Adding Robustness to Hydra

Kapil Khurana
kapilkhurana@iisc.com

Vishal Goel
vishalgoel@iisc.com

December 8, 2018

1 Introduction

HYDRA aims to generate synthetic data whose behaviour is volumetrically similar to the client database on the pre-specified query workload.[1] That is, assuming a common choice of query execution plans at the client and vendor sites, the output row cardinalities of individual operators in these plans are very similar in the original and synthetic databases. Hydra incorporates a novel LP formulation technique, region-partitioning, that can encode volumetric constraints with an LP of low complexity. Currently, data generated through Hydra can satisfy only the annotated query plans (AQPs) that were given as input to the data generation algorithm. For an unseen query, the AQP obtained from Hydra's (output) synthetic database can be extremely different from the AQP obtained from original database. In this project we aim to add robustness to the data generation algorithm. The goals of the project are to:

- Implement a contemporary learning-based selectivity estimation algorithm that takes AQPs as the input.
- Propose a way for integrating the learning-based algorithm into the LP solution of Hydra such that the synthetic data is consistent with the selectivity estimation model generated in first part.

1.1 Shortcomings in HYDRA

Inconsistent inter-region data distribution In region-partitioning approach of HYDRA, variables represent non-overlapping distinct regions. Given cardinality constraints (CCs), the LP formulator constructs an LP and the solver makes sure that the equations corresponding to cardinality constraints are satisfied.

However, LP solver may give any feasible solution for the LP which might be inconsistent with the client database, resulting into inconsistent inter-region data distribution. This means that the cardinality of each region might be different from the actual cardinality of that region, while still satisfying the CCs. Hence, the LP solution given by HYDRA may not satisfy CCs from unseen queries.

Inconsistent intra-region data distribution During database summary generation, HYDRA does not follow any data distribution technique inside each region because it has no information about data distribution pattern inside regions resulting into inconsistent intra-region data distribution. Thus, only queries having AQPs similar to the ones which were given as input to HYDRA will be satisfied. So even if LP solver outputs correct cardinality for each region, the queries corresponding to unseen AQPs will not be satisfied on the synthetic data generated by HYDRA.

1.2 Assumptions

Currently, we have worked on a single table with integral domains. The correlations between multiple tables and join still remain unexplored for now. However, the neural network can be made multi-level, essentially capturing inter-table correlations.

2 Smart-HYDRA Architecture

Figure 1 shows the current HYDRA architecture with the following additional components (present in red coloured boxes):

2.1 Client Site

A learning based cardinality estimation algorithm that queries the database engine to get a set of predicates with output cardinalities and generates the cardinality estimation model $\mathcal{M}$ of the database. This model is independently generated at the client site and is passed to the vendor site.

2.2 Vendor Site

- Using model $\mathcal{M}$ with current HYDRA codebase, we generate an objective function and pass it to the LP solver that ensures that the values of the variables

(cardinalities of the respective regions) follow our model $\mathcal{M}$, hence making inter-region distribution consistent with the model.

- Integrating intra-region data distribution model into summary generation so that the database summary generated for each region follows the same distribution as per the corresponding selectivity in the model $\mathcal{M}$, hence making intra-region distribution consistent with the model.

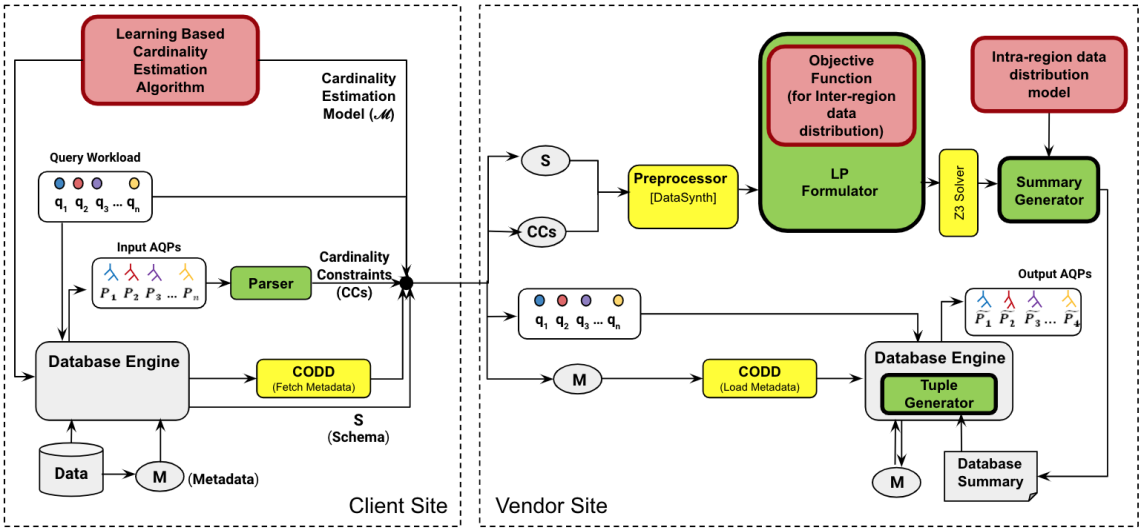

Figure 1: Smart Hydra Architecture

3 Cardinality Estimation Model

Data Distribution in database management systems, as we know, can be very complex. However, deep learning networks could be used to approximate such complex distributions using a large number of learning parameters[2]. During training, the behaviour of the inner layers is not defined by the input data. Instead, these models learn how to use the layers to produce the correct output. Since there is no interaction between the layers and the input training data, these layers are called hidden layers. Given AQPs, our aim is to apply neural network technique to learn the parameters which, when supplied with a new AQPs, can predict the cardinality.

3.1 Neural Network Architecture

Each node in the input layer represents a filter predicate on a single attribute. It is quite straightforward to reduce the set of operators from $\{<, <=, >, >=, \neq\}$ to only $\{<=, >=\}$. This is because the operators in the former set can be easily derived from the latter set in the following way:

$$
\begin{aligned}
(a, b) &\equiv [a + 1, <= b - 1] \\
> a &\equiv [a + 1, max] \\
< b &\equiv [min, b - 1] \\
= a &\equiv [a, a] \\
\neq a &\equiv |R| - [a, a]
\end{aligned}
$$

where *max* and *min* denote the maximum and minimum values of the attribute respectively on which the filter is applied and $|R|$ denotes the relation cardinality.

Thus, the input queries in the training set only include <= and >= operators as part of the filter predicates. Also, we only consider AND operator between filter predicates in the AQPs. The queries involving OR operator can be transformed into the one with combinations of AND and NOT operator using DeMorgan's law. Other complex operators like aggregates remain unexplored.

Now, the architecture can be summarised in the following points:

- One input layer which contains two nodes(one for <= and one for >=) for each attribute.
- One output layer which contains one node that represents result cardinality.
- One hidden layer the number of nodes in which are decided after running diagnostic tests. For our assumptions, we require around five nodes.
- The metadata used includes the maximum and minimum values of each attribute and the total table cardinality.

Note that the architecture of neural network is valid for one relation and independent of the number of attributes in the relation. An example neural network for

relation R(A,B) with 5 hidden layer nodes and 1 output layer node is shown in figure 2.

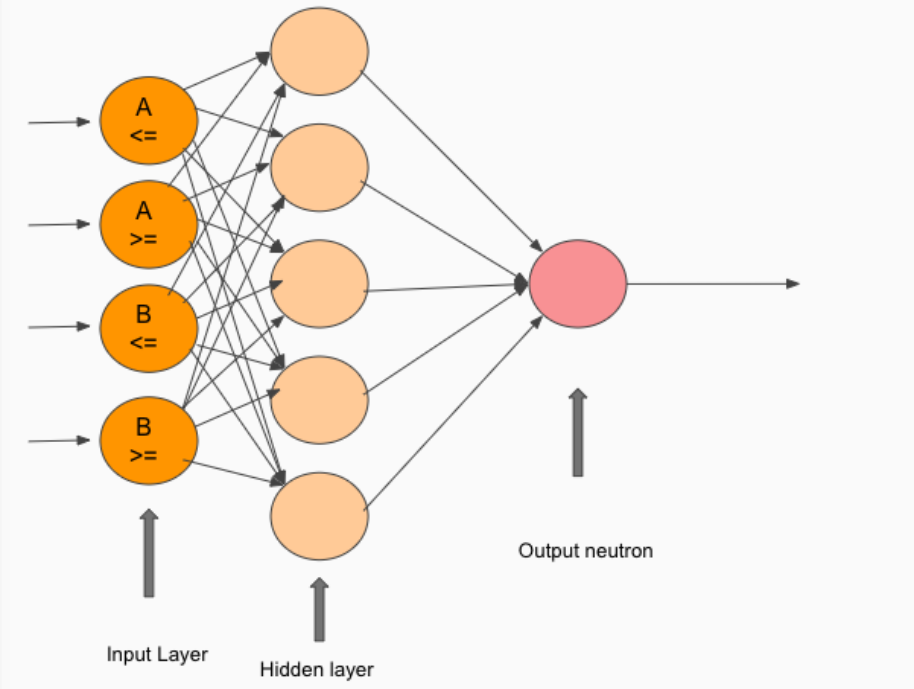

Figure 2: Neural network Architecture

3.2 Experiment

The Join Order Benchmark dataset has been used as the training set for the Model $\mathcal{M}$. We took a training size of 20k queries on a single table *aka_title* with filter predicates on four attributes *id, kind_id, movie_id, production_year*) where 70% of the queries have been used for training the model and the rest for testing. After running diagnostic tests, the optimal hidden layer size was found to be 11 nodes and regularisation parameter value was found to be 0. In the figure 3, the X-axis represents the actual cardinalities and the Y-axis represents the cardinalities estimated by our model. The plot obtained is as shown in the figure below:

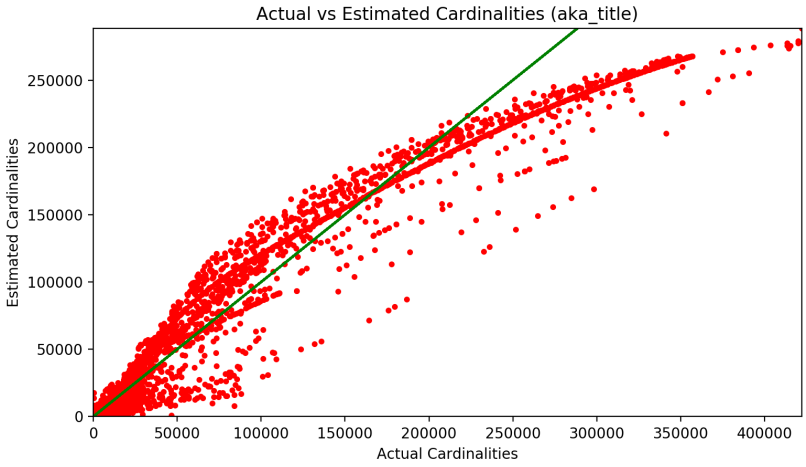

Figure 3: Cardinality Plot

4 Inter-Region Data Distribution Model

To ensure consistency of inter-region data distribution, we integrate an objective function into the LP formulator. To explain this, let's take an example. Consider a relation with two attributes A and B and let there be two filter constraints applied on them. Let us represent one constraint as shown by the red box (A in [20, 40] and B in [15K, 50K]) below and likewise let the other constraint be as shown by the green box. Let the number of tuples satisfying the red constraint be 1000 and let the number of tuples satisfying the green constraint be 400. In addition let the total number of tuples be 5000. We can represent these constraints using the following linear equations:

$$\begin{aligned} x_1 + x_2 &= 400 \\ x_2 + x_3 &= 1000 \\ x_1 + x_2 + x_3 + x_4 &= 5000 \end{aligned}$$

where x_1, x_2, x_3 and x_4 are four variables representing the number of tuples that lie in the respective regions as shown in the figure below. In addition, we add constraints $x_1, x_2, x_3, x_4 \geq 0$. Note that any database that is derived from this LP will satisfy the three constraints discussed above.

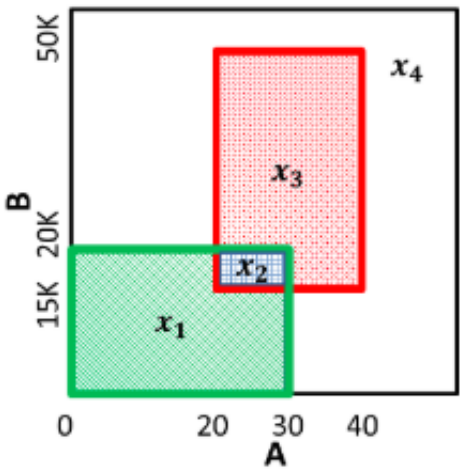

Figure 4: LP region partitioning

For each variable, HYDRA currently stores a region in the form of filters applicable to each dimension, which is two in our example . We extract out filter predicates for each region and build a respective query for which we find out the cardinality using our model $\mathcal{M}$. Suppose we get cardinality estimates a_1, a_2, a_3 and a_4 for x_1, x_2, x_3 and x_4 respectively. Then, we can apply any error minimisation function to minimise the difference between values of the LP variables and the corresponding cardinality estimates. In our project, we used absolute minimum error objective function. Thus, the following objective function will be used for our example in consideration:

$$minimize\ (|x_1 - a_1| + |x_2 - a_2| + |x_3 - a_3| + |x_4 - a_4|)$$

5 Intra-Region Data Distribution Model

5.1 Summary Construction

We propose the following way to generate database summary.

- For each interval for an attribute (X) in each region, we will first fit a Cumulative Density Function (CDF) i.e. Pr ($X \leq x$) using our cardinality estimation model $\mathcal{M}$. The function (say $\mathcal{F}$) is chosen to be an invertible increasing function having range [0,1].

- While creating the database summary, we store the set of CDFs (one for each attribute) corresponding to each region.

This summary would try to ensure that the intra-region data distribution is consistent with our model $\mathcal{M}$.

5.2 Tuple Generation

Say there exists an n-dimensional region for which we have to generate k tuples. Now, we will generate k random sample points from a uniform [0,1] - n dimensional distribution. For each sample point $(u_1, u_2, ..., u_n)$, we compute the following tuple $(\mathcal{F}^{-1}(u_1), \mathcal{F}^{-1}(u_2), ..., \mathcal{F}^{-1}(u_n))$.

6 Conclusion

We have identified shortcomings (inconsistency with actual database in inter and intra-region data distributions) with hydra and suggested integrating a learning based cardinality model with various existing components in HYDRA. We have implemented a neural network based cardinality estimation model. We propose to add an objective function(calculated using current HYDRA codebase and our model) to the Linear Program to handle inter-region data distribution inconsistency. We propose fitting invertible, monotonically increasing Cumulative Density Function for each attribute interval in a region to handle intra-region data distribution inconsistency.

References

[1] A. Sanghi, R. Sood, J. Haritsa, S. Tirthapura. Scalable and Dynamic Regeneration of Big Data Volumes International Conference on Extending Database Technology(EDBT), March,2018

[2] J. Ortiz, M. Balazinska, J. Gehrke, S.S. Keerthi Learning State Representations for Query Optimization with Deep Reinforcement Learning International Workshop on Data Management for End-to-End Machine Learning, June, 2018