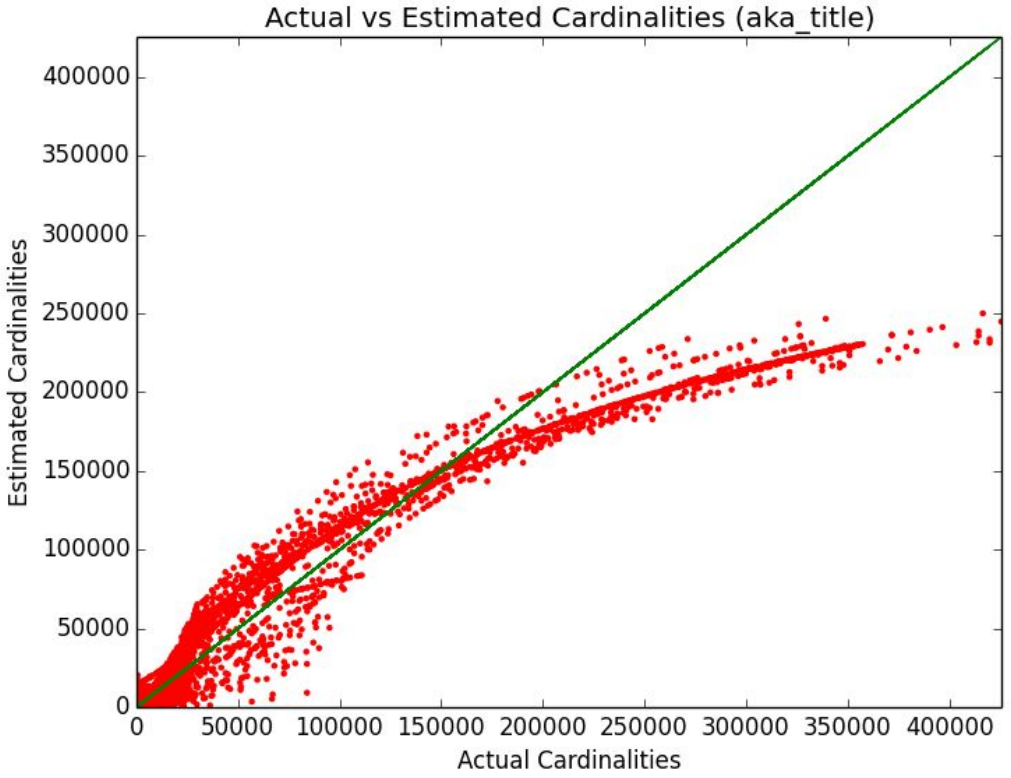
- Normalized Input (using max and min value of attribute)
- Output is compared to normalized target cardinalities.
- Normalization function is invertible so we can get actual cardinalities from the output of neural network.



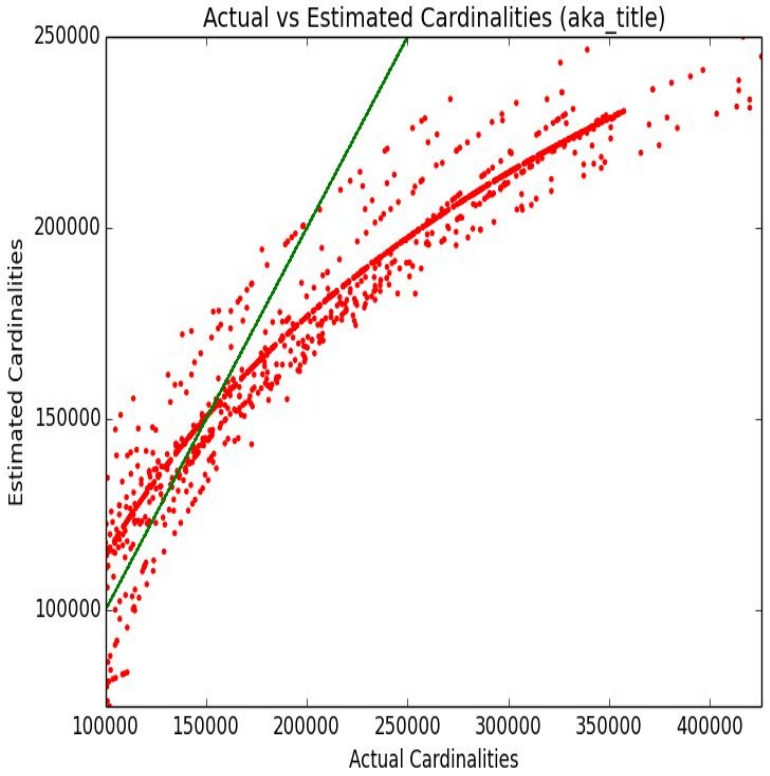
Results

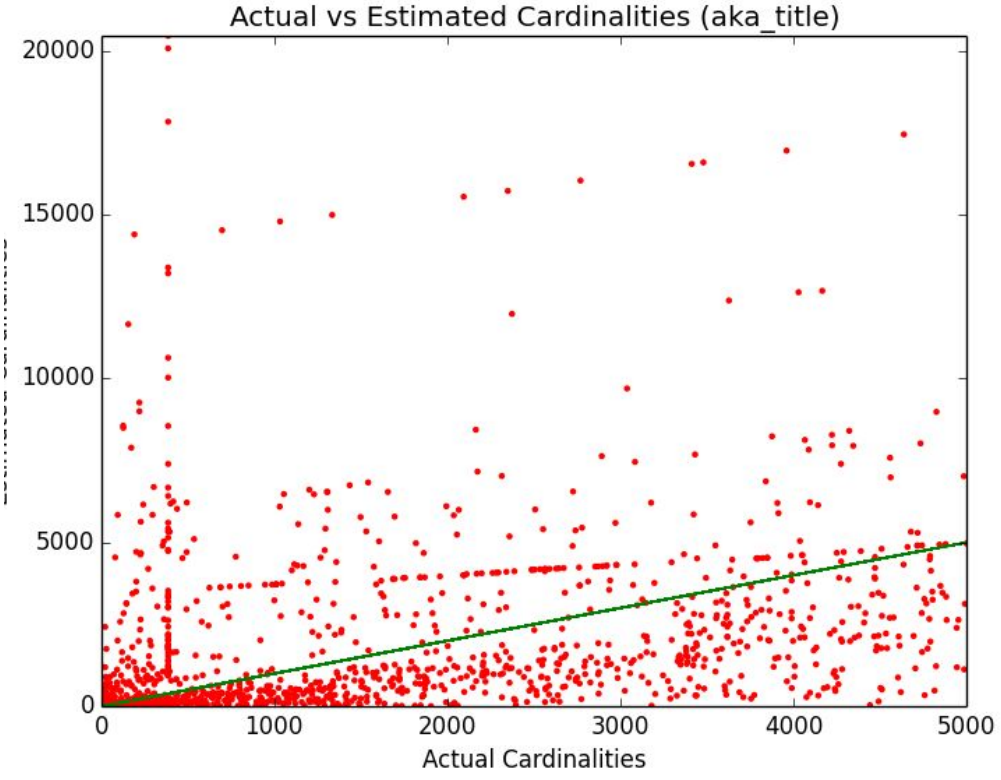
- Density(number of queries) much higher at lower cardinalities.
- Same thing was observed in the training set
- Lack of queries with high output cardinality in training set resulted in performance degradation.

Conclusion regarding training set - Need queries with uniformly distributed output cardinalities.



Results


Actual vs Estimated Cardinalities (aka_title)
Estimated Cardinalities
Actual Cardinalities
250000
200000
150000
100000
100000
150000
200000
250000
300000
350000
400000


Actual vs Estimated Cardinalities (aka_title)
Actual Cardinalities
20000
15000
10000
5000
0
0
1000
2000
3000
4000
5000

Future Plan

1. To make neural network better and faster
2. To integrate the selectivity estimation into HYDRA such that the synthetic data is consistent with the model
3. To evaluate the algorithm on the TPC-DS benchmark database and evaluate if the generated data can satisfy unseen AQPs also

Thanks!

Questions, Comments and Suggestions are appreciated!