Approximate Query Processing Using Deep Learning and Database Techniques

by

SHOHEDUL HASAN

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2022

To my Ammu, Hasina Akter, and my Abbu, Abul Kalam, who made me the person I am today.

ACKNOWLEDGEMENTS

First and foremost, thanks to the Almighty!

I want to thank my supervising professor Dr. Gautam Das for his continuous support, guidance, and encouragement throughout my Ph.D. When I joined DBXLab 5 years ago, I barely knew what research and a Ph.D. were. I wanted to work with a good professor on a good topic. Dr. Das allowed me to grow at my own pace while he always emphasized the highest quality and value of work. Initially, he appointed me as a teaching assistant in his course to quickly improve my communication and public speaking skills. He taught me how to think beyond the mean. In my journey to be an independent researcher, Dr. Das helped me in every step. While research and work are part of any Ph.D. process, he provided his highest support during my health crisis. I am incredibly fortunate to have Dr. Das as my supervisor in my Ph.D.

I want to thank my professors in my Ph.D. committee, Dr. Vassilis Athitsos, Dr. Leonidas Fegaras, and Dr. Mohammad Atiqul Islam. They have always extended their support whenever I needed it. I am extremely grateful to my collaborators, Dr. Saravanan Thirumuruganathan, Dr. Nick Koudas, and Dr. Abolfazl Asudeh, for their continuous guidance in different phases of my Ph.D. I have been fortunate to get friends like Suraj Shetiya, Tanusree Debi, Dr. Jees Augustine, Dr. Sona Hasani, and Dr. Md Abdus Salam in DBXLab. I wish to thank all my friends who supported me with their help, blessing, and kind words. I would specially thank my school teacher Ohiduzzaman Sir. His inspiration transformed my thinking. I am thankful to all my teachers in all phases of my life.

Finally, I would like to thank my family for their sacrifices. My parents managed to bring a kid from a remote village in Bangladesh to my current position with their countless

sacrifices despite limited resources. My deepest gratitude to my lovely wife, Kazi Samina Maraj Mumu, who supported me in all the ups and downs of my Ph.D. journey.

November 9, 2022

ABSTRACT

Approximate Query Processing Using Deep Learning and Database Techniques

Shohedul Hasan, Ph.D.

The University of Texas at Arlington, 2022

Supervising Professor: Gautam Das

Data is generated at an unprecedented rate surpassing our ability to analyze them. In real applications, it is often impractical to find an exact answer by traversing the entire data. As a result, Approximate Query Processing (AQP) is getting extremely popular, which finds an approximate answer in a quick time by sacrificing a fraction of accuracy. This dissertation focuses on developing different AQP techniques to solve fundamental database problems using deep learning. Moreover, we build a fast and scalable algorithm for Quantile Regression, a well-known regression technique that can help minimize the uncertainty in the recent deep learning-based AQP solutions, including ours.

First, we develop solutions for a fundamental database problem called Selectivity Estimation- the problem of estimating the result size of queries. Poor selectivity estimation can seriously impair query optimizer, hence an inefficient query execution. The traditional database techniques often perform poorly for multi-attribute queries involving many attributes. The deep learning-based solution is a promising area of research for Selectivity Estimation. However, many existing approaches suffer from high inaccuracy for low-selectivity queries. We propose two deep learning-based solutions that perform exceptionally well for low-selectivity multi-attribute queries involving many predicates. Our first

approach treats selectivity estimation as a density estimation problem and learns the distribution using an unsupervised deep learning technique from data. The second approach is a supervised technique that learns from queries. Both of our approaches are fast and accurate, even for a large number of predicates.

The second problem we tackle in this dissertation is one of the most important traditional database problems widely known as Approximate Query Processing (AQP) for aggregate queries like Sum, Average, and Count, which a data scientist widely uses for real-time data analysis. We propose a deep learning-based technique that learns the data distribution to generate samples representative of the input data from the model. Once we have a learned model, samples can be generated as needed, and aggregate queries can be answered without affecting the database. Moreover, the sample quality can be improved by our proposed techniques for minimizing model bias and ensemble approaches.

Finally, we propose a scalable algorithm for a famous regression problem, Quantile Regression, which can help minimize the uncertainty of deep learning-based solutions, including our proposed solutions. We use computational geometry concepts of arrangement and duality to design an algorithm to calculate objective cost from a neighboring point in the arrangement in $O(d)$. Moreover, we propose an algorithm for 2-dimensional Quantile Regression, which is better than any other existing techniques.

While our deep learning-based approaches' efficacy is demonstrated with extensive experiment results, our Quantile Regression algorithm for 2-dimension is theoretically superior to all existing techniques.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The exponential growth of data brings unprecedented challenges in today's world. Although computational power has been increasing rapidly, it is failing to handle the pace of data growth. Due to the vastness of the database table, it is often impractical and unnecessary to calculate a database query's exact answer. Approximate Query Processing (AQP) is a technique that focuses on answering queries in near real-time by sacrificing a fraction of accuracy. This dissertation proposes different AQP techniques for two fundamental database problems using deep learning. Moreover, we also develop a scalable algorithm for the Quantile Regression problem, which is a crucial arsenal in minimizing the uncertainty of deep learning-based techniques, including our proposed approaches.

Our first problem is widely known as Selectivity Estimation – estimating the result size of a SQL query. The second problem concerns Approximate Query Processing for aggregate queries like Sum, Count, and Average. In the following two sections, we briefly overview our deep learning-based solutions for these two extremely important database problems. Finally, we highlight our scalable algorithm for Quantile Regression, followed by the dissertation outline.

1.1 Deep Learning Models for Selectivity Estimation of Multi-Attribute Queries

The Selectivity Estimation problem focuses on predicting the result size of a database query which is a fundamental database problem. Poor Selectivity estimation jeopardizes the query planning, resulting in inefficient query execution. There has been a continuous quest to find better Selectivity Estimation techniques. Many of these techniques are sampling-

based approaches which could have been better for multi-attribute queries. In recent years, some deep learning-based approaches are also proposed, which are promising yet need to be more accurate for a query with a large number of predicates and low selectivity. We propose two complementary deep learning-based solutions for the Selectivity Estimation problem.

In the first solution, we try to solve the problem as a neural density estimation problem. As selectivity estimation can be seen as a density estimation problem, finding the joint probability distribution is one way to solve this problem. The joint probability distribution can be calculated from conditional probabilities. However, calculating these different conditional probability tables is significantly space inefficient and time-consuming. A better alternative is to learn the conditional probability using a deep learning model. We propose an unsupervised technique that can accurately estimate the selectivity of a multi-attribute query with low selectivity. Our second approach focuses on finding selectivity in a supervised fashion.

We extend both of our approaches for range queries with multi-attributes. Moreover, we address other practical challenges like incorporating query workload, query/data featurization, etc. We show a detailed comparison of all the traditional approaches and current deep learning-based methods for selectivity estimation. Our experiment results demonstrate that our approaches are better than our competitors for queries with a large number of predicates and low selectivity.

1.2 Approximate Query Processing for Data Exploration using Deep Generative Models

Approximately answering aggregate queries is one of the most exciting research areas in the database community. Generally, this problem is widely referred to as Approximate Query Processing (AQP). As business uses data-driven approaches to reach a decision,

it is often required to answer a query quickly yet very accurately. Due to the immense importance of AQP, the database community has innovated many AQP techniques in the last few decades.

Many traditional database AQP techniques are based on sampling. An offline sampling often fails to capture the complex correlation among attributes. One alternative is to perform on-demand sampling. However, this approach is time-consuming. Moreover, a small percentage of data as samples requires a lot of space for big data. Due to the limitation of traditional approaches, we propose a deep learning-based method that can overcome these challenges.

We propose a deep generative model-based AQP technique that can generate samples to answer aggregate queries. Once the model is learned, the user can generate as many as the sample they want to answer an aggregate query approximately. One beauty of this approach is the model is extremely lightweight, only a few hundred kilobytes. We show how model bias can be removed by incorporating variational rejection sampling and generating high-quality samples. Our extensive experiments on multiple datasets demonstrate the efficacy of our technique.

1.3 Can Geometry Resolve Quantile Regression's Inefficiency?

In the previous two sections, we summarized our deep learning-based solutions for AQP. However, there is one problem with the learned model-based approach. These approaches are excellent in practice. However, they do not provide any confidence interval. In many critical applications, blindly using a black box model can be devastating. Uncertainty Quantification is an important research area that helps to overcome this issue. Quantile Regression (QR), a classical statistical/machine learning technique, is one of the most critical components of Uncertainty Quantification. This section highlights our contri-

bution to developing a scalable algorithm for Quantile Regression, especially for big data scenarios.

Although QR has been extensively studied for the last 50 years, QR still needs to be ready to handle big data. For a large dataset, QR state-of-the-art approaches are memory-inefficient and time-consuming. One approach to making QR faster is approximately calculating the QR. In this dissertation, we are only interested in the exact answer without surprise, as QR is a tool of uncertainty quantification for our specific case.

We utilize the computational geometry concepts of arrangement and duality to design an oracle that can calculate the objective function of QR from a neighboring point in the arrangement. Moreover, we propose a scalable algorithm using the computational geometry concepts of $k - set$, which is superior to any other QR technique in 2-dimension.

1.4 Dissertation Outline

In Chapter 2, we discuss our deep learning based Selectivity Estimation approach for multi-attribute queries. In Chapter 3, we present our AQP technique for Data Exploration using Deep Generative Models. Finally, in Chapter 3, we discuss our scalable Quantile Regression technique.

CHAPTER 2

Deep Learning Models for Selectivity Estimation of Multi-Attribute Queries

Selectivity estimation – the problem of estimating the result size of queries – is a fundamental problem in databases. Accurate estimation of query selectivity involving multiple correlated attributes is especially challenging. Poor cardinality estimates could result in the selection of bad plans by the query optimizer. Recently, deep learning has been applied to this problem with promising results. However, many of the proposed approaches often struggle to provide accurate results for multi attribute queries involving large number of predicates and with low selectivity.

In this paper, we propose two complementary approaches that are effective for this scenario. Our first approach models selectivity estimation as a density estimation problem where one seeks to estimate the joint probability distribution from a finite number of samples. We leverage techniques from neural density estimation to build an accurate selectivity estimator. The key idea is to decompose the joint distribution into a set of tractable conditional probability distributions such that they satisfy the autoregressive property. Our second approach formulates selectivity estimation as a supervised deep learning problem that predicts the selectivity of a given query. We describe how to extend our algorithms for range queries. We also introduce and address a number of practical challenges arising when adapting deep learning for relational data. These include query/data featurization, incorporating query workload information in a deep learning framework and the dynamic scenario where both data and workload queries could be updated. Our extensive experiments *with a special emphasis on queries with a large number of predicates and/or small*

result sizes demonstrates that our proposed techniques provide fast and accurate selective estimates with minimal space overhead.

2.1 Introduction

Selectivity estimation – the problem of estimating the result size of queries with multiple predicates – is a fundamental yet challenging problem in databases. It has diverse applications in query optimization, query profiling, database tuning, approximate query processing etc. Poor cardinality estimates could result in the selection of bad plans by the query optimizer [1]. Due to its importance, this problem has attracted intense interest from the database community.

Current Approaches and their Limitations. Accurate estimation of query selectivity involving multiple (correlated) attributes is especially challenging. Exactly representing the joint distribution is often infeasible when many attributes are involved or each attribute could take large number of values. Broadly speaking, major database systems tackle this problem by approximating this joint distribution via synopses or sampling. Synopsis techniques such as histograms approximate the joint frequency distribution in a bounded space by making assumptions such as uniformity and attribute value independence [2, 1]. These assumptions are often violated in real-world datasets resulting in large errors in selectivity estimation [1]. Building multidimensional histograms could partially ameliorate this issue but often has substantial space requirements. Sampling based approaches could handle attribute dependencies and correlations more effectively. However, it is not a panacea – for queries with low selectivity, the optimizer could be made to rely on magic constants [1], resulting in poor estimates.

2.1.1 Outline of Technical Results

In this paper, we investigate the suitability of Deep Learning (DL) for selectivity estimation. Building a DL model that is lightweight, fast to train and estimate, and optionally allow injection of domain knowledge such as query workload is non trivial. We propose two complementary approaches that operate in two phases. In the offline phase, we train an appropriate DL model from the data. During the online phase, the model accepts a query and outputs its selectivity.

Selectivity Estimation as Unsupervised Learning. Our first approach models selectivity estimation as a density estimation problem where one seeks to estimate the joint probability distribution from a finite number of samples. Intuitively, the traditional sampling and synopses approaches can be considered as approximate non-parameteric density estimators. However, instead of directly estimating the joint probability, we seek to decompose it into a series of simpler and tractable conditional probability distributions. Specifically, we consider a specific decomposition with *autoregressive* property (formally defined in Section 2.3). We then build a single DL model to simultaneously learn the parameters for each of the conditional distributions. Our approach is based on MADE [3] that adapts a standard autoencoder into an efficient neural density estimator.

Selectivity Estimation as Supervised Learning. We investigate if, given a training set of queries along with their true selectivity, is it possible to build a DL model that accurately predicts the selectivity of unknown queries involving multiple correlated attributes? Our proposed approach can be utilized to quickly train a model *without* having seen the data! The training set of queries with their true selectivities could be obtained from the query logs. Our DL models are lightweight and can provide selectivity estimates for datasets in few milliseconds. Our model outperforms other supervised estimation techniques such as Multiple Linear Regression and Support Vector Regression that have been applied for the

related problem of query performance prediction [4]. The key benefit factor is the ability of DL models to handle complex non linear relationships between queries involving correlated attributes and their selectivity.

Comparison with Other DL based Approaches. Recently, there has been intense interest in applying DL for selectivity estimation. Please refer to Section 2.6 for additional details. We primarily focus on multi-attribute queries involving large number of predicates and have low selectivity. This is often an Achilles heel of prior DL based approaches [5, 6, 7, 8]. As we shall show later in experiments, our proposed approaches can answer multi-attribute queries more accurately than competing approaches.

Summary of Experiments. We conducted an extensive set of experiments over two real-world datasets – Census and IMDB – that exhibit complex correlation and conditional independence between attributes and have been extensively used in prior work including [1, 5, 6]. We specifically focus on queries that have multiple attributes and/or small selectivity. We evaluated our supervised and unsupervised DL models on a query workload of 10K queries. Our supervised model was trained on a training data of 10K queries. Our results demonstrate that DL based approaches provide substantial improvement - for a fixed space budget - over prior approaches for *multi-attribute selectivity estimation* which has been historically a highly challenging scenario in database selectivity estimation.

Summary of Contributions.

- **Deep Learning for Selectivity Estimation.** We introduce an alternate view of database selectivity estimation namely as an neural density estimation problem and report highly promising results for both point and range predicates.
- **Making the approach suitable for Databases.** We describe adaptations making these models suitable for various data types, large number of attributes and associ-

ated domain cardinalities, availability of query workload and incremental queries and data.

Paper Outline. Section 4.2 introduces relevant notations and the design considerations for DL based selectivity estimators. Section 2.3 formulates selectivity estimation as an unsupervised neural density estimation problem and proposes an algorithm based on autoregressive models. In Section 2.4, we introduce the problem of selectivity estimation and propose a supervised Deep Learning based model for it. Section 3.6 describes our extensive experiments on real-world datasets, related work in Section 2.6 and finally conclude in Section 2.7.

2.2 Preliminaries

2.2.1 Notations

Let R be a relation with n tuples and m attributes $A = \{A_1, A_2, \ldots, A_m\}$. The domain of the attribute A_i is given by $Dom(A_i)$. We denote the value of attribute A_i of an arbitrary tuple as $t[A_i]$. We consider conjunctive queries on a single relation with both point and range predicates. Point queries are of the form $A_i = a_i$ AND $A_j = a_j$ AND $\ldots$ for attributes $\{A_i, A_j\} \subseteq A$ where $a_i \in Dom(A_i)$ and $a_j \in Dom(A_j)$. Range queries are of the form $lb_i \leq A_i \leq ub_i$ AND $lb_j \leq A_j \leq ub_j$ AND $\ldots$ Let q denote such a conjunctive query while $Sel(q)$ represents the result size. We use the normalized selectivity between $[0, 1]$ by dividing the result size by n, number of tuples.

Performance Measures. Given a query q, let the estimate provided by selectivity estimation algorithm be $\widehat{Sel(Q)}$. We use *q-error* for measuring the quality of estimates. Intuitively, q-error describes the factor by which the estimate differs from true selectivity. This metric is widely used for evaluating selectivity estimation approaches [1, 9, 10, 11] and is relevant for applications such as query optimization where the relative ordering is more im-

portant [1]. We do not consider the use of relative error due to its asymmetric penalization of estimation error [11] that results in models that systematically under-estimate selectivity.

$$\text{q-error} = \max\left(\frac{Sel(Q)}{\widehat{Sel(Q)}}, \frac{\widehat{Sel(Q)}}{Sel(Q)}\right) \tag{2.1}$$

2.2.2 Selectivity Estimation as Distribution Estimation

Given a set of attributes $A' = \{A_i, A_j, \ldots, \}$, the normalized selectivity distribution defines a valid (joint) probability distribution. The selectivity of a query q with $\{A_i = a_i, A_j = a_j, \ldots\}$ can be identified by locating the appropriate entry in the joint distribution table. Unfortunately, the number of entries in this table increases exponentially in the number of attributes and their domain cardinality.

Distribution estimation is the problem of learning the joint distribution from a set of finite samples. Often, distribution estimators seek to approximate the distribution by making simplifying assumptions. There is a clear trade-off between accuracy and space. Storing the entire distribution table produces accurate estimates but requires exponential space. On the other hand, heuristics such as attribute value independence (AVI) assume that the distributions of individual attributes A_i are independent of each other. In this case one needs to only store the individual attribute distributions and compute the joint probability as

$$p(A_i = a_i, A_j = a_j, \ldots) = \prod_{A_k \in A'} p(A_k = a_k)$$

Of course, this approach fails for most real-world datasets that exhibit correlated attributes. Most popular selectivity estimators such as multidimensional histograms, wavelets, kernel density estimations and samples can be construed as simplified non-parametric density estimators on their own.

2.2.3 Desiderata for DL Estimator

Given that selectivity estimation is just one component in the larger query optimization framework, we would like to design a model that aids in the identification of good query plans. Ideally, the estimator should be able to avoid the unrealistic assumptions of uniformity and attribute value independence (afflicting most synopses based approaches) and ameliorate issues caused by low selectivity queries (afflicting sampling based approaches). We would like to decouple training-accuracy tradeoff. For example, increasing the sample sizes improves the accuracy - at the cost of increasing the estimation time. If necessary, the estimator could have a large training time to increase accuracy but should have near constant estimation time. We would also like to decouple the space-accuracy tradeoff. Multi-dimensional histograms can provide reasonable estimates in almost constant time - but require very large space (that grows exponentially to the number of attributes) for accurate results. In other words, we would like to achieve high accuracy through a lightweight model. The desired model must be fast to train and given the latency requirements of query optimizer, generate estimates in milliseconds. It must also be able to appropriately model the complex relationship between queries and their selectivities. Finally, it must be able to leverage additional information such as query workload and domain knowledge.

2.3 Selectivity Estimation as Neural Density Estimation

We introduce an alternate view of selectivity estimation namely as an neural density estimation problem. This new perspective allows us to leverage the powerful tools from deep learning to get accurate selectivity estimation while also raising a number of non-trivial challenges in wrangling these techniques for a relational database setting.

Previously, there has been attempts on using Bayesian networks (BN) to approximate the joint distribution through a set of conditional probability distributions [12, 13]. However, learning the optimal structure of BN is prohibitively expensive. Representing conditional probability tables requires substantial storage for attributes with large domain cardinality. Our solution (a) avoids the expensive conditional independence decomposition using a simpler autoregressive decomposition; (b) *learns* the conditional probability tables instead of storing them using universal function approximation [14] capability of neural networks.

2.3.1 Density Estimation via Autoregressive Decomposition

The fundamental challenge is to construct density estimators that are expressive enough to model complex distributions while still being tractable and efficient to train. In this paper, we focus on autoregressive models [14] that satisfy these properties. Given a *specific ordering of attributes*, autoregressive models decompose the joint distribution into m conditional distributions $P(A_i|A_1, \ldots, A_{i-1})$. Specifically,

$$\begin{aligned} &p(A_1 = a_1, A_2 = a_2, \ldots, A_m = a_m) \\ &= \prod_{i=1}^{m} p(A_i = a_i | A_1 = a_1, A_2 = a_2, \ldots, A_{i-1} = a_{i-1}) \end{aligned} \tag{2.2}$$

Each of these conditional distributions is then learned using an appropriate DL architecture. The DL model first learns the distribution $p(A_1)$, followed by conditional distributions such as $p(A_2|A_1)$, $p(A_3|A_1, A_2)$ and so on. This process of sequentially *regressing* each attribute through its predecessors is known as autoregression [14].

2.3.2 Autoregressive Density Estimators

Given such a setting, we need to address two questions. First, which DL architecture should be used to learn the autoregressive conditional distributions? Second, how can we

identify an effective ordering of attributes for the decomposition? We answer the former in this section and the latter in Section 2.3.4.

Encoding Tuples. The first step in modelling is to encode the tuples such that they can be used efficiently for density estimation by a DL model. A naive approach would be to use one-hot encoding [14] of the tuples. Instead, we propose a *binary encoding* that represents them as a $\lceil \log_2 |Dom(A_j)| \rceil$ dimensional vector. If $Dom(A_j) = \{v_{j1}, v_{j2}, v_{j3}, v_{j4}\} = [0, 1, 2, 3]$, we represent $Dom(A_j)$ as $00, 01, 10, 11$ respectively. This approach is then repeated for each attribute individually and the representation for the tuple is simply the concatenation of the binary encoding of each attribute. This approach requires less storage - $\sum_{i=1}^{m} \lceil \log_2 |Dom(A_i)| \rceil$ dimensions instead of $\sum_{i=1}^{m} |Dom(A_i)|$ required by one-hot encoding.

Autoregressive Density Estimators using MADE. Given the encoding of the tuples, one could learn the conditional distributions by using any one of the neural autoregressive density estimators specifically designed for this purpose [15, 16, 3]. While our approach is agnostic to the specific estimator used, we advocate for the masked autoencoder architecture from [3]. MADE modifies the autoencoders [14, 17, 18] for efficiently estimating autoregressive distributions. Its flexible architecture allows us to effectively adapt it to relational domains.

2.3.3 Answering Range Queries

Once the autoregressive density estimator has been trained it could be used to answer *point* queries. Given a query $q : A_i = a_i$ AND $A_j = a_j$ AND $\ldots$, we can encode this query and feed it to the autoregressive density estimator model which will output the normalized selectivity. While these models cannot directly answer range queries, it is possible

to use their ability to answer point queries in a sophisticated way to obtain accurate range selectivity estimates.

Specifically, let us consider the question of answering range queries of the form: $q : A_1 \in R_1$ AND $A_2 \in R_2$ AND $\ldots$ where R_i is a range of the form $lb_i \leq A_i \leq ub_i$. Point queries of the form $A_i = a_i$ could be made into a range query $a_i \leq A_i \leq a_i$. Finally, if an attribute A_i is unspecified then it could be modeled as $\min(Dom(A_i)) \leq A_i \leq \max(Dom(A_i))$. For the rest of subsection, we consider a query Q of the form $A_1 \in R_1$ AND $\ldots$ AND $A_k \in R_k$. We would like to note that we do not *directly* feed the range queries to the autoregressive model which is designed to answer point queries only. Instead, we have proposed a Monte-Carlo approach that uses sampling and the point query selectivity estimator to get an estimate of the range query. The end-points of each range predicate are encoded using binary encoding.

Exhaustive enumeration where we enumerate all possible combination of the ranges and invoking the point query estimate is not feasible unless the range query is relative simple – involving small number of attributes and/or small ranges. The uniform sampling approach generates random queries by uniformly sampling from the ranges $R_1, \ldots, R_k$ and extrapolating it to get an overall estimate. However, this provides bad selectivity estimates when the number (and range) of predicates increases due to curse of dimensionality [19].

Adaptive Importance Sampling. The key insight to improve the naive uniform sampling is to make it *adaptive* and *weighted*. In other words, each sample could have a different weight and the probability with which a new point is selected could vary based on previously obtained samples. However, naively implementing this idea results in biased and incorrect results.

We adapt an algorithm [20] that was originally designed for Monte-Carlo multi-dimensional integration for the range selectivity estimation problem. Intuitively, we wish

to select samples S in proportion to the contribution they make to $sel(q)$. However, this leads to a chicken-and-egg problem as we use sampling to estimate $sel(q)$. The solution is to proceed in stages and use the information collected from samples of previous stages to improve the next stage.

Let $f(\cdot)$ be the probability density function based on query q such that if we sample points proportional to $f(\cdot)$, we might get accurate estimates. Of course, this information is not always available. Suppose that we have access to another simpler probability density function $g(\cdot)$ that is an approximation of $f(\cdot)$ and is also easier to sample from. Obviously, sampling from $g(\cdot)$ would provide much better estimates than uniform sampling. Given sample queries $q_1, \ldots, q_k$ generated using $g(\cdot)$, we can derive the estimate as

$$sel(Q) = \frac{|R_1| \times \ldots |R_k|}{|S|} \sum_{i=1}^{|S|} \frac{sel(q_i)}{g(q_i)} \tag{2.3}$$

Intuitively, we generated random queries based on $g(\cdot)$ and then appropriately corrected the bias to get an unbiased estimate. Now the remaining question is to obtain an efficient instantiation of $g(\cdot)$. We propose a simple approach inspired by Attribute Value Independence (AVI) assumption where

$$\begin{aligned} Sel(A_1 = a_1 \text{ AND } \ldots \text{ AND } A_k = a_k) = \\ Sel(A_1 = a_1) \times \ldots \times Sel(A_k = a_k) \end{aligned} \tag{2.4}$$

It is known that AVI assumption often provides an underestimate for correlated attributes [21]. We leverage this fact to decompose $g(A_1, A_2, \ldots, A_k)$ as k component functions $g_1(A_1)$, $g_2(A_2) \ldots$. One can then approximate the density of each of these attributes individually through existing synopses approaches such as histograms.

Our proposed approach operates in stages. We generate an initial batch of random queries through uniform sampling from the ranges. Using these random queries, we bootstrap the histograms for individual attributes. In the future stages, we generate samples in a

non-uniform way using the sampling distribution imposed by the attribute wise histograms. For example, a query $q_i = \{x_1 = a_1^i, \ldots, x_k = a_k^i\}$ will be picked proportional to the probability $g_1(a_1^i) \times \ldots \times g_k(a_k^i)$. So if some value $A_i = a_i$ occurs much more frequently then it will be reflected in the histogram of A_i and thereby will occur more frequently in randomly generated queries. Once all the sample queries are created, we then use Equation 2.4 to generate unbiased estimates for range selectivity.

A concurrent work [22] also proposed a Monte Carlo integration based technique dubbed progressive sampling for answering range queries. Our proposed approach based on adaptive importance sampling is faster due to the possibility of batching. In Section 3.6, we compare the performance of our model against progressive sampling and found that they provide comparable results.

2.3.4 Attribute Ordering for Autoregression

In practice, the best attribute ordering for autoregressive decomposition is not given to us and must be chosen appropriately for accurate selectivity estimation. Each of the permutations of the attributes forms a valid attribute ordering and could be used to estimate the joint distribution.

$$\begin{aligned} p(x) &= p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_1, x_2) \\ &= p(x_2) \cdot p(x_3|x_2) \cdot p(x_1|x_2, x_3) \\ &= p(x_3) \cdot p(x_2|x_3) \cdot p(x_1|x_2, x_3) \\ &= \ldots \end{aligned} \tag{2.5}$$

Random Attribute Ordering. Prior approaches such as Bayesian Networks deploy an expensive approach to identify a good ordering. We do away with this expensive step by choosing several random orderings of attributes. As we shall show experimentally, this approach works exceedingly well in practice. This is due to two facts: (a) the vast majority

of the $d!$ possible permutations are amenable to tractable and accurate learning; and (b) the powerful learning capacity of neural networks (and masked encoders) can readily learn even a challenging decomposition by increasing the depth of the MADE model. MADE architecture allows this to be easily and efficiently conducted by randomly permuting both the input tuple that is binary encoded and the internal mask vectors in each layer.

Ensembles of Attribute Orderings. While a random ordering often provides good results, it is desirable to guard against the worst case scenario of a bad permutation. We observe from Equation 2.5 that numerous attribute orderings could be used for estimating the joint distribution. We build on this insight by choosing κ random attribute orderings. Of course, different orderings result in different models with their corresponding estimate and associated accuracy. MADE could be used to learn the conditional distribution for each of these orderings and utilize them to estimate the value of $p(x)$ by averaging the individual estimates. An attribute ordering can be represented as $m^0 = [m^0(1), \ldots m^0(D)]$. In this, $m^0(d)$ represents the position of the d-th dimension of input $\mathbf{x}$ in the product of conditionals. Thus multiple random orderings can be obtained by permuting $[1, \ldots, D]$. This has also been discussed in [23].

During training, before each minibatch [14] update of the model, we apply κ random permutations in parallel on the input vectors and mask matrices. Each of these permutations corresponds to a different ordering. The models are learned independently and the joint probability is computed for each ordering and averaged to produce the final estimate. This ensemble approach minimizes the likelihood of a bad estimates due to an unlucky attribute ordering.

Injecting Domain Knowledge. If a domain expert possesses apriori knowledge that attributes A_i and A_j are order sensitive, then we only chose permutations where the desired order is observed. As a concrete example, assume one knows (say via data profiling), that

a functional dependency *EmpID* $\rightarrow$ *Department* exists on the schema. Then, we would prefer permutations where the *Department* occurs after *EmpID*. This is due to the fact that the conditional distribution $p(\text{Department}|\text{EmpID})$ is simpler and thereby easier to learn than the other way around.

2.3.5 Incorporating Query Workload

The autoregressive approach outlined above does not require a training dataset such as a query workload. However, it is possible to improve performance by leveraging query workload if available. Suppose that we are given a query workload $Q = \{q_1, \ldots, q_l\}$. We associate a weight $w(t)$ for each tuple $t \in R$ that corresponds to the number of queries that match t. So $w(t)$ can vary between 0 and l. Next, we assign higher penalties for poor estimates for tuples in the result set of multiple queries. The intuition is that a poor estimate for tuple t was caused by sub-optimal learning of parameter weights of the conditional distributions corresponding to the attribute values of t. As an example, consider a tuple $t = [0, 1]$ with two binary attributes A_1 and A_2. Suppose that we use a single attribute ordering A_2, A_1. If the selectivity of t was incorrectly estimated, then the entries corresponding to $p(A_2 = 1)$ and $p(A_1 = 0|A_2 = 1)$ must be improved. If t is in the result set of by many queries, then we prioritize learning the aforementioned parameter values through larger penalty. This could be achieved using the weighted cross-entropy loss function defined as,

$$-\log p(R) = \sum_{t \in R} \mathbf{w}(\mathbf{t}) \cdot \ell(t) \tag{2.6}$$

2.3.6 Incremental Data and Query Workload

Incremental Learning. The naive solution of retraining the entire model from scratch becomes progressively expensive as more and more batches of incremental data are added to R. We propose an incremental learning approach that *extends* the existing pre-trained

model by training it further only on the new data by initializing the model with the weights learned from the previous training, instead of performing the standard random initialization. We then continue training the model on new data. This two-step process preserves the knowledge gained from the past, absorbing knowledge from new data and is also more efficient. We use a smaller value for learning rate [14] and epochs than for the complete retraining so that the model is *fine-tuned*.

While incremental learning is conceptually simple, it must be done carefully. A naive training could cause *catastrophic forgetting* where the model "forgets" the old data and focuses exclusively on the new data. This is undesirable and must be avoided [24]. We propose the use of Dropout [25] and related techniques [26] to *learn without forgetting*. In our paper, we utilize a dropout value of $p = 0.1$ when training over the new batch of data. An additional complication arises from the fact that we already uses masks for maintaining the autoregressive property. Hence, we apply the dropout only on the neurons for which the masks are non-zero.

Incremental Workload. An autoregressive approach does not directly utilize query workload and hence could not use the information available from an incremental query workload. It is possible to reuse the techniques from Section 2.3.5 for this scenario. For each tuple, we update the number of queries it satisfies and retrain the model based on the new weights.

2.4 Selectivity Estimation as Supervised Learning

Our objective is to build a model that accepts an arbitrary query as input and outputs its selectivity. This falls under the umbrella of supervised learning methodologies using regression. Each query is represented as a set of features and the model learns appropriate weights for these features utilizing them to estimate the selectivity. The weights are learned

by training the model on a dataset of past queries (such as from query log or workload) and their true selectivities. Approaches such as linear regression, support vector regression etc that have been utilized for query performance prediction [4] are not suitable for building selectivity estimators. The impediment is the complex relationship between queries and their selectivities where simplifying assumptions such as attribute value independence do not hold. We leverage the powerful learning capacity of neural networks - with appropriate architecture and loss functions - to model this relationship.

2.4.1 Query Featurization

The first step is to encode the queries and their selectivities in an appropriate form suitable for learning.

Training Set. We are given a query training dataset $Q = \{(q_1, s_1), \ldots, \}$. Each query $q \in Q$ can be represented as an ordered list of m attribute pairs of (A_i, v_i) where $v_i \in Dom(A_i) \cup \{*\}$ (where * is used when A_i is unspecified). s_i denotes the normalized selectivity of q_i (*i.e.,* $\frac{Sel(q_i)}{n}$) where n is the number of tuples.

Example. Let $Q = \{(\{A_1 = 0, A_2 = 1\}, 0.3), (\{A_1 = 1, A_2 = *\}, 0.2), (\{A_1 = *, A_2 = *\}, 1.0)\}$. *i.e.,* Query q_2 with $A_1 = 1$ AND $A_2 = *$ has a selectivity of 0.2.

Encoding Queries. An intuitive representation for categorical attributes is one-hot encoding. It represents attribute A_i as $|Dom(A_i)| + 1$ dimensional vector that has 0 for all positions except the one corresponding to the value A_i takes. Given m attributes, the representation of the query is simply the concatenation of one-hot encoding of each of the attributes. The numeric attributes can be handled by treating them as categorical attributes by automatic discretization [27]. Alternatively, they can be specified as a normalized value $\in [0, 1]$ by min-max scaling. Note that this scheme can be easily extended to operators other than $=$. The only modification required is to represent the triplet $(A_i, operator_i, v_i)$

instead of just (A_i, v_i). Each operator could be represented as a fixed one-hot encoding of its own. Given d operators, each operator is represented as a d dimensional vector where the entry corresponding to $operator_i$ is set to 1. Of course, the rest of our discussion is oblivious to other mechanisms to encode the queries.

Encoding Selectivities. Each query $q \in Q$ is associated with the normalized selectivity $s_i \in [0, 1]$. Selectivities of queries often follow a skewed distribution where few queries have a large selectivity and the vast majority of queries have much smaller selectivities. Building a predictive model for such skewed data is often quite challenging. We begin by applying log transformation over the selectivity by replacing the selectivity s_i by its absolute log value as $abs(\log(s_i))$. For example, a selectivity of 0.00001 is specified as 5 (using log to the base of 10 for convenience). This has a smoothing effect on the values of a skewed distribution [28]. Our second transformation is min-max scaling where we rescale the output of the log transformation back to $[0, 1]$ range. Given a set of selectivities $S = \{s_1, s_2, \ldots, \}$ and a selectivity s_i, min-max scaling is computed as

$$s_i' = \frac{s_i - min(S)}{max(S) - min(S)} \tag{2.7}$$

While this transformation does not impact skew, it enables us to deploy well known activation functions such as sigmoid that are numerically stable. Prior works such as [10, 29] have also used log transformation followed by min max scaling to improve effectiveness of regression.

Example. Let the selectivities of 3 queries be $[0.1, 0.01, 0.001]$. By applying the log transformation, we get $[1, 2, 3]$. The corresponding min-max scaling gives $[0.0, 0.5, 1.0]$ where $0.5 = \frac{2-min([1,2,3])}{max([1,2,3])-min([1,2,3])}$.

2.4.2 DL Model for Selectivity Estimation

DL Architecture. Our DL architecture is based on a 3-layer fully connected neural network with rectifier activation function (ReLU) specified as $f(x) = max(0, x)$ ReLU is a simple non-linear activation function with known advantages such as faster training and sparser representations. The final layer uses a Sigmoid activation function $f(x) = \frac{1}{1+e^{-x}}$ Sigmoid is a popular function that squashes its parameter into a $[0, 1]$ range. One can then convert this output to true selectivity by applying inverse of min-max and log scaling. We used the Adam optimizer [30] for training the model.

Loss Function. Recall from Section 4.2 that the q-error metric is widely used to evaluate the selectivity estimator. Hence, it is desirable to train the DL model to directly minimize the mean Q-error of the training dataset.

$$\text{q-error}(Q) = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \max\left(\frac{s_i}{\widehat{s_i}}, \frac{\widehat{s_i}}{s_i}\right) \tag{2.8}$$

Selectivity Estimation via Inference Once the model is trained, it can be used for estimating the selectivity. Given a new query, we extract its features through one-hot encoding and feed it to the model. We apply the inverse transformation of min-max and log scaling on the output so that it represents the actual selectivity. The model is lightweight and the inference process often takes few milli-seconds when run on a GPU and/or CPU. Note that the time taken for training and estimation are decoupled. While the training time is proportional to the size of the training data, the inference is fixed for a given model.

2.4.3 Generating Training Data

While our DL model is relatively straightforward, it is possible to get good results through a careful construction of training data. We next describe how a training dataset could be constructed when query workload is not available. If query workload is available,

we describe a novel augmentation strategy such that the DL model can generate accurate estimates for unknown queries that are similar to the query workload.

No Query Workload Available. Naive sampling from the space of all queries results in a highly non-uniform training dataset and a sub-optimal selectivity estimator. Thus one must obtain a training set of queries that are diverse both in the number of predicates and their selectivities. Let the query budget be B - *i.e.,* we wish to construct a dataset with B queries and their selectivities. We begin by enumerating all queries with 1 predicates that are the atomic units from which multi-predicate queries could be estimated. We then generate multi-predicate queries where the predicates are chosen at random while the values are chosen based on their frequency. In order to generate a random query q_i, we first choose the number of predicates $k \in \{2, \ldots, m\}$ uniformly at random. Then we choose k attributes uniformly at random from the set of attributes $A = \{A_1, \ldots, A_m\}$. Let the selected attributes be $\{A_{i1}, \ldots, A_{ik}\}$. These two steps ensure that we have a diverse set of multi predicate queries both in terms of the number of predicates and the chosen predicates. Next, we choose a tuple t uniformly at random from the relation R. We create a random query q_i as the conjunction of predicates $A_{ij} = t[A_{ij}]$. This process ensures that the random query is selected proportional to the selectivity of query q_i.

Query Workload Available. If a query workload Q is available, one could directly utilize it to train the DL model. However, one can do much better by augmenting it, obtaining a more informative training set of queries. The key idea is to select queries from the distribution induced by the query workload such that the model generalizes to unknown queries from the same distribution. We need to address two issues. First, how can one generate random queries to augment the query workload? Second, how do we tune the model such that it provides accurate results for the workload?

We begin by assigning weights to attributes and attribute values based on their occurrence in the query workload. For example, if A_1 occurs 100 times while A_2 occurs 50 times, then weight of $A_1 = 2/3$ and $A_2 = 1/3$. We repeat this process for attribute values also. If an attribute value does not occur in the query workload, we assign a token frequency of 1. For example, if $A_1 = 1$ occurs 100 times while A_2 occurred none, then their weights are $100/101$ and $1/101$ respectively. We compute the frequency distribution of the number of predicates from the query workload (such as # queries with 1, 2, 3, ... predicates). This information is used to perform weighted sampling of the queries by extending the algorithm for the no workload scenario. This ensures that queries involving popular attributes and attribute values are generated at a higher frequency. Of course, sampling takes place without replacement so that all the queries in the augmented query workload are distinct.

Next, we assign different weights $w(q_i)$ to the queries $q_i \in Q'$ from the augmented workload to ensure that the model prioritizes the accuracy of queries from the workload.

$$w(q_i) = \begin{cases} 1 & \text{if } q_i \in Q \\ \frac{|Q|}{|Q'|} & \text{if } q_i \notin Q \end{cases} \tag{2.9}$$

We then train the DL model where the penalty for a query q is weighed proportionally to whether it came from the original or the augmented query workload.

$$\text{q-error}(Q) = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \mathbf{w(q_i)} \times \left(\frac{s_i}{\widehat{s_i}} + \frac{\widehat{s_i}}{s_i} \right) \tag{2.10}$$

2.4.4 Miscellaneous Issues

Incremental Data. Our supervised approach does not even look at the data and only uses the query training dataset. When incremental data arrives, the selectivities of some of these queries would change. We then train the model on the dataset with the updated selectivites.

Incremental Query Workload. In this case, it is possible to use the incremental training algorithm as described in Section 2.3.6. We initialize the supervised model with the weights

from previous training run instead of random initialization. We train the model on the new data with a reduced learning rate and a smaller number of epochs. We also use Dropout regularization technique with probability $p = 0.1$ to avoid catastrophic forgetting.

2.5 Experiments

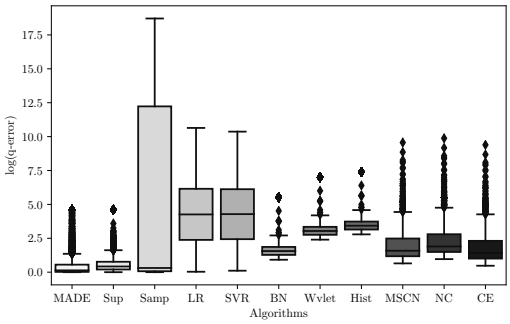

Figure 2.1: Comparison with Baselines (Census)

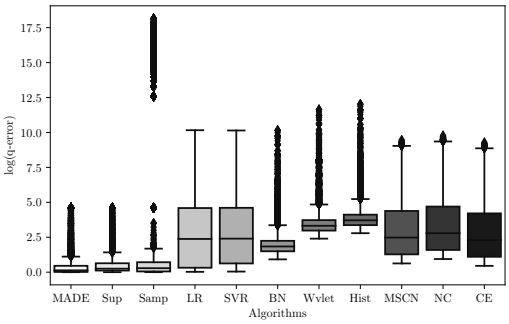

Figure 2.2: Comparison with Baselines (IMDB)

In our evaluation, we consider the following key questions:

1. How does our proposed methods compare against traditional selectivity estimation approaches?
2. How does our methods compare against prior supervised DL based approaches?
3. How does our work compare with a concurrent work Naru [22] in terms of point and range queries?

2.5.1 Experimental Setup

Hardware and Platform. All our experiments were performed on a NVidia V100 GPU. The CPU is a quad-core 2.2 GHz machine with 16 GB of RAM. We used PyTorch for building the DL models.

Datasets. We conducted our experiments on two real-world datasets: Census [12] and IMDB dataset from Join Order Benchmark [1]. Each of these datasets have complex correlated attributes and conditional dependencies. Selectivity estimation on multiple predicates especially on IMDB datasets is quite challenging. The Census dataset has 50K rows, 8 categorical and 6 numerical attributes. The IMDB dataset consists of 21 tables with information about movies, actors, directors etc. It contains as much as 2.5M movie titles and over 4M actors. For our experiments, we used two large tables Title.akas and Title.basics containing 3.4M and 5.3M tuples with 8 and 9 attributes respectively. Finally, we also evaluate our algorithms on a synthetic dataset from TPC-H benchmark using a scale factor of 1.

Algorithms for Selectivity Estimation. The unsupervised model consists of a 4 layer masked autoencoder with 128 neurons in each layer and was trained for 10 epochs by default. The supervised model consists of 3 fully connected layers with 100 neurons and ReLU activation function. The final layer has sigmoid activation function to convert the output in the range $[0, 1]$. The training data consists of 10K queries (see details in Section 2.4). The true selectivities are transformed by log scaling followed by min-max scaling. We trained it for 100 epochs.

Query Workload. We compared the algorithms on a test query log of 10K queries. We generated the log to thoroughly evaluate the performance of the estimators for various facets such as number of predicates, selectivity, size of joint probability distribution, attribute correlation etc. Census has 8 categorical attributes thereby creating $\binom{8}{1} + \binom{8}{2} + \ldots \binom{8}{8} = 255$ possible attribute combinations. The 10K workload was equally allotted such that there are 1250 queries with exactly 1 predicate, 1250 queries with 2 predicates and so on. There are $\binom{8}{i}$ combinations with exactly i attributes. We allocate 1250 equally

between $\binom{8}{i}$ combinations. For a specific attribute combination, we pick their values randomly without replacement from their respective domains.

Performance Measures. We used q-error defined in Section 4.2 for measuring the estimation quality. Recall that q-error of 1 corresponds to perfect estimate while a q-error of 2 corresponds to an under- or over-estimate by a factor of 2 and so on. We also use box-plots to concisely describe the results of 10K queries. The middle line corresponds to the median q-error while the box boundaries correspond to the 25th and 75th percentiles. The top and bottom whiskers are set to show the 95th and 5th percentiles.

2.5.2 Comparison with Baselines

In our first set of experiments, we demonstrate the efficacy of our approaches against popular baseline approaches such as multi-dimensional histograms [2, 31, 32, 33, 34, 35], wavelets [36], Bayesian networks [12, 13] and sampling [37]. Our experiments were conducted on Postgres and leverage the recently introduced multi-column statistics feature from Postgres 10. We use the TABLESAMPLE command in Postgres 10 with Bernoulli sampling to obtain a 1% sample. Haar wavelets are widely used in selectivity estimation and approximate query processing [36, 38] as they are accurate and can be computed in linear time. We used standard Haar wavelet decomposition algorithm described in [38] for handling multi-dimensional data. We implemented the algorithm described in [12]. We also evaluated our approach against Linear Regression (denoted as LR) [39] and Support Vector Regression (denoted as SR) [40] that has been previously used for a related area of query performance monitoring [4].

Bayesian Network (BN) are disproportionately affected by the domain cardinality. For example, consider three attributes A_1, A_2, A_3 such that A_1 and A_2 are the parents of A_3. The conditional probability table of A_3 needs $O(|A_1| \times |A_2| \times |A_3|)$ entries. For the

IMDB dataset that has millions of actors and/or movie titles, this results in an explosion of space requirements. For the initial set of experiments involving non-DL baselines over IMDB dataset, we perform it only on a 1% sample. Since even the 1% sample has 25K movies and 110K actors, we use an additional entropy based discretization [27] so that it fits into the space budget. Unfortunately, our efforts to discretize the entire IMDB dataset so as to fit the space budget produced inferior BN with poor results. We repeated our experiment over 10 different 1% samples and report the average results in Figure 2.2. Once we show the superiority of our method over non-DL baselines, we report the results for the DL based approaches for the entire IMDB and TPC-H datasets in Tables 2.1 and 2.2.

We also evaluated our algorithms against 3 representative DL based approaches [10, 8, 7] for which either the code or the complete DL architecture specification was available. MSCN [10] is a multi-set convolutional network that has been used for answering correlated joins effectively. MSCN can express query features using sets resulting in smaller model sizes [10]. NeuralCubes (NN) [7] is another approach that uses DL to estimate queries with application in interactive data exploration. It is a holistic approach that uses an encoder-decoder based architecture to learn embedding for each feature and uses a DL model to estimate the aggregates. CE is a cost based estimator first proposed in [8]. While it is used for estimating the cost of a query, it has a component for query estimation that we use for estimation.

For fair comparison, all the selectivity estimators are allocated the same space budget. Specifically, MSCN required 3MB of space and all other non-DL models were modified to fit this space budget. Sup, MADE and Naru used less than 2MB of space. Increasing the space budget did not materially improve the accuracy. We used additional optimizations such as half-precision to reduce the storage of our supervised and unsupervised models. This is done post-training and results in a smaller model with almost no loss in accuracy.

We also applied model pruning on the supervised model. This resulted in our supervised model requiring space of around 200KB. Our MADE model used a space of around 1MB.

Figures 2.1 and 2.2 present the results. We can observe that our DL based approaches dramatically outperform all the prior methods. The baseline approaches of LR [39] and SVR [40] provide inaccurate results; both Census and IMDB exhibit complex correlation and conditional dependencies, which these techniques are unable to adapt to. The sampling based approach provides good estimates for queries with high selectivities but dramatically drops off in accuracy for queries with low/very-low selectivities. Wavelets and histograms provide performance comparable to our methods. However, this is due to the fact that we disproportionately allocated much more resources for them than our approaches. Interestingly, the closest baseline is BN that is related in principle to our algorithm. However, our approaches are superior to BN in both accuracy and time. Specifically our approaches are 2 times more accurate on average for Census and 100 times more accurate for the worst case error. Similar trends hold for IMDB in terms of accuracy. As a point of reference, it takes one minute to train our approaches on Census versus 16 minutes for BN (correspondingly 12 minutes of training for our approaches for IMDB versus 516 minutes for BN). Figures 2.1 and 2.2 shows that both our unsupervised and supervised approaches either outperform the competing approaches or are comparable with them. For supervised approaches, this is due to our approach to construct and/or augment an existing workload. For unsupervised approaches, the outperformance is due to the ability of MADE to accurate learn the (autoregressive) conditional probability distributions.

Comparing DL based Models. In the next set of experiments, we focus on the DL based approaches. Our experiments were conducted on the entire IMDB and TPC-H datasets. Table 2.1 shows the results for the IMDB dataset. We partition the queries based on the

number of predicates and display the median and 99-th percentile for each of the cases. We can see that our proposed approaches outperform other supervised DL based approaches.

	1-3		**4-6**		**7-9**	
	50th	**99th**	**50th**	**99th**	**50th**	**99th**
Sup	1.01	1.12	1.09	2.78	1.77	4.2
MADE	1.01	1.06	1.02	1.22	1.08	2.26
NARU	1.01	1.06	1.02	1.21	1.08	2.27
MSCN	1.08	1.67	1.78	2.1	3.8	281
NC	1.32	1.98	3.2	98	9.6	323
CE	1.18	1.87	1.55	4.7	5.6	108

Table 2.1: Breakdown of DL model performance based on #predicates (1-3, 4-6 and 7-9) for IMDB Dataset.

We also performed additional experiments on TPC-H dataset to show that our proposed approaches can generalize across multiple distributions. The query workload was constructed in the same way for IMDB and Census. Table 2.2 shows that the results are similar as before. Our proposed approaches can achieve excellent results on TPC-H.

	1-3		**4-6**		**7-9**	
	50th	**99th**	**50th**	**99th**	**50th**	**99th**
Sup	1.03	1.08	1.1	2.23	1.7	4.9
MADE	1.01	1.02	1.03	1.08	1.12	1.9
NARU	1.01	1.02	1.03	1.08	1.12	1.91
MSCN	1.06	1.12	1.34	1.56	4.7	172
NC	1.24	1.89	5.6	66	10.7	264
CE	1.12	1.43	1.77	3.8	6.3	131

Table 2.2: Point Queries on TPC-H. Queries are partitioned based on the number of predicates.

The sample workload provided by Kipf et al [10] is for answering queries over multiple tables. Since our focus is on answering multi-predicate queries on a single table, this

was not applicable. Hence, we implemented the training data generation scheme described in Section 3.3 of [10] and obtained a training dataset of equivalent size (10K queries). We found that the performance of MSCN is heavily dependent on the sampling scheme and is especially vulnerable to queries with low selectivities. Even when we doubled the query workload (to 20K) queries, the performance of MSCN was worse than ours. A similar behaviour was also observed in [22].

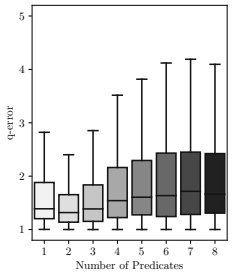

Figure 2.3: Varying #Predicates (Supervised)

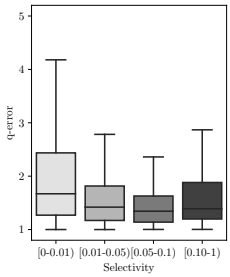

Figure 2.4: Varying Selectivity (Supervised)

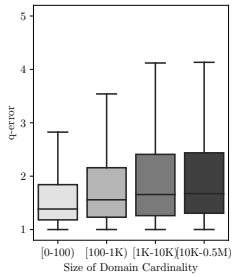

Figure 2.5: Varying Domain Cardinality (Supervised)

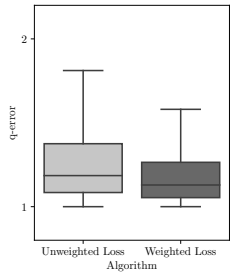

Figure 2.6: Workload (Supervised)

2.5.3 Supervised Cardinality Estimation

We have established that our proposed approaches outperform DL and non-DL baselines. In this subsection, we describe additional experimental results that investigates the impact of various relevant factors of this outperformance.

Varying #Predicates in Query. Figure 2.3 presents the result of varying the number of predicates for the census dataset. The results for the larger and more challenging datasets of IMDB and TPC-H can be found in Tables 2.1 and 2.2. We partition the query workload based on the number of predicates involved and report the median and 99-th percentile of q-error. Not surprisingly, the q-errors of DL based approaches are low for queries with

small number of predicates which are often easy to accurately estimate. However, these approaches start to falter for queries with higher number of predicates. These queries often have low selectivity and exhibit complex correlations between the attributes. In contrast, our supervised approach gives accurate estimates across the board for queries with small as well as large number of predicates.

Varying Query Selectivity. We group the queries in the test set based on their selectivity and investigate how the selectivity of the query impacts of our approach. Figure 2.4 shows the results. Not surprisingly, our approach provides very accurate estimates for queries with selectivity of 1% or more. Even when the selectivity is less than 1%, our approach is quite accurate with a median q-error of less than 2.

Varying Domain Cardinality. The attributes of IMDB dataset vary between 4 to almost half a million. Figure 2.5 show that our proposed approach performs well for all types of attributes thanks to the effective encoding of the values and the construction of training set.

Leveraging Query Workload. Our proposed approach leverages workload in two ways. First, the training dataset for the supervised approach is generated to be similar to the workload. Second, the objective function is modified so that errors on queries similar to workload are penalized with a higher weight. It is plain that using a training dataset based on workload gives better results. Hence, we test the impact of weighting queries differently on the performance of the model. Figure 2.6 shows the performance of two variants of the supervised approach. While both these approaches construct the training dataset similar to the workload, the way the objective function is modified is different. The *Unweighted Loss* variant gives equal weight to the errors from all queries in the training query workload. On the other hand, *Weighted Loss* gives higher penalty to errors on queries similar to the workload. Figure 2.6 shows that both these modifications are needed for obtaining good results.

	Encoding	50th	75th	99th
Census	One-Hot	1.01	1.04	1.1
Census	Binary	1.01	1.04	1.08
IMDB	One-Hot	1.02	1.17	1.61
IMDB	Binary	1.01	1.04	1.12

Table 2.3: Impact of Value Encoding.

Impact of Encoding. In our paper, we discussed two ways of encoding the query – one-hot encoding or the binary encoding. Table 2.3 shows the impact of encoding on the model performance in terms of percentile q-errors. When the domain cardinality is small such as for Census dataset, the impact of encoding is minimal and is only visible at the tail end of q-error. However, when the cardinality is huge such as for IMDB dataset, then the impact is non-trivial. This to be expected as the binary encoding requires substantially less number of parameters to learn in the input layer of the DL model than the one-hot encoding. Given a fixed training dataset, a model with lesser number of parameters will avoid the overfitting. Furthermore, the encoding also has a nice side-effect of reducing the model size.

Impact Log Transformation and Min-Max Scaling. We use log transformation and min-max scaling as a pre-processing step for query selectivity. Table 2.4 shows that the transformation has a significant impact on the performance of our supervised model. This is due to the fact that the selectivity of queries in the training dataset are very skewed. Furthermore, the selectivity of a significant number of queries (such as the queries with multiple predicates) are often very small. Unscaled selectivity has a deletrious effect on the model performance and gives large errors for queries with low selectivity as shown by the 99-th percentile q-errors. It also slows down the convergence of the model training.

Hyperparameter Tuning. We conducted additional experiments varying hyperparameters such as epochs, number of layers and number of hidden units. Our DL model consists of 3 fully connected layers with 100 hidden units. We varied the number of epochs as (50,

	Transformation	50th	75th	99th
Census	No	1.07	1.73	4.8
Census	Yes	1.01	1.04	1.08
IMDB	No	1.04	1.92	11.27
IMDB	Yes	1.01	1.04	1.12

Table 2.4: Impact of Log Transformation.

100, 200, 500), number of layers between 2-6, number of hidden units as (50, 100, 200, 500). We found that using a 100 epochs provided the best result. A smaller number of epochs resulted in underfitting while more than that resulted in overfitting as measured by the performance over a separately held validation set. We found that increasing the number of layers beyond 3 provided minimal improvements and made the model prone to overfitting and increased the model size. The choice of 3 layers with 100 hidden units each fit the sweet spot in terms of lower q-error and smaller model size.

Summary. Our supervised model produces more accurate results and requires less storage than each of the prior work that use supervised DL models. Better accuracy is due to the careful selection of training dataset, log-transformation / min-max scaling and an effective use of workload when available. As shown from the experiments above, the combination of these characteristics allows us to have a smaller model architecture while achieving good performance for queries with low selectivity and/or multiple predicates. We also use number of tricks like half-precision and model pruning to further reduce the model size. This does not materially affect the accuracy.

2.5.4 Unsupervised Density Estimation

Varying #Predicates in Query. Figure 2.7 depicts how our unsupervised approach behaves for queries with varying number of predicates. As expected, the approach is very accurate for queries with small number of predicates. This is unsurprising as they could be

easily learnt by most selectivity estimators. We can observe however that our estimates are very accurate and within a factor of two even for queries *with as much as 7-8 predicates*. Often queries with large number of predicates have small selectivities and exhibit complex correlations. Despite those challenges our methods provide very good performance.

Varying Query Selectivity. Next, we group the queries in the 10K test set based on their selectivity. Figure 2.8 presents that, our approach provides very accurate estimates for queries with selectivity of 5% or more. Even when the selectivities are low or very low, our method is still able to provide excellent estimates that are off by a factor of at most 2 for 75% percent of the query test set.

Varying Domain Cardinality. Figure 2.9 shows that the overall trend is comparable to that of the supervised approach. For attributes with small number of domain values, the estimates are very accurate. It slowly deteriorates for attributes with larger domain cardinality.

Impact of Query Workload. As expected, the availability of query workload improves the performance of our unsupervised approach. As expected, the improvement for supervised is much better than for the unsupervised model. This is to be expected as the supervised estimator is trained on the query workload while the unsupervised is trained on the relation and does not use the workload information directly. Nevertheless, our retrofitted approach that uses a weighted cross entropy loss function does provide a meaningful improvement in performance.

Handling Incremental Data. Our proposed approaches can naturally handle incremental data. In order to evaluate them, we randomly permuted the IMDB dataset and grouped them into three partitions. The first partition P1 consists of 50% of the data and P2 and P3 consist of the remainder. We consider two cases - NoIns and Ins. In the former, the new tuples does not contain any attribute domain values. In the latter, this could result in new domain

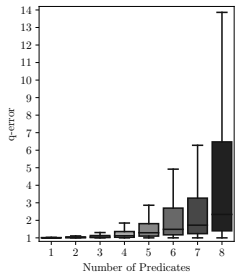

Figure 2.7: Varying #Predicates (Unsupervised)

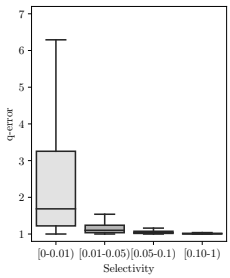

Figure 2.8: Varying Selectivity (Unsupervised)

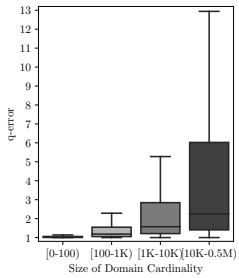

Figure 2.9: Varying Domain Cardinality

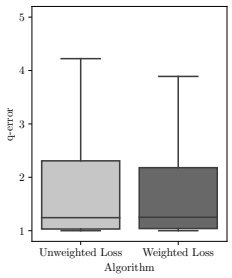

Figure 2.10: Workload (Unsupervised)

	P1	**P2**	**P3**
Sup-NoIns	2.8	2.82	2.86
MADE-NoIns	1.53	1.56	1.63
Sup-Ins	2.8	8.7	119
MADE-Ins	1.53	4.2	6.8

Table 2.5: Incremental Data with and without insertions on IMDB.

values. For an unordered (categorical) attributes, we impose an arbitrary ordering of the attribute values and use binary encoding to represent the values. For a new domain value, the next available binary representation is allotted. Similarly, the domain values need not be dense. Consider an attribute whose domain is $\{1, 20, 100, 1000\}$. By default, our binary encoding will use 10 bits to encode these values $(2^{10} = 1024)$. This allows one to encode any value between $0 - 1023$ without any major changes. If the inserted values are larger than 1023, then one would require a different encoding and re-training. This approach is not unique to binary encoding. Even other approaches such as one-hot encoding or embedding based also suffer from the same issue. We have conducted an experiment to highlight this phenomenon (Table 2.5). We report the 99-th percentile of q-error. Our current approach could easily and effectively handle incremental data where the domain of the attributes are either not modified or only have limited insertions. Their efficacy drops if there is a large amount of insertions and would require retraining of the model.

2.5.5 Comparison with Naru

Our unsupervised selectivity estimator is very similar to Naru [22] which is a concurrent work. We conduct additional experiments comparing our method with Naru's MADE based model using the code from its GitHub repository. Both our approaches uses an neural autoregressive model for learning the conditional probability distribution and use it for answering point queries. Given this similarity, it is not surprising that the results from Tables 2.1 and 2.2 for IMDB and TPC-H dataset show comparable accuracy. Our model is smaller than Naru's due to the use of half-precision and simpler binary encoding. This also has an auxilliary benefit of making the inference faster.

Range Queries. We next evaluated the performance of our approach and Naru for range queries. Since IMDB had limited number of numerical attributes, we increased the pool by artificially encoding each categorical variable as an integer. For example, {a, b, c} is now encoded as {0, 1, 2}. Table 2.6 shows the results. Both our approaches leverage the autoregressive model that was designed for answering point queries for answering range queries. We use an adaptive importance sampling algorithm inspired by Monte-Carlo multi-dimensional integration [20] while Naru uses a progressive sampling [22].

Given a range query with say K predicates, Naru works as follows. First, it obtains the probability of i-th predicate conditioned on previous values. Then, it generates a sample value for i-th attribute. Finally, the conditional probabilities are multiplied together to get an estimate of the selectivity. We can see that this approach proceeds sequentially in stages one attribute at a time where the probability estimation and sampling steps are interspersed.

In contrast, our approach has the notion of batches that is used to parallelize the estimation. Given a query budget, we partition them into batches (say of size 25 or 50). Given a query with K range predicates, we use K functions $(g_1, g_2, \ldots, g_K)$. The function g_i help us in smartly sampling a value $a_i \in R_i$ (where R_i is the range predicate for attribute

A_i). Each of g_i is based on a single attribute and the sampling can be done in parallel for each attribute. Additionally, we also keep the value of g_i fixed during the batch. This allows us to generate multiple samples from a weighted distribution efficiently. After each batch is processed, we issue the entire batch of randomly constructed to the point query estimator, update the values of g_i and repeat the process. We can see that this notion of batching allows one to *vectorize* key steps such as sampling, invoking the estimator, updating g_i and so on. Since GPUs can run the vectorized code efficiently, higher the batch size, faster is the running time.

	1-3		**4-6**		**7-9**	
	50th	**99th**	**50th**	**99th**	**50th**	**99th**
MADE	1.01	1.08	1.06	1.34	1.12	3.17
NARU	1.01	1.08	1.06	1.36	1.12	3.23

Table 2.6: Range Queries on IMDB.

Attribute Ordering. We show the results of attribute ordering in Table 2.7. Naru uses an ordering based on mutual information. Our approach is based order-agnostic training of MADE where we train the model on multiple orderings. Even using a small number of orderings gives good performance. In addition, as mentioned in Section 3.4, we use domain knowledge or information gleaned from data profiling such as functional dependencies to choose attribute ordering. The experimental results show that both these heuristics provide comparable results.

	IMDB		**TPC-H**	
	50th	**99th**	**50th**	**90th**
MADE	1.02	1.32	1.01	1.92
NARU	1.02	1.31	1.01	1.91

Table 2.7: Impact of Attribute Ordering on number of predicates (1-3, 4-6 and 7-9) for IMDB Dataset.

Inference Timings. Even when using the same model architecture as Naru, the median inference time of our model is slightly faster by 0.1ms (7.4ms to 7.3ms). The difference is due to smaller model size (at the input layer) required by our simpler encoding. We also use additional optimizations such as half-precision that dramatically shrink our model and decrease the inference time to 3.8ms

2.6 Related Work

Deep Learning for Databases. Recently, there has been extensive work on applying techniques from DL for solving challenging database problems. One of the first work was by Kraska et. al [41] that sought to build learned indexes. There are two conceptual connections between our approach and [41]. First, they both seek to leverage the data distribution for the task at hand. Second, both of these leverage the concept of ensembles/mixtures to improve the performance of individual DL models. However, the problem they tackle are very different – density estimation vs indexing. Another difference is that [41] uses the cumulative distribution function (CDF) [42] while our unsupervised approach uses probability density estimation (PDF). There has been extensive work on using DL techniques including reinforcement learning for query optimization (and join order enumeration) such as [43, 44, 45, 46]. Recently, there has been effort to build a learned database systems [47] and an end-to-end learned optimizers [48, 8]. DL has also been applied to the problem of entity resolution in [49] and data integration [50, 51].

Traditional Approaches for Selectivity Estimation. Due to the importance of selectivity estimation, there has been extensive work on accurate estimation. Popular approaches for estimation include sampling [37], histograms [2, 31, 32, 33, 34, 35], wavelets [36], kernel density estimation [52, 53, 54] and graphical models [12, 13].

ML based Approaches for Selectivity Estimation. Due to its versatility, ML has been explored for the problem of selectivity estimation. One of the earliest approaches to use neural networks is [55]. While promising, the recently proposed techniques such as neural density estimation are much more accurate. Another relevant recent work is [10, 56] that focuses on estimating correlated join selectivities. It proposes a novel set based DL model but focuses mostly on supervised learning. In contrast we consider both supervised and unsupervised approaches. An empirical analysis of various approaches can be found in [5]. Recently, there has been efforts to focus on challenging types of queries including group-by [57], range [22, 29] and spatial [58] queries. Our algorithm for range queries is much simpler than that of [22] and can provide comparable results. There is also some promising work on using deep learning for approximate query processing such as [59, 6, 7].

2.7 Final Remarks

In this paper, we proposed DL models for the fundamental problem of selectivity estimation. We proposed two complementary approaches that modeled the problem as an unsupervised and supervised learning respectively. For the former, we leveraged MADE – a neural density estimation technique based on masked autoencoders. For the latter, we proposed a DL architecture to learn the selectivity from a query log augmented with true selectivities. Our extensive experiments showed that the results are very promising and can address some of the pain points of popular DL based selectivity estimators. There are a number of promising avenues to explore. For one, how to extend the unsupervised selectivity estimators over single tables to multiple tables involving correlated joins. Another intriguing direction is to investigate the possibility of other deep generative models such as deep belief networks (DBN), variational auto encoders (VAE) and generative adversarial networks (GANs) for the purpose of selectivity estimation.

CHAPTER 3

Approximate Query Processing for Data Exploration using Deep Generative Models

Data is generated at an unprecedented rate surpassing our ability to analyze them. The database community has pioneered many novel techniques for Approximate Query Processing (AQP) that could give approximate results in a fraction of time needed for computing exact results. In this work, we explore the usage of deep learning (DL) for answering aggregate queries specifically for interactive applications such as data exploration and visualization. We use *deep generative models*, an unsupervised learning based approach, to learn the data distribution faithfully such that aggregate queries could be answered approximately by *generating* samples from the learned model. The model is often compact – few hundred KBs – so that arbitrary AQP queries could be answered on the client side without contacting the database server. Our other contributions include identifying model bias and minimizing it through a rejection sampling based approach and an algorithm to build model ensembles for AQP for improved accuracy. Our extensive experiments show that our proposed approach can provide answers with high accuracy and low latency.

3.1 Introduction

Data driven decision making has become the dominant paradigm for businesses seeking to gain an edge over competitors. However, the unprecedented rate at which data is generated surpasses our ability to analyze them. Approximate Query Processing (AQP) is a promising technique that provides *approximate* answers to queries at a fraction of the cost needed to answer it exactly. AQP has numerous applications in data exploration and

visualization where approximate results are acceptable as long as they can be obtained near real-time.

Case Study. Consider an user who performs data exploration and visualization on a popular dataset such as NYC Taxi dataset. The user issues ad-hoc aggregate queries, involving arbitrary subsets of attributes of interest, such as *what is the average number of passengers on trips starting from Manhattan?* or *what is the average trip duration grouped by hour?* and so on. Since this is for exploratory purposes, an imprecise answer is often adequate. A traditional approach is to issue aggregate queries to the database server, get exact or approximate answers accessing the base data or pre-computed/on-demand samples and display the returned results to the user. However, this could suffer from high latency that is not conducive for interactive analysis. In this paper, we propose an alternate approach where the approximate results could be computed entirely at the client side. Specifically, we build a deep generative model that approximates the data distribution with high fidelity and is lightweight (few hundreds of KBs). This model is sent to the client and could be used to generate *synthetic* samples over which AQP could be performed locally on arbitrary subsets of attributes, without any communication with the server. Our approach is complementary to traditional AQP exploring a new research direction of utilizing deep generative models for data exploration. It offers a lightweight model that can answer arbitrary queries which we experimentally demonstrate exhibit superior accuracy. For queries requiring provable guarantees we default to traditional AQP or exact query evaluation.

3.1.1 Outline of Technical Results

Deep Learning for AQP. Deep Learning (DL) [14] has become popular due to its excellent performance in many complex applications. In this paper, we investigate the feasibility of using DL for answering aggregate queries for data exploration and visualization. Struc-

tured databases seem intrinsically different from prior areas where DL has shined - such as computer vision and natural language processing. Furthermore, the task of generating approximate estimates for an aggregate query is quite different from common DL tasks. However, we show that AQP can be achieved in an effective and efficient manner using DL models.

Deep Generative Models for AQP. Our key insight is to train a DL model to learn the data distribution of the underlying data set effectively. Once such a model is trained, it acts as a concise representation of the dataset. The samples generated from the model have a data distribution that is almost identical to that of the underlying dataset. Hence, existing AQP techniques [60, 61] could be transparently applied on these samples. Furthermore, the model could generate as many samples as required without the need to access the underlying dataset. This makes it very useful for interactive applications as all the computations could be done locally.

Technical Challenges. The key challenge is to identify a DL based distribution estimation approach that is expressive enough to reflect statistical properties of real-world datasets and yet tractable and efficient to train. It must be non-parametric and not make any prior assumption about data characteristics. A large class of DL techniques - dubbed collectively as deep generative models - could be used for this purpose. Intuitively, a deep generative model is an unsupervised approach that learns the probability distribution of the dataset from a set of tuples. Often, learning the exact distribution is challenging, thus generative models learn a model that is very similar to the true distribution of the underlying data. This is often achieved through neural networks that learn a function that maps the approximate distribution to the true distribution. Each of the generative models have their respective advantages and disadvantages. We focus on variational autoencoders [62] that aim to learn

a low dimensional latent representation of the training data that optimizes the log-likelihood of the data through evidence lower bound.

3.2 Preliminaries

Consider a relation R with n tuples and m attributes $A_1, A_2, \ldots, A_m$. Given a tuple t and an attribute A_i, we denote the value of A_i in t as $t[A_i]$. Let $Dom(A_i)$ be the domain of attribute A_i.

Queries for AQP. In this paper, we focus on aggregate analytic queries of the general format:

```
SELECT g, AGG(A) FROM R
WHERE filter GROUP BY G
```

Of course, both the WHERE and GROUP BY clauses are optional. Each attribute A_i could be used as a *filter* attribute involved in a predicate or as a *measure* attribute involved in an aggregate. The filter could be a conjunctive or disjunctive combination of conditions. Each of the conditions could be any relational expression of the format `A op CONST` where A is an attribute and *op* is one of $\{=, \neq, <, >, \leq, \geq\}$. AGG could be one of the standard aggregates `AVG, SUM, COUNT` that have been extensively studied in prior AQP literature. One could use other aggregates such as `QUANTILES` as long as a statistical estimator exists to generate aggregate estimates.

Performance Measures. Let q be an aggregate query whose true value is θ. Let $\tilde{\theta}$ be the estimate provided by the AQP system. Then, we can measure the estimation accuracy through relative error defined as

$$RelErr(q) = \frac{|\tilde{\theta} - \theta|}{\theta} \tag{3.1}$$

For a set of queries $Q = \{q_1, \ldots, q_r\}$, the effectiveness of the AQP system could be computed through average relative error. Let θ_j and $\tilde{\theta_j}$ be the true and estimated value of the aggregate for query q_j.

$$AvgRelErr(Q) = \frac{1}{r}\sum_{j=1}^{r}\frac{|\tilde{\theta_j} - \theta_j|}{\theta_j} \tag{3.2}$$

We could also use the average relative error to measure the accuracy of the estimate for `GROUP BY` queries. Suppose we are given a group by query q with groups $G = \{g_1, \ldots, g_r\}$. It is possible that the sample does not contain all of these groups and the AQP system generates estimates for groups $\{g_{j_1}, \ldots, g_{j'_r}\}$ where each $g_{j_i} \in G$. As before, let θ_{j_i} and $\widetilde{\theta_{j_i}}$ be the true and estimated value of the aggregate for group g_{j_i}. By assigning 100% relative error for missing groups, the average relative error for group by queries is defined as,

$$AvgRelErr(q) = \frac{1}{r}\left((r - r') + \sum_{i=1}^{r'}\frac{|\widetilde{\theta_{j_i}} - \theta_{j_i}|}{\theta_{j_i}}\right) \tag{3.3}$$

3.3 Background

In this section, we provide necessary background about generative models and variational autoencoders in particular.

Generative Models. Suppose we are given a set of data points $X = \{x_1, \ldots, x_n\}$ that are distributed according to some unknown probability distribution $P(X)$. Generative models seek to learn an approximate probability distribution Q such that Q is very similar to P. Most generative models also allow one to generate samples $X' = \{x'_1, \ldots, \}$ from the model Q such that the X' has similar statistical properties to X. Deep generative models use powerful function approximators (typically, deep neural networks) for learning to approximate the distribution.

Variational Autoencoders (VAEs). VAEs are a class of generative models [62, 63, 64] that can model various complicated data distributions and generate samples. They are very efficient to train, have an interpretable latent space and could be adapted effectively to different domains such as images, text and music. Latent variables are an intermediate data representation that captures *data characteristics* used for generative modelling. Let X be the relational data that we wish to model and z a latent variable. Let $P(X)$ be the probability distribution from which the underlying relation consisting of attributes $A_1, \ldots, A_m$ was derived and $P(z)$ as the probability distribution of the latent variable. Then $P(X|z)$ is the distribution of generating data given latent variable. We can model $P(X)$ in relation to z as $P(X) = \int P(X|z)P(z)dz$ marginalizing z out of the joint probability $P(X, z)$. The challenge is that we do not know $P(z)$ and $P(X|z)$. The underlying idea in variational modelling is to infer $P(z)$ using $P(z|X)$.

Variational Inference. We use a method called Variational Inference (VI) to infer $P(z|X)$ in VAE. The main idea of VI is to approach inference as an optimization problem. We model the true distribution $P(z|X)$ using a simpler distribution (denoted as Q) that is easy to evaluate, e.g. Gaussian, and minimize the difference between those two distribution using KL divergence metric, which tells us how different P is from Q. Typically, the simpler distribution depends on the attribute type. Gaussian distribution is often appropriate for real numbers while Bernoulli distribution is often used for categorical attributes. Assume we wish to infer $P(z|X)$ using $Q(z|X)$. The KL divergence is specified as:

$$\begin{aligned} D_{KL}[Q(z|X)||P(z|X)] = \sum_z Q(z|X) \log(\frac{Q(z|X)}{P(z|X)}) = \\ E[\log(\frac{Q(z|X)}{P(z|X)})] = E[\log(Q(z|X)) - \log(P(z|X))] \end{aligned} \tag{3.4}$$

We can connect [62] $Q(z|X)$ which is a projection of the data into the latent space and $P(X|z)$ which generates data given a latent variable z through Equation 3.5 that is also called as the variational objective.

$$\begin{aligned} &\log P(X) - D_{KL}[Q(z|X||P(z|X)] \\ &= E[\log P(X|z)] - D_{KL}[Q(z|X||P(z)] \end{aligned} \tag{3.5}$$

Encoders and Decoders. A different way to think of this equation is as $Q(z|X)$ encoding the data using z as an intermediate data representation and $P(X|z)$ generates data given a latent variable z. Typically $Q(z|X)$ is implemented with a neural network mapping the underlying data space into the latent space (*encoder network*). Similarly $P(X|z)$ is implemented with a neural network and is responsible to generate data following the distribution $P(X)$ given sample latent variables z from the latent space (*decoder network*). The variational objective has a very natural interpretation. We wish to model our data $P(X)$ under some error function $D_{KL}[Q(z|X||P(z|X)]$. In other words, VAE tries to identify the lower bound of $\log(P(X))$, which in practice is good enough as trying to determine the exact distribution is often intractable. For this we aim to maximize over some mapping from latent variables to $\log P(X|z)$ and minimize the difference between our simple distribution $Q(z|X)$ and the true latent distribution $P(z)$. Since we need to sample from $P(z)$ in VAE typically one chooses a simple distribution to sample from such as $N(0, 1)$. Since we wish to minimize the distance between $Q(z|X)$ and $P(z)$ in VAE one typically assumes that $Q(z|X)$ is also normal with mean $\mu(X)$ and variance $\Sigma(X)$. Both the encoder and the decoder networks are trained end-to-end. After training, data can be generated by sampling z from a normal distribution and passing it to the decoder network.

3.4 AQP Using Variational AutoEncoders

In this section, we provide an overview of our two phase approach for using VAE for AQP. This requires solving a number of theoretical and practical challenges such as input encodings and approximation errors due to model bias.

Our Approach. Our proposed approach proceeds in two phases. In the *model building* phase, we train a deep generative model M_R over the dataset R such that it learns the underlying data distribution. In this section, we assume that a single model is built for the entire dataset that we relax in Section 3.5. Once the DL model is trained, it can act as a succinct representation of the dataset. In the *run-time* phase, the AQP system uses the DL model to *generate* samples S from the underlying distribution. The given query is rewritten to run on S. The existing AQP techniques could be transparently used to generate the aggregate estimate. Figure 3.1 illustrates our approach.

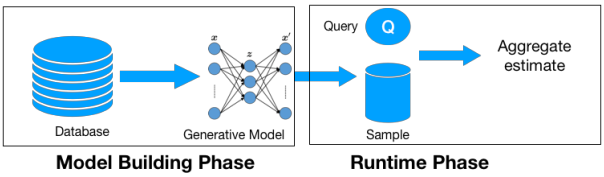

Figure 3.1: Two Phase Approach for DL based AQP

3.4.1 Using VAE for AQP

In this subsection, we describe how to train a VAE over relational data and use it for AQP.

Input Encoding. In contrast to homogeneous domains such as images and text, relations often consist of mixed data types that could be discrete or continuous. The first step is to represent each tuple t as a vector of dimension d. For ease of exposition, we consider one-

hot encoding and describe other effective encodings in Section 3.4.5. One-hot encoding represents each tuple as a $d = \sum_{i=1}^{m} |Dom(A_i)|$ dimensional vector where the position corresponding to a given domain value is set to 1. Each tuple in a relation R with two binary attributes A_1 and A_2, is represented as a 4 dimensional binary vector. A tuple with $A_1 = 0, A_2 = 1$ is represented as $[1, 0, 0, 1]$ while a tuple with $A_1 = 1, A_2 = 1$ is represented as $[0, 1, 0, 1]$. This approach is efficient for small attribute domains but becomes cumbersome if a relation has millions of distinct values.

Model Building and Sampling from VAE. Once all the tuples are encoded appropriately, we could use VAE to learn the underlying distribution. We denote the size of the input and latent dimension by d and d' respectively. For one hot encoding, $d = \sum_{i=1}^{m} |Dom(A_i)|$. As d' increases, it results in more accurate learning of the distribution at the cost of a larger model. Once the model is trained, it could be used to generate samples X'. The randomly generated tuples often share similar statistical properties to tuples sampled from the underlying relation R and hence are a viable substitute for R. One could apply the existing AQP mechanisms on the generated samples and use it to generate aggregate estimates along with confidence intervals.

The sample tuples are generated as follows: we generate samples from the latent space z and then apply the decoder network to convert points in latent space to tuples. Recall from Section 3.3 that the latent space is often a probability distribution that is easy to sample such as Gaussian. It is possible to speed up the sampling from arbitrary Normal distributions using the reparameterization trick. Instead of sampling from a distribution $N(\mu, \sigma)$, we could sample from the standard Normal distribution $N(0, 1)$ with zero mean and unit variance. A sample ϵ from $N(0, 1)$ could be converted to a sample $N(\mu, \sigma)$ as $z = \mu + \sigma \odot \epsilon$. Intuitively, this shifts ϵ by the mean μ and scales it based on the variance σ.

3.4.2 Handling Approximation Errors.

We consider approximation error caused due to model bias and propose an effective rejection sampling to mitigate it.

Sampling Error. Aggregates estimated over the sample could differ from the exact results computed over the entire dataset and their difference is called the sampling error. Both the traditional AQP and our proposed approach suffer from sampling error. The techniques used to mitigate it - such as increasing sample size - can also be applied to the samples from the generative model.

Errors due to Model Bias. Another source of error is sampling bias. This could occur when the samples are not representative of the underlying dataset and do not approximate its data distribution appropriately. Aggregates generated over these samples are often biased and need to be corrected. This problem is present even in traditional AQP [60] and mitigated through techniques such as importance weighting [65] and bootstrapping [66, 60].

Our proposed approach also suffers from sampling bias due to a subtle reason. Generative models learn the data distribution which is a very challenging problem - especially in high dimensions. A DL model learns an approximate distribution that is *close enough*. Uniform samples generated from the approximate distribution would be biased samples from the original distribution resulting in biased estimates. As we shall show later in the experiments, it is important to remove or reduce the impact of model bias to get accurate estimates. Bootstrapping is not applicable as it often works by *resampling* the sample data and performing inference on the sampling distribution from them. Due to the biased nature of samples, this approach provides incorrect results [66]. It is challenging to estimate the importance weight of a sample generated by VAE. Popular approaches such as IWAE [67] and AIS [68] do not provide strong bounds for the estimates.

Rejection Sampling. We advocate for a rejection sampling based approach [69, 70] that has a number of appealing properties and is well suited for AQP. Intuitively, rejection sampling works as follows. Let x be a sample generated from the VAE model with probabilities $p(x)$ and $q(x)$ from the original and approximate probability distributions respectively. We accept the sample x with probability $\frac{p(x)}{M \times q(x)}$ where M is a constant upper bound on the ratio $p(x)/q(x)$ for all x. We can see that the closer the ratio is to 1, the higher the likelihood that the sample is accepted. On the other hand, if the two distributions are far enough, then a larger fraction of samples will be rejected. One can generate arbitrary number of samples from the VAE model, apply rejection sampling on them and use the accepted samples to generate unbiased and accurate aggregate estimates.

In order to accept/reject a sample x, we need the value of $p(x)$. Estimating this value - such as by going to the underlying dataset - is very expensive and defeats the purpose of using generative models. A better approach is to approximately estimate it purely from the VAE model.

Variational Rejection Sampling Primer. We leverage an approach for variational rejection sampling that was recently proposed in [69]. For the sake of completeness, we describe the approach as applied to AQP. Please refer to [69] for further details. Sample generation from VAE takes place in two steps. First, we generate a sample z in the latent space using the variational posterior $q(z|x)$ and then we use the decoder to convert z into a sample x in the original space. In order to generate samples from the true posterior $p(z|x)$, we need to accept/reject sample z with acceptance probability

$$a(z|x, M) = \frac{p(z|x)}{M \times q(z|x)} \tag{3.6}$$

where M is an upper bound on the ratio $p(z|x)/q(z|x)$. Estimating the true posterior $p(z|x)$ requires access to the dataset and is very expensive. However, we do know that the value

of $p(x, z)$ from the VAE is within a constant normalization factor $p(x)$ as $p(z|x) = \frac{p(x,z)}{p(x)}$. Thus, we can redefine Equation 3.6 as

$$a(z|x, M') = \frac{p(x, z)}{M \times p(x) \times q(z|x)} = \frac{p(x, z)}{M' \times q(z|x)} \tag{3.7}$$

We can now conduct rejection sampling if we know the value of M'. First, we generate a sample z from the variational posterior $q(z|x)$. Next, we draw a random number U in the interval $[0, 1]$ uniformly at random. If this number is smaller than the acceptance probability $a(z|x, M')$, then we accept the sample and reject it otherwise. That way the number of times that we have to repeat this process until we accept a sample is itself a random variable with geometric distribution p = $P(U \leq a(z|x, M'))$; $P(N = n) = (1 - p)^{n-1}p, n \geq 1$. Thus on average the number of trials required to generate a sample is $E(N) = 1/p$. By a direct calculation it is easy to show [70] that $p = 1/M'$. We set the value of M' as $M' = e^{-T}$ where $T \in [-\infty, +\infty]$ is an arbitrary threshold function. This definition has a number of appealing properties. First, this function is differentiable and can be easily plugged into the VAE's objective function thereby allowing us to learn a suitable value of T for the dataset during training [69]. Please refer to Section 3.6 for a heuristic method for setting appropriate values of T during model building and sample generation. Second, the parameter T when set, establishes a trade-off between computational efficiency and accuracy. If $T \to +\infty$, then every sample is accepted (*i.e.,* no rejection) resulting into fast sample generation at the expense of the quality of the approximation to the true underlying distribution. In contrast when $T \to -\infty$, we ensure that almost every sample is guaranteed to be from the true posterior distribution, by making the acceptance probability small and as a result increasing sample generation time. Since a should be a probability we change equation Equation 3.7 to:

$$a(z|x, M') = \min\left[1, \frac{p(x, z)}{M' \times q(z|x)}\right] \tag{3.8}$$

3.4.3 Towards Accuracy Guarantees

As mentioned in Section 3.1, our approach is complementary to traditional AQP system. Our objective is to design a lightweight deep generative model that could be used to obtain quick-and-dirty aggregate estimates that are often sufficient for preliminary data exploration. Once the user has identified promising queries that requires provable guarantees, we can defer traditional AQP techniques or even obtain exact answers. In this subsection, we describe an initial approach for obtaining the accuracy guarantees. We would like to note that developing a framework to quantify approximation errors of AQP based on deep generative models is a challenging problem and a focus of our future research.

Eliminating Model Bias. Recall from Section 3.4.2 that approximation errors incurred in our approach are due to model bias and sampling error. If the model bias is eliminated, then our problem boils down to the traditional AQP setting. We could readily leverage the rich set of accuracy guarantees and confidence intervals developed for handling sampling error. This is achieved by setting $T = -\infty$ and applying variational rejection sampling (VRS). However, this comes with a large computational cost whereby the vast majority of generated tuples are rejected. Ideally, we would like a granular accuracy-computation tradeoff. Increasing T improves the sampling efficiency at the cost of model bias.

Distribution Testing. We adapt techniques designed for high-dimensional two-sample hypothesis testing [71, 72] for choosing appropriate T. Suppose we are given two sets of uniform random samples S_D and S_M from the original dataset and the learned model respectively. Let $|S_D| = |S_M|$. Suppose that these samples were drawn from probability distributions P_D and P_M. If we can ascertain that the two distributions are the same (i.e. $P_D = P_M$), by using the corresponding samples, then we can safely ignore the issue of model bias. This is achieved by testing the null hypothesis $H_0 : P_D = P_M$.

There are two factors that makes this challenging: high-dimensionality and test-statistics for AQP. First, we train VAE model by transforming tuples into a vector whose dimensionality ranges in the thousands. Classical tests such as Kolmogrov-Smirnov are not suitable for testing such high dimensional distributions. Second, hypothesis testing methods rely on a test statistic that is a function of S_D and S_M that could be used to distinguish P_D and P_M. For example, a simple test statistic is to choose an aggregate query such as estimating the average value of some attribute A_i. If the average of A_i computed over S_D and S_M deviates beyond certain threshold we can reject the null hypothesis. However, this is not appropriate for our scenario. We wish to test the null hypothesis for arbitrary aggregate queries. The way out of this conundrum is to use *Cross-Match Test* [71, 72].

Cross-Match Test for AQP. We begin by projecting tuples in S_D and S_M into the latent space of VAE using the encoder. We abuse the notation by representing the projected tuples as S_D and S_M. Let $Z = S_D \cup S_M$. We associate a label of 0 if tuple $t \in S_D$ and a label of 1 if $t \in S_M$. We construct a *complete* graph where each node corresponds to a tuple in Z while the edge corresponds the Euclidean distance between the latent space representation of the corresponding tuples. We then compute a minimum weight perfect matching using the Blossom algorithm [73]. The output is a collection of non-overlapping pairs of tuples. Consider a specific pair of tuples (Z_i, Z_j). There are three possibilities: both tuples are from S_D, both tuples are from S_M or one each from S_D and S_M. Let $a_{D,D}, a_{M,M}, a_{D,M}$ be the frequency of pairs from the matching of these three categories. The cross-match test [71, 72] specifies $a_{D,M}$ as the test statistic. Let $\eta = a_{D,D} + a_{M,M} + a_{D,M}$. We accept or reject the null hypothesis based on the probability computed as

$$\frac{2^{a_{D,M}} \times \eta!}{\binom{\eta}{|S_D|}(a_{D,D})!(a_{M,M})!(a_{M,D})!} \tag{3.9}$$

3.4.4 Variational Autoencoder AQP Workflow

Algorithm 1 provides the pseudocode for the overall workflow of performing AQP using VAE. In the model building phase, we encode the input relation R using an appropriate mechanism (see Section 3.4.5). The VAE model is trained on the encoded input and stored along with appropriate metadata.

During the runtime phase, we generate sample S_M from VAE using variational rejection sampling with $T = 0$. We then apply the hypothesis testing to ensure that the two distributions cannot be distinguished. If the null hypothesis is rejected, we generate a new sample S_D with a lower value of T. This will ensure that the model bias issue is eliminated. One can then apply existing techniques for generating approximation guarantees and confidence intervals. Note that we use the VAE model for data exploration only after it passed the hypothesis testing.

3.4.5 Making VAE practical for relational AQP

In this subsection, we propose two practical improvements for training VAE for AQP over relational data.

Effective Input Encoding. One-hot encoding of tuples is an effective approach for relatively small attribute domains. If the relation has millions of distinct values, then it causes two major issues. First, the encoded vector becomes very sparse resulting in poor performance [74]. Second, it increases the number of parameters learned by the model thereby increasing the model size and the training time.

A promising approach to improve one-hot encoding is to make the representation denser using *binary encoding*. Without loss of generality, let the domain $Dom(A_j)$ be its zero-indexed position $[0, 1, \ldots, |Dom(A_j)| - 1]$. We can now concisely represent these values using $\lceil \log_2 |Dom(A_j)| \rceil$ dimensional vector. Once again consider the example

Algorithm 1 AQP using VAE

1: **Input:** VAE model $\mathcal{V}$

2: $T = 0$, S_D = sample from D,

3: $S_z = \{\}$ //set of samples

4: **while** samples are still needed **do**

5: Sample $z \sim q(z|x)$

6: Accept or reject z based on Equation 3.8

7: **If** z is accepted, $S_z = S_z \cup \{z\}$

8: S_M =Decoder(S_z) // Convert samples to original space

9: Test null hypothesis $H_0 : P_S = P_D$ using Equation 3.9

10: **if** H_0 is rejected **then**

11: $T = T - 1$

12: Goto Step 3

13: **Output:** Model $\mathcal{V}$ and T

$Dom(A_j) = \{Low, Medium, High\}$. Instead of representing A_j as a 3-dimensional vectors (*i.e.,* $001, 010, 100$), we can now represent them in $\lceil \log_2(3) \rceil = 2$-dimensional vector *i.e.,* $\eta(Low) = 00, \eta(Medium) = 01, \eta(High) = 10$. This approach is then repeated for each attribute resulting a $d = \sum_{i=1}^{n} \lceil \log_2 |Dom(A_i)| \rceil$-dimensional vector (for n attributes) that is exponentially smaller and denser than the one-hot encoding that requires $\sum_{i=1}^{n} |Dom(A_i)|$ dimensions.

Effective Decoding of Samples. Typically, samples are obtained from VAE in two steps: (a) generate a sample z in the latent space *i.e.,* $z \sim q(z|x)$ and (b) generate a sample x' in the original space by passing z to the decoder. While this approach is widely used in many domains such as images and music, it is not appropriate for databases. Typically, the output of the decoder is stochastic. In other words, for the same value of z, it is possible to generate

multiple reconstructed tuples from the distribution $p(x|z)$. However, blindly generating a random tuple from the decoder output could return an invalid tuple. For images and music, obtaining incorrect values for a few pixels/notes is often imperceptible. However, getting an attribute wrong could result in a (slightly) incorrect estimate Typically, the samples generated are often more correct than wrong. We could minimize the likelihood of an aberration by generating multiple samples for the same value of z. In other words, for the same latent space sample z, we generate multiple samples $X' = \{x'_1, x'_2, \ldots, \}$ in the tuple space. These samples could then be aggregated to obtain a single sample tuple x'. The aggregation could be based on max (*i.e.,* for each attribute A_j, pick the value that occurred most in X') or weighted random sampling (*i.e.,* for each attribute A_j, pick the value based on the frequency distribution of A_j in X'). Both these approaches provide sample tuples that are much more robust resulting in better accuracy estimates.

3.5 AQP using Multiple VAEs

So far we have assumed that a single VAE model is used to learn the data distribution. As our experimental results show, even a single model could generate effective samples for AQP. However, it is possible to improve this performance and generate better samples. One way to accomplish this is to split the dataset into say K non-overlapping partitions and learn a VAE model for each of the partitions. Intuitively, we would expect each of the models to learn the finer characteristics of the data from the corresponding partition and thereby generate better samples for that partition. In this section, we investigate the problem of identifying the optimal set of K partitions for building VAE models.

3.5.1 Problem Setup

Typically, especially in OLAP settings, tuples are grouped according to hierarchies on given attributes. Such hierarchies reflect meaningful groupings which are application

specific such as for example location, product semantics, year, etc. Often, these groupings have a semantic interpretation and building models for such groupings makes more sense than doing so on an arbitrary subset of the tuples in the dataset. As an example, the dataset could be partitioned based on the attribute Country such that all tuples belonging to a particular country is an atomic group. We wish to identify K non-overlapping groups of countries such that a VAE model is trained on each group.

More formally, let $G = \{g_1, g_2, \ldots, g_l\}$ be the set of existing groups with $g_i \subseteq R$ such that $\cup_{i=1}^{l} g_i = R$. We wish to identify a partition $S = \{s_1, \ldots, s_K\}$ of R where $s_i \subseteq G$ and $s_i \cap s_j = \emptyset$ when $i \neq j$. Our objective is to group these l subsets into K non-overlapping partitions such that the aggregate error of the VAEs over these partitions is minimized.

Efficiently solving this problem involves two steps: (a) given a partition, a mechanism to *estimate* the error of K VAEs trained over the partition *without* conducting the actual training and (b) an algorithm that uses (a) to identify the best partition over the space of partitions. Both of these challenges are non-trivial.

3.5.2 Bounding VAE Errors

Quantifying VAE Approximation. The parameters of VAE are learned by optimizing an evidence lower bound (ELBO) given by

$$E[\log P(X|z)] - D_{KL}[Q(z|X)||P(z)]$$

(from Equation 3.5) which is a tight bound on the marginal log likelihood. ELBO provides a meaningful way to measure the distribution approximation by the VAE. Recall from Section 3.4.2 that we perform rejection sampling on the VAE that results in a related measure we call R-ELBO (resampled ELBO) defined as

$$E[\log P(X|z)] - D_{KL}[R(z|X,T)||P(z)]$$

where $R(z|X,T)$ is the resampled distribution for a user-specified threshold of T. Given two VAEs trained on the same dataset for a fixed value of T, the VAE with lower R-ELBO provides a better approximation.

Bounding R-ELBO for a Partition. Let us assume that we will train a VAE model for each of the atomic groups $g_j \in G$. We train the model using variational rejection sampling [69] for a fixed T and compute its R-ELBO. In order to find the optimal partition, we have to compute the value of R-ELBO for arbitrary subsets $s_i = \{g_{i1}, \ldots, \} \subseteq G$. The naive approach would be to train a VAE on the union of the data from atomic groups in s_i which is time consuming. Instead, we empirically show that it is possible to bound the R-ELBO of VAE trained on s_i if we know the value of R-ELBO of each of $g_{i1}, \ldots,$. Let $f(\cdot)$ be such a function. In this paper, we take a conservative approach and bound it by sum $f(r_1, r_2, \ldots, r_k) = \sum_{i=1}^{k} r_i$ where r_i is the R-ELBO for group g_i. In other words, $f(\cdot)$ bounds the R-ELBO of VAE trained $\cup_{i=1}^{k} g_i$ by $\sum_{i=1}^{k} r_i$. It is possible to use other functions that provide tighter bounds.

Empirical Validation. We empirically validated the function $f(\cdot)$ on a number of datasets under a variety of settings. Table 3.1 show the results for Census and Flights dataset that has been widely used in prior work on AQP such as [75, 76, 12]. Please refer to Section 3.6 for a description of the two datasets. We obtained similar results for other benchmark datasets. For each of the datasets, we constructed multiple atomic groups for different categorical attributes. For example, one could group the Census dataset using attributes such as gender, income, race etc. We ensured that each of the groups are at least 5% of the data set size to avoid outlier groups and if necessary merged smaller groups into a miscellaneous group. We trained a VAE model on each of the groups for different values of T using variational rejection sampling and computed their R-ELBO. We then construct all pairs, triples, and other larger subsets of the groups and compare the bound obtained by

$f(\cdot)$ with the actual R-ELBO value of the VAE trained on the data of these subsets. For each dataset, we evaluated 1000 randomly selected subsets and report the fraction in which the bound was true. As is evident in table 3.1 the bound almost always holds.

Table 3.1: Empirical validation of R-ELBO Bounding

Dataset	$T = -10$	$T = 0$	$T = +10$
Census	0.992	0.997	0.996
Flights	0.961	0.972	0.977

3.5.3 Choosing Optimal Partition

In this section we assume we are provided with the value of R-ELBO for each of the groups $g_i \in G, 1 \leq i \leq l$, a bounding function $f(\cdot)$ and a user specified value K. We propose an algorithm that optimally splits a relation D into K non overlapping partitions $S = \{s_1, \ldots, s_K\}$ where $s_i \subseteq G$ and $s_i \cap s_j = \emptyset$ when $i \neq j$. The key objective is to choose the split S in such a way that the $\sum_{i=1}^{K}$ R-ELBO(s_i) is minimized. Note that there are K^l possible partitions and exhaustively enumerating and choosing the best partition is often infeasible. R-ELBO(g) corresponds to the actual R-ELBO for atomic groups $g \in G$ while for $s_i \subseteq G$, this is estimated using the bounding function $f(s_i)$. We investigate scenarios that occur in practice.

Optimal Partition using OLAP Hierarchy. In OLAP settings, tuples are grouped according to hierarchies on given attributes that reflect meaningful semantics. We assume the availability of an OLAP hierarchy in the form of a tree where the leaf node corresponds to the atomic groups (*e.g.,* Nikon Digital Cameras) while the intermediate groups correspond to product semantics (*e.g.,* Digital Camera $\rightarrow$ Camera $\rightarrow$ Electronics and so on). We wish to build VAE on meaningful groups of tuples by constraining s_i to be selected from the leafs

or intermediate nodes, be mutually exclusive and have the least aggregate R-ELBO score. We observe that the selected nodes forms a tree cut that partitions the OLAP hierarchy into K disjoint sub-trees.

Let us begin by considering the simple scenario where the OLAP hierarchy is a binary tree. Let h denote an arbitrary node in the hierarchy with *left(h)* and *right(h)* returning the left and right children of h if they exist. We propose a dynamic programming algorithm to compute the optimal partition. We use the table $Err[h, k]$ to denote aggregate R-ELBO of splitting the sub-tree rooted at node h using at most k partitions where $k \leq K$. The base case $k = 1$ is simply building the VAE on all the tuples falling under node h. When $k > 1$, we evaluate the various ways to split h such that the aggregate R-ELBO is minimized. For example, when $k = 2$, there are two possibilities. We could either not split h or build two VAE models over *left(h)* and *right(h)*. The optimal decision could be decided by choosing the option with least aggregate error. In general, we consider all possible ways of apportioning K between the left and right sub-trees of h and pick the allocation resulting in least error. The recurrence relation is specified by,

$$Err[h,k] = \begin{cases} \text{R-ELBO(h)} & \text{if k=1} \\ \min_{1 \leq i \leq k}(Err[\text{left(h)}, i] + & \\ Err[\text{right(h)}, k-i]) & \text{otherwise} \end{cases} \tag{3.10}$$

The extension to non-binary trees is also straightforward. Let $C = \{c_1, \ldots, c_j\}$ be the children of node h. We systematically partition the space of children into various groups of two and identify the best partitioning that gives the least error (eq. 3.11). A

similar dynamic programming approach was also used for constructing histograms over hierarchical data in [77].

$$Err[h,k] = \begin{cases} \text{R-ELBO(h)} & \text{if k=1} \\ \min_{1 \le i \le k}(Err[\{c_1, \ldots, c_{j/2}\}, i] + & \\ Err[\{c_{j/2+1}, \ldots, c_j\}, k-i]) & \text{otherwise} \end{cases} \tag{3.11}$$

Scenario 2: Partitioning with Contiguous Atomic Groups. Given the atomic groups $G = \{g_1, \ldots, g_l\}$, a common scenario is to partition them into K contiguous subsets. This could be specified as $K+1$ integers $1 = b_1 \le b_2 \ldots \le b_{K+1} = l$ where the boundary of the i-th subset is specified by $[b_i, b_{i+1}]$ and consists of a set of atomic groups $\{g_{b_i}, \ldots, g_{b_{i+1}}]$. This is often desirable when the underlying attribute has a natural ordering such as year. So we would prefer to train VAE models over data from consecutive years such as $\{2016-2017, 2018-2019\}$ instead of arbitrary groupings such as $\{\{2016, 2018\}, \{2017, 2019\}\}$. This problem could be solved in near linear time (*i.e.,* $O(l)$) by using the approach first proposed in [78]. The key insight is the notion of *sparse interval set system* that could be used to express any interval using a bounded number of sparse intervals. The authors then use a dynamic programming approach on the set of sparse intervals to identify the best partitioning.

In practice, K is often determined by various other factors such as space budget for persisting the generative models. Identifying K automatically is an interesting orthogonal problem. Our bounding function for R-ELBO has a natural monotonic property. We empirically found that common heuristics for selecting number of clusters such as Elbow method [79] works well for our purpose.

3.6 Experiments

We conduct a comprehensive set of experiments and demonstrate that VAE (and deep generative models) are a promising mechanism for AQP. We reiterate that our proposed approach is an alternate way for generating samples, albeit very fast. Most of the prior work for improving AQP estimates could be transparently used on the samples from VAE.

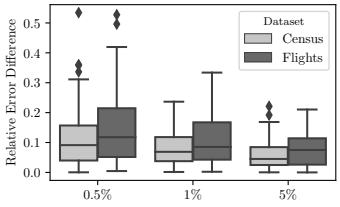

Figure 3.2: Varying Sample Size

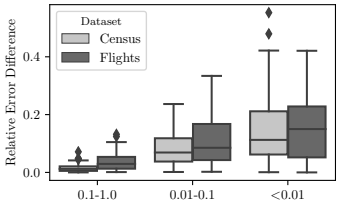

Figure 3.3: Varying Query Selectivity

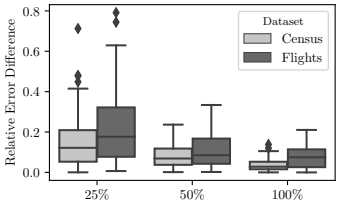

Figure 3.4: Varying Latent Dimension

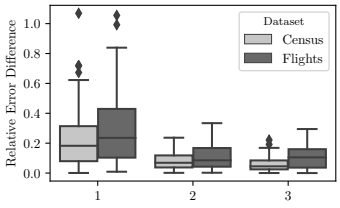

Figure 3.5: Varying Model Depth

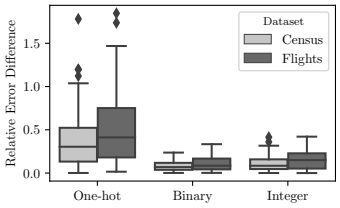

Figure 3.6: Varying Input Encoding

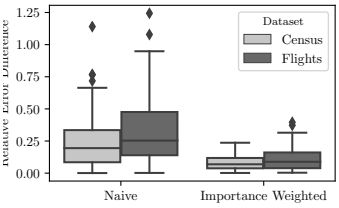

Figure 3.7: Varying Output Encoding

3.6.1 Experimental Setup

Hardware and Platform. All our experiments were performed on a server with 16 cores, 128 GB of RAM and NVidia Tesla K80 GPU. We used PyTorch [80] for training VAE and GAN, bnlearn [81] for learning Bayesian Networks and MSPN [82] for mixed sum-product networks (MSPN).

Datasets. We conducted our experiments on two real-world datasets: Census [83] and Flights [84, 85]. Both datasets have complex correlated attributes and conditional dependencies that make AQP challenging. The Census dataset has 8 categorical attributes and 6 numerical attributes and contains demographic and employment information. The Flights dataset has 6 categorical and 6 numerical attributes and contains information about on-arrival statistics for the last few years. We used the data generator from [85] to scale the datasets to arbitrary sizes while also ensuring that the relationships between attributes are maintained. By default, our experiments were run on datasets with 1 million tuples.

Deep Generative Models for AQP. In our experiments, we primarily focus on VAE for AQP as it is easy and efficient to train and generates realistic samples [62]. By default, our VAE model consists of a 2 layer encoder and decoder that are parameterized by Normal and Bernoulli distributions respectively. We used binary encoding (Section 3.4.5) for converting tuples into a representation consumed by the encoder.

In order to generate high quality samples, we use rejection sampling during both VAE training and sample generation albeit at different granularities. During training, the value of threshold $T(x)$ is set for each tuple x so that the acceptance probability of samples generated from $q(z|x)$ is roughly 0.9 for most tuples. We use the procedure from [69] to generate a Monte Carlo estimate for $T(x)$ satisfying acceptance probability constraints. While the trained model already produces realistic samples, we further ensure this by performing rejection sampling with a fixed threshold T (for the entire dataset) during sample generation (as detailed in Section 3.4.2). There are many ways for choosing the value of T. It could be provided by the user or chosen by cross validation such that it provides the best performance on query workload. By default, we compute the value of T from the final epoch of training as follows. For each tuple x, we have the Monte-Carlo estimate $T(x)$. We select the 90-th percentile of the distribution $T(x)$. Intuitively, this ensures that samples

generated for 90% of the tuples would have acceptance probability of 0.9. Of course, it is possible to specify different values of T for queries with stringent accuracy requirements. We used Wasserstein GAN as the architecture for generative adversarial networks [14]. We used entropy based discretization [27] for continuous attributes when training discrete Bayesian networks. We used the default settings from [82] for training MSPN.

Query Workload. We used IDEBench [85] to generate aggregate queries involving filter and group-by conditions. We then selected a set of 1000 queries that are diverse in various facets such as number of predicates, selectivity, number of groups, attribute correlation etc.

Performance Measures. As detailed in Section 3.4.2, AQP using VAE introduces two sources of errors: sampling error and errors due to model bias. The accuracy of an estimate could be evaluated by relative error (see Equation 3.1). For each query in the workload, we compute the relative error over a fixed size sample (1% by default) obtained from the underlying dataset R and the learned VAE model. For a given query, the relative error difference (RED) computed as the absolute difference between the two relative errors provides a meaningful way to compare them. Intuitively, RED will be close to 0 for a well trained VAE model. We repeat this process over 10 different samples and report the average results. Given that our query workload has 1000 queries, we use box plots to concisely visualize the distribution of the relative error difference. The middle line corresponds to the median value of the difference while the box boundaries correspond to the 25th and 75th percentiles. The top and bottom whiskers are set to show the 95th and 5th percentiles respectively.

3.6.2 Experimental Results

Evaluating Model Quality. In our first experiment, we demonstrate that VAE could meaningfully learn the data distribution and generate realistic samples. Figure 3.2 shows the

distribution of relative error differences for both datasets over the entire query workload for various sample sizes. We can see that the differences are less than 1% for almost all the cases for the Census dataset. The flights dataset has many attributes with large domain cardinalities which makes learning the data distribution very challenging. Nevertheless, our proposed approach is still within 3% of the relative error obtained from the samples of R.

Impact of Selectivity. In this experiment, we group the queries based on their selectivity and compute the relative error difference for each group. As shown in Figure 3.3, the difference is vanishingly small for queries with large selectivities and slowly increases for decreasing selectivities. In general, generating estimates for low selectivity queries is challenging for any sampling based AQP. The capacity/model size constraints imposed on the VAE model could result in generating bad estimates for some queries with very low selectivities. However, this issue could be readily ameliorated by building multiple VAE models that learn the finer characteristics of data minimizing such errors in these cases.

Impact of Model Capacity and Depth. Figures 3.4 and 3.5 shows the impact of two important hyper parameters - the number of latent dimensions and depth of the encoder and decoder. We vary the latent dimension from 10% to 100% of the input dimension. Large latent dimension results in an expressive model that can learn complex data distributions at the cost of increased model size and training time. Increasing the depth results in a more accurate model but with larger model size and slower training time. Empirically, we found that setting latent dimension size to 50% (for binary encoding) and encoder/decoder network depth of 2 provides good results.

Effectiveness of Input Encoding and Output Decoding. It is our observation that the traditional approach of one-hot encoding coupled with generating a single sample tuple for each sample from the latent space does not provide realistic tuples. It may be suitable for

image data but certainly not suitable for relational data. Figure 3.6 shows how different encodings affect the generated samples. For datasets such as Census where almost all attributes have small domain cardinality, all the three approaches provide similar results. However, for the flights dataset where some attributes have domain cardinality in tens of thousands, naive approaches such as one-hot encoding provides sub-optimal results. This is due to the fact that there are simply too many parameters to be learnt and even a large dataset of 1 Million tuples is insufficient. Similarly, Figure 3.6 shows that our proposed decoding approach dramatically decreases the relative error difference making the approach suitable for relational data. This is due to the fact that the naive decoding could produce unrealistic tuples that could violate common integrity constraints an effect that is minimized when using our proposed decoding.

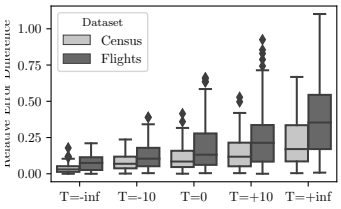

Figure 3.8: Varying T

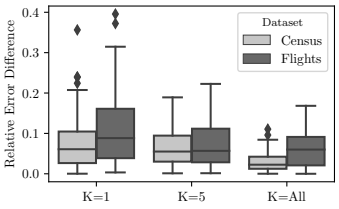

Figure 3.9: Varying K

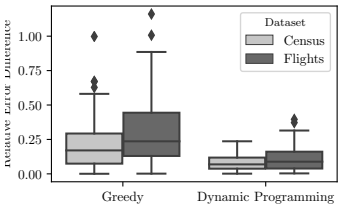

Figure 3.10: Partition Algorithms

Impact of Rejection Sampling. Figure 3.8 shows how varying the value of T impacts the sample quality. Recall from Section 3.4.2 that as $T \to +\infty$, almost all samples from VAE are accepted, while when $T \to -\infty$, samples are rejected unless they are likely to be from the true posterior distribution. As expected, decreasing the value of T results in decreased value of relative error difference. However, this results in a larger number of samples being rejected. Our approach allows T to be varied across queries such that queries with stringent

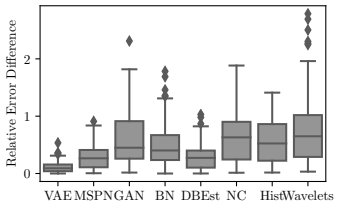

Figure 3.11: Performance of DL models for AQP

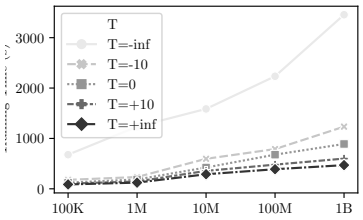

Figure 3.12: Performance of Model Building

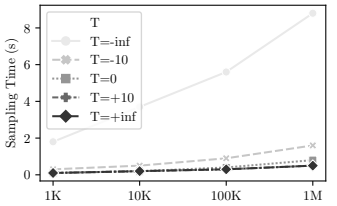

Figure 3.13: Performance of Sample Generation

accuracy requirements can use small T for better estimates. We investigate the impact of rejection sampling on model building and sample generation later in the section.

One versus Multiple VAEs. In the next set of experiments, we consider the case where one uses multiple VAEs to learn the underlying data distribution. We partitioned the attributes based on marital-status for Census and origin-state for Flights. We evaluated partitioning data over other attributes and observed similar results. In order to compare the models fairly, we ensured that the cumulative model capacity for both scenarios were the same. For example, if we built K VAE models with capacity C each, then we compared it against a single VAE model with capacity $K \times C$. Figure 3.9 shows the results. As expected, the sample quality improves with larger number of VAE models enabling them to learn finer data characteristics. Interestingly, we observe that increasing the model capacity for the single VAE case has diminishing returns due to the fixed size of the training data. In other words, increasing the capacity does not improve the performance beyond certain model capacity. Figure 3.10 compares the performance of partitions selected by the dynamic pro-

gramming algorithm for the scenario where an OLAP hierarchy is provided. We compare it against a greedy algorithm. As expected, our proposed approach that is cognizant of the R-ELBO metric provides better partitions - especially datasets such as Flight that have complex R-ELBO distributions.

3.6.3 Comparison with DL Model for AQP.

While we primarily focused on VAE, it is possible to leverage other deep generative models for AQP. Figure 3.11 compares the performance of three common models : VAE, GAN and Bayesian Networks (BN). Generative Adversarial Networks (GANs) [86, 14] are a popular and powerful class of generative models that learn the distribution as a minimax game between two components - generator (that generates data) and discriminator (that identifies if the sample is from the true distribution or not). (Deep) Bayesian networks (BN) are another effective generative model that specifies the joint distribution as a directed graphical model where nodes correspond to random variable A_i (for attribute A_i) and directed edges between nodes signify (direct) dependencies between the corresponding attributes. Please refer to [14] for more details. In order to ensure a fair comparison, we imposed a constraint that the model size for all three approaches are fixed. Furthermore, VAE provides the best results for a fixed model size. GANs provide reasonable performance but was heavily reliant on tuning. Training a GAN requires identifying an equilibria and tuning of many parameters such as the model architecture and learning rate [86]. This renders the approach hard to use in practise for general data sets. Identifying appropriate mechanisms for training GANs over relational data for AQP is a promising avenue for future research. BNs provide the worst result among the three models. While BNs are easy to train for datasets involving discrete attributes, a hybrid dataset with discrete and continuous attributes, and attributes with large domain cardinalities are challenging. When the budget on model size is strict, BNs often learn a sub-optimal model.

We also evaluated VAE against the recently proposed MSPN [82] that has been utilized for AQP in [75]. Similar to Bayesian Networks, MSPNs are acyclic graphs (albeit rooted graphs) with sum and product nodes as internal nodes. Intuitively, the sum nodes split the dataset into subsets while product nodes split the attributes. The leaf nodes define the probability distributions for an individual variable. MSPN could be used to represent an arbitrary probability distribution [82]. We used the random sampling procedure from [75] for generating samples from a trained MSPN. We observed that MSPN often struggles to model distributions involving large number of attributes and/or tuples and that using a single MSPN for the entire model did not provide good results. As a comparison to train a VAE on 1M tuples of the Census data set on all attributes requires a few minutes versus almost 3.5 hours for MSPN. In addition the accuracy of queries with larger number of attributes for the case of MSPN was very poor and not close to any of the other models. Hence, we decided to provide *an advantage* to MSPN, building the model over subsets of attributes. That way we let the model focus only on specific queries and improve its accuracy. There were around 120 distinct combination of measure and filter attributes in our query workload. We built MSPN models for each combination of attributes, generate samples from it and evaluate our queries over it. For example, if a query involved an aggregate over A_x and filter condition over $\{A_i, A_j, A_k\}$, we built an MSPN over the projected dataset containing only $\{A_x, A_i, A_j, A_k\}$. Unlike GAN and BN, we did not control the number of leaf nodes. However, the size of the MSPN models that were trained over attribute subsets were in the same ballpark as the other generative models. Figure 3.11 presents the performance of VAE and MSPN (build on specialized subsets of attributes) to be superior over GAN and BN. However, in the case of VAE the model was trained over the entire dataset being able to answer *arbitrary queries* while MSPN was trained over specific attribute subsets utilized by specific queries. Even in this case, providing full advantage to MSPN, the median relative error difference for VAE and MSPN were 0.060835 and 0.137699 respectively, more than

two times better for VAE. This clearly demonstrates that a VAE model can learn a better approximation of the data, being able to answer *arbitrary queries* while it can be trained an order of magnitude faster than MSPN as detailed next.

Next, we compare our approach with DBEst [87] and NeuralCubes [88] that use ML models for answer AQP queries. Figure 3.11 compares the performance of our approach against these methods. In contrast to our approach that uses synthetic samples, DBEst and NeuralCubes use pre-built models to directly answer AQP queries. For simple aggregate queries, the performance of both these methods are comparable to that of our approach. However, our approach produces more accurate result for ad-hoc queries that are very common in data exploration. Furthermore, the ability of our approach to create arbitrary number of samples to achieve low error that is not possible with DBEst and NeuralCubes.

Performance Experiments. Our next set of experiments investigate the scalability of VAE for different dataset sizes and values of threshold T. Figure 3.12 depicts the results for training over a single GPU. All results would be substantially better with the use of multiple GPUs. As expected, the training time increases with larger dataset size. However, due to batching and other memory optimizations, the increase is sublinear. Next, incorporating rejection sampling has an impact on the training time with stringent values of T requiring more training time. The increased time is due to the larger number of training epochs needed for the model to learn the distribution. The validation procedure for evaluating the rejection rate uses a Monte Carlo approach [69] that also contributes to the increased training time. However overall it is evident from our results that very large data sets can be trained very efficiently even on a single GPU. This attests to the practical utility of the proposed approach. Figure 3.13 presents the cost of generating samples of different sizes and for various values of T. Not surprisingly, lower values of T require a larger

sampling time due to the higher number of rejected samples. As T becomes less stringent, sampling time dramatically decreases. Interestingly, the sampling time does not vary a lot for different sampling sizes. This is due to the efficient vectorized implementation of the sampling procedure in PyTorch and the availability of larger memory that could easily handle samples of large size. It is evident again that the proposed approach can generate large number of samples in fractions of a second making the approach highly suitable for fast query answering with increased accuracy.

3.7 Related Work

Deep Learning for Databases. Recently, there has been increasing interest in applying deep learning techniques for solving fundamental problems in databases. SageDB [47] proposes a new database architecture that integrates deep learning techniques to model data distribution, workload and hardware and use it for indexing, join processing and query optimization. Deep learning has also been used for learning data distribution to support index structures [41], join cardinality estimation [10, 43], join order enumeration [46, 44], physical design [89], entity matching [49], workload management [90] and performance prediction [91].

Sampling based Approximate Query Processing. AQP has been extensively studied by the database community. A detailed surveys is available elsewhere [65, 60]. Non sampling based approaches involve synopses data structures such as histograms, wavelets and sketches. They are often designed for specific types of queries and could answer them efficiently. In our paper, we restrict ourselves to sampling based approaches [92, 93, 94, 95, 96]. Samples could either be pre-computed or obtained during runtime. Pre-computed samples often leverage prior knowledge about workloads to select samples that minimize the estimation error. However, if workload is not available or is inaccurate, the chosen sam-

ples could result in worse approximations. In this case, recomputing samples is often quite expensive. Our model based approach could easily avoid this issue by generating samples as much as needed on-demand. Online aggregation based approaches such as [97, 98] continuously refine the aggregate estimates during query execution. The execution can be stopped at any time if the user is satisfied with the estimate. Prior approaches often expect the data to be retrieved in a random order which could be challenging. Our model based approach could be easily retrofitted into online aggregation systems as they could generate random samples efficiently. Answering ad-hoc queries and aggregates over rare sub-populations is especially challenging [99] . Our approach offers a promising approach where as many samples as needed could be generated to answer such challenging queries without having to access the dataset. [75] uses mixed sum-product networks (MSPN) to generate aggregate estimates for interactive visualizations. While in the same spirit as our work, their proposed approach suffers from scalability issues that limits its widespread applicability. Even for a small dataset with 1 million tuples, it requires hours for training. This renders such an approach hard to apply for very large data sets. In contrast a VAE model can be trained in a matter of minutes making it ideal for very large data sets.

3.8 Conclusion

We proposed a model based approach for AQP and demonstrated experimentally that the generated samples are realistic and produce accurate aggregate estimates. We identify the issue of model bias and propose a rejection sampling based approach to mitigate it. We proposed dynamic programming based algorithms for identifying optimal partitions to train multiple generative models. Our approach could integrated easily into AQP systems and can satisfy arbitrary accuracy requirements by generating as many samples as needed without going back to the data. There are a number of interesting questions to consider in

the future. Some of them include better mechanisms for generating conditional samples that satisfy certain constraints. Moreover, it would be interesting to study the applicability of generative models in other data management problems such as synthetic data generation for structured and graph databases extending ideas in [100].

CHAPTER 4

Can Geometry Resolve Quantile Regression's Inefficiency?

Runtime complexity and robustness are two critical attributes of machine learning algorithms in today's world of big data. In this paper, we aim to investigate a classical machine learning/statistics problem, quantile regression, especially for database setups associated with billions of records. We utilize the computational geometry concept of Arrangement and Duality to design an algorithm named UPDATENEIGHBOR oracle which can calculate the objective function in $O(d)$ time from a neighboring point in the arrangement. We combine our UPDATENEIGHBOR oracle algorithm and a computational geometry concept of k-set to design a quantile regression algorithm in 2-dimension, which is asymptotically better than all other existing approaches. Our expected run time and deterministic runtime are better than the existing state of the art.

4.1 Introduction

In this paper, we revisit the classical problem of *Quantile Regression (QR)*, a robust alternative to the well-known *Ordinary Least Squares (OLS) Linear Regression*, and develop novel algorithm that is asymptotically faster than all the existing algorithms.

In the rest of the introduction, we first motivate the general need for speed and robustness of ML systems for big data. We then describe the specific problem of Quantile Regression, its challenges, and summarize our novel technical contributions.

4.1.1 Motivation

The modern era is blessed with an unprecedented abundance of data. Such rapidly growing data offers opportunities for machine learning (ML) techniques to harness knowledge at a scale that had not been possible before. However, the growth of data brings additional challenges like longer training time for ML models. One solution is increasing computational power. However, the available computational power to support these machine learning techniques is not increasing at the same rate as the growth in data. Before 2012, computational power needed to train artificial intelligence was doubling in every two years, but since then the need had doubled in every three-four months (i.e., approximately seven times per year) [101], mainly due to the increased cost of training and maintaining ML models on such ever increasing datasets. As a result, there is a continuous quest for algorithms with better run time complexity, especially in big data scenarios. Even a small improvement in runtime complexity can make a substantial difference in execution time for big data. In summary, there is a pressing need to develop *faster machine learning algorithms* for many application scenarios.

Besides a fast algorithm, *robustness* is another crucial requirement in machine learning [102, 103, 104, 105]. The research area of robust ML draws its inspiration from the classical field of *robust statistics*, where the canonical example is *mean* versus *median* estimators – the latter is less affected by outliers in the data compared to the former []. In modern machine learning, the notion of robustness has broadened to include ML models that are impervious to outliers, noise, variability between training and test distributions, adversarial corruptions in the training data, and various other notions. In certain applications, robustness can be a critical requirement. For example, recent works (see [106]) have shown through adversarial methods how models used in an autonomous vehicle trained to detect STOP signs lack robustness, which can be extremely devastating and life threatening. Smart health-care [107], data-driven policy making [108], and predictive policing [109] are

a few example domains underscoring the importance of developing ML models that are robust in practice. Recently, There has been a lot of ML-based work[110][111] in database research in recent years, which are replacing the traditional database techniques. However, it is crucial to have the right tools and techniques to ensure the robustness of such ML techniques.

4.1.2 Quantile Regression: Challenges, and State of Art

Linear Regression is a seminal statistical approach to building linear predictive models between a response (i.e., dependent) variable and one or more predictor (i.e., independent) variables. Linear regression is over two centuries old (dating back to Legendre and Gauss) and is often considered as the forerunner of modern machine learning [**?**]. We revisit two classical linear regression techniques: *Ordinary Least Square Regression* (OLS) and *Quantile Regression* (QR). Both build linear models, but with different objective functions: in OLS the objective is to minimize the mean squared error (i.e., ℓ_2 norm) between the dependent variable and that predicted by the model; whereas in QR, the objective is to minimize the mean absolute error (i.e., ℓ_1 *norm*).

Why Quantile Regression? Historically, OLS has been much more widely used than QR. This is primarily because (as we will discuss shortly) QR is plagued by unacceptably high computational resource needs, whereas there exist extremely resource efficient and scalable algorithms for building OLS models.

Nevertheless, despite its computational advantages, OLS has a major limitation. *OLS is not a robust model.* It can be easily skewed in the presence of outliers in the data, which makes OLS a poor model of choice in several emerging applications where robustness of the predictive model is critically needed. On the other hand, *QR is a robust model.* In simple terms, the comparison of OLS and QR is analogous to the "mean versus median"

comparison mentioned in Section 1.1. That is why QR is considered for applications where model robustness in the presence of outliers and skew in the data is critically needed, such as healthcare [112], ecology [113], and others [114]. Beyond robustness, another attraction of quantile regression is that it is advantageous when conditional quantile functions are of interest, which have application in uncertainty quantification and conformal prediction in AI systems as well as approximate query processing in databases [115, 116]. In fact, if the scalability and computational efficiency of QR could be significantly improved, its applicability would increase dramatically across an even more broad range of applications [113].

Computational Challenges of Quantile Regression: The computational challenges of QR is best highlighted by the following observations. Consider a database with n tuples and d attributes, of which one attribute is the response (i.e., independent) variable and the rest are predictor (i.e., dependent) variables. Typically $n >> d$ in most data fitting problems. For OLS, since the objective function to be minimized (least square) is quadratic and differentiable, this yields a highly scalable training algorithm with time complexity of $O(nd^3)$ time and space complexity of $O(d^2)$ [117]. In contrast, the objective function in quantile regression is defined by the ℓ_1 *norm, i.e., sum of the absolute error*, which has "piecewise linear" characteristics, and hence its optimization is much more challenging. The state of art approach for QR [118] applies a reparameterization technique to convert the problem into a huge linear programming formulation involving n linear constraints and $(n + 2d)$ variables. Solving these linear programs require significant more resources (time and memory) compared to OLS [118], making QR prohibitive for all but small-to-moderate sized datasets. This need for extensive computational resources has been one of the main reasons for the relative lack of adoption of QR in emerging Big Data applications.

State of Art Techniques and Tools: Quantile Regression is a well-studied research problem. In Section 4.6, we provide a more detailed overview of the literature, but highlight a few key works here.

Two prominent class of exact techniques to solve the QR problem are *exterior-point based* [119] and *interior-point based* [118] approaches. Barrodale and Robert (*BR*) proposed a simplex-based exterior point technique [119], which moves from one exterior point (i.e, a "corner" of the feasible polytope defined by the linear program) to another exterior point, in the direction of steepest gradient descent. The core idea of BR originates from Edgeworth's bi-variate weighted median based 1888 approach [120]. The run time of this algorithm is $O(n^2)$, which is still the state of art algorithm for 2-dimension. All the exterior points based approaches are only suitable for small problem instances with around 1000 tuples and few attributes [118]. Portnoy and Koenkar later proposed a primal-dual interior point method (*IPM*) which finds the optimal solution by minimizing the difference between primal and dual objective cost [118]. This method is reasonably fast for larger input sizes in practice but worst case theoretical runtime complexity is $O(n^{3.5} \log 1/\epsilon)$ where ϵ is the desired accuracy [121]. Moreover, BR and IPM are extremely memory hungry for big data, hence, not suitable for big data instances under resource constraints.

4.1.3 Our Technical Contributions

In this paper, we make two key technical contributions.

Our first contribution is designing an Oracle, named UPDATENEIGHBOR oracle, which efficiently finds the objective cost of an exterior point from its neighbor's aggregate information. We capitalize a computational geometry concepts of *arrangement* and *duality* [122] to define the neighborhood. The aggregate information contains the sum along every feature for current exterior point with both positive and negative errors. Our

UPDATENEIGHBOR oracle performs an update operation in $O(d)$ time and $O(d)$ space complexity for any input dimension d.

Our second major contribution is an efficient quantile regression algorithm for 2-dimension, named QREG2D. We use computational geometry concept like *arrangement*, *duality*, and k-*set* [122] to solve 2-dimensional quantile regression problem. The deterministic run time complexity for our QR algorithm is in $\mathcal{O}(n^{4/3}\log^{1+a} n)$ where where $a > 0$ is an arbitrarily small constant. The expected run time complexity of our algorithm is in $\mathcal{O}(n^{4/3})$ which is asymptotically better than any other existing approach in 2-dimension. However, our computational geometry based approach is not extendable in higher dimension which we will discuss later in Section 4.5.

Summary of Contributions:

- We investigate the classical machine learning and statistical technique of Quantile Regression and proposed novel approaches to solve it optimally and efficiently.
- We propose an innovative computational geometry based algorithm named UPDATENEIGHBOR oracle which can update the objective cost at an exterior point from its neighbor in $O(d)$ time and space complexity.
- We also design a novel algorithm named QREG2D utilizing UPDATENEIGHBOR oracle and computational geometry concepts like arrangement and k-set for quantile regression problem in 2-dimension. The deterministic version of our algorithm runs in in $\mathcal{O}(n^{4/3}\log^{1+a} n)$ time where where $a > 0$ is an arbitrarily small constant. However, our Algorithm has an expected run time complexity of $\mathcal{O}(n^{4/3})$ and is asymptotically superior than the existing $O(n^2)$ approach for Quantile regression in 2-dimension.

Paper Organization: In Section 4.2, we introduce the problem with all the necessary details to understand the problem and solutions in this paper. We present our UPDATENEIGH-

id	A_1	A_2
t_1	3.15	3.13
t_2	1.97	1
t_3	1.369	2.43
t_4	0.149	1.287
t_5	-0.39	0.222
t_6	-0.51	-0.65
t_7	-2.04	7.30

Table 4.1: A 2D dataset.

BOR oracle algorithm in Section 4.3, the quantile regression algorithm in 2-dimensional in 4.4, challenges in higher dimension in 4.5 and followed by related work Section.

4.2 Preliminaries

In this section we provide useful definitions and notations, as well as a formal description of the main problem considered in this paper. Some of the notations and formal problem definitions are borrowed from existing literature such as [119, 118].

4.2.1 Running Example

In this paper, we use an example dataset, shown in Table 4.1, which will be used throughout the article. The dataset contains seven input tuples and two attributes A_1 and A_2. our goal is to predict one of the attributes from the other attribute. For our example, let us assume that we want to predict A_2 from A_1. Let us denote $A1$ and $A2$ by x and y respectively for simplicity.

Figure 4.1 shows a scatter plot of input tuples in 2-dimensional space where each input tuple is a point with (x, y) coordinate. We get a line if we fit a *Linear Regression(LR)* model for these points. Figure 4.1 demonstrates two classical *LR* fits. One of the *LR* fits is *Ordinary Least Square Regression(OLS)*, which minimizes the sum of squared error. The

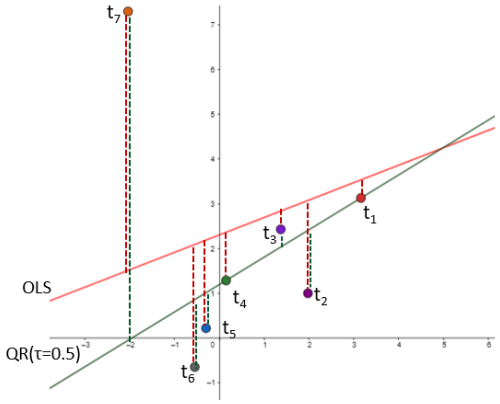

Figure 4.1: The Visual demonstration of database

other *LR* fit is a special type of *Quantile Regression(QR)* ($\ell 1$-*regression*) which minimizes sum of absolute error.

For any input point, the vertical distance between the point and the *LR* fit is the error in regression. In Figure 4.1, the red and green dotted line shows the error for *OLS* and ℓ_1-*regression* fits respectively. The figure shows that the *OLS* model is heavily affected by the presence of outlier t_7. However, the ℓ_1-*regression* is less sensitive to the outlier. For all but one point, the length of a green dotted line is smaller than that of a red dotted line.

4.2.2 Problem Definition

Data Model: Let $\mathcal{D}$ be a dataset with n tuples and d attributes. Our goal is to predict one of the attributes from rest of the attributes. The attribute we predict is called *response variable*. The attributes used in prediction is called *predictor variables*. For our running example, y (A_2) is response variable and x (A_1) is predictor variable. For d attributes, we denote the response variable and the vector of predictor variables by y_i and $X_i = [1\ x_1\ x_2 \ldots x_{d-1}]$ respectively for i^{th} tuple. A constant 1 is added in X_i to make the problem definition

(defined later) consistent. X is a matrix where i^{th} row corresponds to X_i. The X and y for the running example are following.

$$X = \begin{bmatrix} 1 & 50 \\ 1 & 24 \\ 1 & 12 \\ 1 & 5.5 \\ 1 & 2.4 \\ 1 & 1 \\ 1 & 0.05 \end{bmatrix} \; and, y = \begin{bmatrix} 0.6 \\ 0.55 \\ 0.45 \\ 0.35 \\ 0.28 \\ 0.2 \\ 0.12 \end{bmatrix}$$

Residual: If the actual value is y and the regressed value is $\hat{y}$, then the residual, r, can be defined as $y - \hat{y}$. The value of r can be positive, negative, and zero. From this view point, we can also define $r_i = r_i^+ - r_i^-$ where, for each tuple i in $\mathcal{D}$, $r_i^+ = max(r_i, 0)$ and $r_i^- = -min(r_i, 0)$. Note that r_i^+, and r_i^- contain only non-negative values. In Figure 4.1, the dotted lines shows the residuals. t_5 has positive residuals and t_7 has negative residuals.

Residual sets: Given a *QR* hyperplane with parameters β in the primal space, the set of tuples for which actual value y_i is greater than or equal to (resp. lesser than) the predicted value $\hat{y}_i = \beta^T X_i$ is denoted by $I^+ = \{i : y_i \geq \hat{y}_i\})$ (resp. $I^- = \{j : y_j < \hat{y}_j\}$). Note that the two sets are mutually exclusive, $I^+ \cap I^- = \emptyset$; $I^+ \cup I^- = \mathbb{U}$. Geometrically, the entity I^+ represents the set of points that lie on or above the regression hyperplane and I^- represents points that lie below the hyperplane.

Quantile Parameter, τ: In Quantile Regression, quantile parameter $\tau \in (0, 1)$ determines what fraction of total number of input tuples will have negative residual. For $\tau = 0.5$, half of the input tuple will have negative residuals and other half will have positive residuals. In practice, median, quartiles, deciles, 0.05, and 0.95 are some of the most frequently used quantile parameters.

Formal Problem Definition:

The formal definition of *Quantile Regression(QR)* problem in this paper is defined as follows.

Quantile Regression

Given quantile parameter τ and a dataset $\mathcal{D}$ with n tuples, where y is a vector of actual response variable and X is a $n \times d$ matrix constructed from predictor variables, find a parameter vector $\beta \in \mathbb{R}^d$ that optimizes the following objective function.

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1} \tau r_i^+ \ + \ (1-\tau) r_i^-$$

$$where\ r_i^+ = max(y_i - X_i^T \beta,\ 0)$$

$$and\ r_i^- = -min(y_i - X_i^T \beta, 0)$$

If $\tau = 0.95$, the objective function puts a high penalty (0.95) on any points with positive residuals and a low penalty (0.05) on any points with negative residuals. In Figure 4.1, we used $\tau = 0.5$ for the *QR* which puts equal penalty for both positive and negative residuals. For $\tau = 0.5$, *QR* is known as ℓ_1*-regression* which is often used as an alternative to *OLS*.

4.2.3 Geometric Mapping of Problem

Each input tuple $\mathcal{D}_i$ can be viewed as a point in d dimensional space. In this subsection, we define the *dual space* of $\mathcal{D}$, its connection with the quantile regression problem, and some of the basic properties that we will use throughout the paper.

***QR* Hyperplane, $\mathcal{H}$:** LR estimates the response variable as a linear combination of predictor variables with an offset. There is a geometric interpretation of an LR fit. If we assume the input tuple as points in space, the LR fit is a hyperplane in space. We call such

a hyperplane *QR* Hyperplane and denote it by $\mathcal{H}$. As shown in Figure 4.1 , the regression hyperplanes are lines for our running example in 2-dimension. The coefficients associated with the *QR* Hyperplane are regression parameters (β).

Optimal QR Hyperplane, $\mathcal{H}^*$: In primal space, there is a *QR* hyperplane for which *QR* optimization function is minimum. We call this hyperplane *Optimal QR Hyperplane* and denote it by $\mathcal{H}^*$. The *Optimal QR Hyperplane* geometrically divides the points into τn points on one side and $(1-\tau)n$ points on the other side [123].

Dual space: A duality transformation function transforms a given point (hyperplane resp.) in primal space into a hyperplane (point resp.) in the dual space, such that certain properties are maintained in the dual space and vice-versa. Following along similar lines as Edgeworth [120], we define the duality transformation function.

Given a hyperplane $\mathcal{H}$ in d dimensional primal space, the dual of $\mathcal{H}$ is a point $F(\mathcal{H}) \in \mathbb{R}^d$ such that:

$$\mathcal{H} : y - a_1 x_1 - a_2 x_2 \ldots - a_{d-1} x_{d-1} - a_d = 0$$
$$\mathcal{F}(\mathcal{H}) : \{a_1, a_2, \ldots, a_d\} \in \mathbb{R}^d$$

Given a point p in d dimensional primal space, the dual of the point is a d-dimensional hyperplane $\mathcal{F}(p)$ such that:

$$p : \{p_1, p_2, \ldots, p_d\} \in \mathbb{R}^d$$
$$\mathcal{F}(p) : z_d + \sum_{i=1}^{d-1} p_i z_i - p_d = 0$$

Here z_i represents the variables defining the hyperplane in dual space. We would like to note that the n points in the primal space transform into n corresponding hyperplanes in the dual space.

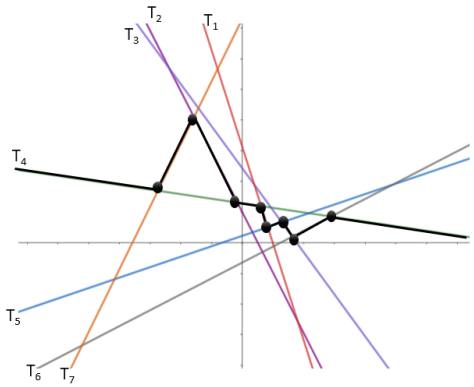

Figure 4.2: Dual space of the Sample Database

4.2.4 Arrangements and k-Sets

One of the contributions of our paper is the connection of computational geometry notions of *arrangements* and k-sets with the seemingly unrelated problem of quantile regression. We briefly review these concepts below.

Arrangements: Given n hyperplanes in d-dimensional space, the entire space is divided into an *arrangement* consisting of $O(n^d)$ d-dimensional convex *cells*. Much is known about the geometric properties of such arrangements, as well as algorithms for their construction. [122].

In our case, the n points in the primal space transform into n dual hyperplanes, creating a natural division of the dual space into such an arrangement, which provide us with an alternate view of the problem. Figure 4.2 shows the lines in the dual space for the sample database. The colored lines in the database represent the dual lines, with the color representing the tuple's color in the database.

Complete Skeleton, S: The relative location of a point p based on a hyperplane can be three types: above the line, below the line, and on the line which can be denoted by $+1$, -1 and 0 respectively. Given a set of n hyperplanes H, relative locations for all the hyperplanes in H can be found for p and a vector named *position vector* $U(p)$ can be constructed using

those n relative locations. For another point q, p and q belongs to an equivalence class defined as face if $U(q) = U(p)$. A face is a set of points such that all points in the face have exactly same position vector. A face is called a k-face if its dimension is k A 0-face is called vertex and an 1-face is called edge. The intersection of d hyperplanes creates a vertex. In Figure 4.2, the intersection between dual lines are vertices and line segments between two vertices are edges.

let E is a subset of edges in the arrangement $\mathcal{A}$. The end points of the edges are vertices. let V be a set of all vertices from E. A graph $S_E = (V, E)$ is called *skeleton* if S_E is connected. If a skeleton contains all the edges in $\mathcal{A}$, it is called *complete skeleton*. The details about the complete skeleton can be found in [122].

The dual hyperplanes, transformed from primal input points, create an arrangement. Any vertex in complete skeleton of such arrangement is a ***QR*** hyperplane in primal space. If we traverse all the vertices in complete skeleton, the optimal ***QR*** hyperplane can be found from these vertices. In this paper, we design efficient techniques to solve ***QR*** by traversing the complete skeleton smartly.

Neighbor: Let $S = (V, E)$ be the complete skeleton of a given arrangement where V is the set of vertices and E is the set of edges. Two vertices $v_1 \in V$ and $v_2 \in V$ are neighbors if there is the edge e between v_1 and v_2 in E. A vertex is created from the intersection of d dual hyperplanes. if we drop any of d dual hyperplanes, the intersection of remaining hyperplanes will be a dual line. As a dual line is an one-dimensional entity, there can be at most two neighbors of a vertex along a dual line. As we can drop any of d dual hyperplanes, there can be d dual lines going through a vertex. As a result, the total number of neighbors of a vertex is at most $2d$.

k-set: Given a set of n points in d-dimensional space, a k-set is a subset of k points that can be separated by a hyperplane from the remaining points. A lot is known about k-sets,

their counts, as well as algorithms for their enumeration and construction [122]. For our geometric mapping, consider the n points is given by the data points in the primal space. As the optimal quantile regression hyperplane $\mathcal{H}^*$ must partition the n points such that τn points are on one side and the remaining $(1-\tau)n$ lie on the other, the optimal hyperplane is a separating k-set hyperplane. We exploit these observations in our algorithm for both 2 and higher dimensions.

Further details of duality, arrangements, skeleton and k-sets can be found in [122].

4.2.5 Existing State-of-Art Exact Approaches

In this subsection, we describe a few properties of *QR*, connection to linear programming, and existing exact techniques to solve problem.

There has been extensive research conducted on *QR* over the last two hundred years. Many important properties of quantile regression have been identified[124]. The description of two important *QR* properties used in this paper is following.

Theorem 1. *[124] Quantile Regression objective function is continuous and convex.*

Theorem 2. *[124] The* Optimal QR Hyperplane *goes through at least d input tuples.*

Wagner [125] identified the connection between linear programming and $\ell 1$-*regression*. A reparameterization technique[123] is used to convert the *QR* problem into a linear programming problem. The linear constraints of a linear programming problem creates a convex polytope. *QR* problems solutions revolve around the utilization of the exterior(corner) and interior points of that convex polytope.

Next we give a brief overview of prior algorithms for *QR*. The focus of *QR* research has been primarily on how to make the algorithms more faster, with the eventual goal of trying to make it an alternative to OLS regression. There have been two broad category of techniques, exact techniques (where the objective function is minimized), and approximate/heuristic techniques. There is a wide body of techniques that belong

to the latter category, e.g., modifying the original objective function to achieve a faster performance[126, 127], as well as other approximation approaches [128, 129, 130]. However, since our focus in this paper is to consider regression as a robust and trustworthy technique, we are only interested in exact approaches.

Prior research on exact quantile regression techniques can be divided into two categories: *exterior point based approaches* and *interior point based approaches*.

In 1888, Edgeworth designed a computational geometry based exterior point approach [120] to solve 2-dimensional ℓ_1-regression. Edgeworth's technique with randomized median finding algorithm can solve 2-dimensional QR in $O(n^2)$ time. Barrodale and Roberts (BR)[119] designed a simplex algorithm-based approach for any dimension that starts from an exterior point, and moves towards the highest gradient descent of the objective function. The *quantreg*[131] package in R includes the implementation of BR in their package, which we named *QR-BR*. Unfortunately, *QR-BR* is extremely slow and memory inefficient for large datasets.

For large datasets, interior point based solutions are more appropriate. Portnoy and Koenkar [118] introduced a primal dual interior point method (*IPM*) solution for *QR*. In this approach, the *QR* problem is solved using primal and dual space both simultaneously. *IPM* adds a barrier function in the objective function which helps exploring the the solutions in interior points. The algorithm stops when the difference of primal and dual objective is smaller than a given small threshold ϵ. The *quantreg* package includes the latest *IPM* based implementation which we call *QR-IPM*. Although, *QR-IPM* is quite fast, it is extremely memory hungry.

Notation	Description
$\mathcal{D}$	Dataset
n	The number of input tuples in dataset
d	the number of attributes in dataset
τ	Quantile parameter
β	Vector of regression parameter
t	Input tuple or data points in primal space
$\mathcal{H}$	QR Hyperplane in primal space
$\mathcal{H}^*$	Optimal QR Hyperplane in primal space
r	Residual, the error in prediction.
r^+	Positive residual
r^-	Negative residual
I^+	A set of indices of input tuples with positive residual
I^-	A set of indices of input tuples with negative residual

Table 4.2: Important Notations

4.3 UPDATENEIGHBOR oracle

The exterior point based approaches to solve QR often move from one exterior point to a neighboring exterior point such that the objective cost decreases. Note that each exterior point has an optimization function value associated with it. While moving from one exterior point to a neighboring point, the objective function is often recalculated from scratch by traversing all the input points which is expensive and requires $O(nd)$ time. In this section, we propose an innovative technique UPDATENEIGHBOR oracle which can efficiently calculate the objective cost by maintaining only a few *aggregate values*. Our approach reduces the time complexity to $O(d)$ for any dimension d.

In the following subsection, we illustrate our idea of UPDATENEIGHBOR oracle with an example. We then introduce UPDATENEIGHBOR oracle formally to update the optimization function value in $\mathcal{O}(d)$ time when moving from vertex in the complete skeleton graph to its neighbor.

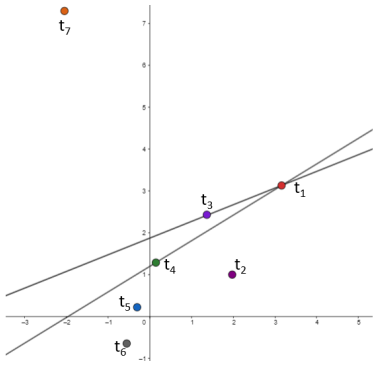

Figure 4.3: Update Operation

4.3.1 Illustration of UPDATENEIGHBOR oracle

In this subsection, we illustrate how we can reduce the computational cost while moving from one vertex to a neighboring vertex by maintaining only a few aggregate values. For this illustration of UPDATENEIGHBOR oracle, we use our running example from Table 4.1 with $\tau = 0.5$.

As seen in Section 4.2, a vertex in the complete skeleton graph (point in dual space) refers to a QR hyperplane in primal space. The optimal solution lies in one of these vertices in the complete skeleton graph. Let S be a complete skeleton graph constructed from Figure 4.2 and $P_{(i,j)}$ be any vertex constructed from the intersection of dual line T_i and T_j. As shown Figure 4.2, $P_{(1,3)}$ and $P_{(1,4)}$ are neighbors which will be used in our illustration. Figure 4.3 shows the primal representation of $P_{(1,3)}$ and $P_{(1,4)}$ where each represents a line. From the definition of residual set in Section 4.2, $I^+ = \{1, 3, 4, 7\}$ and $I^- = \{2, 5, 6\}$ for $P_{(1,4)}$. We calculate the sum along X-axis and Y-axis for these two group of points and call these aggregate values at $P_{(1,4)}$. Given these aggregate values for $P_{(1,4)}$, we would like to answer following questions as steps of our demonstration.

1. How to calculate the objective value at $P_{(1,4)}$?
2. How to calculate the aggregate values at $P_{(1,3)}$ from $P_{(1,4)}$ in $O(1)$?
3. If aggregate values are known for $P_{(1,3)}$, how to calculate the objective value at $P_{(1,3)}$?

As shown in Figure 4.3, $P_{(1,4)}$ passes through input tuple t_1 and t_4 in primal space and its parameter vector is $\beta = [1.19, 0.71]$.

For all $i \in I^+$, let $\sum x_i$, $\sum y_i$ be the sum along X and Y axis. Similarly, for all $j \in I^-$, let $\sum x_j$, and $\sum y_j$ be the sum along X and Y axis.

Given the aggregate values for $P_{(1,4)}$, the objective cost $\mathcal{L}$ at $P_{(1,4)}$ can be calculated using following process.

$$\begin{aligned}
\sum r_i &= (\sum y_i - \sum x_i \times \beta[1] - \beta[0] \times |I^+|) \\
&= 7.52 \\
\sum r_j &= (\sum x_j \times \beta[1] + \beta[0] \times |I^-| - \sum y_j) \\
&= 3.79 \\
\mathcal{L} &= 0.5 \times \sum r_i + (1 - 0.5) \times \sum r_j \\
&= 5.65
\end{aligned}$$

Now, we will try to answer how we will update the aggregate values at $P_{(1,3)}$ from $P_{(1,4)}$. As shown in Figure 4.3, only one input point t_4 changes its residual sign from positive to negative in $P_{(1,4)}$ to $P_{(1,3)}$ transition. For $P_{(1,3)}$, $I^+ = \{1, 3, 7\}$ and $I^- = \{2, 4, 5, 6\}$. If we want to calculate the aggregate values at $P_{(1,3)}$ from $P_{(1,4)}$, the information of input point $t4$ can be utilized in the following fashion.

$$\begin{aligned}
\sum y_i &= \sum y_i - t_4.y = 12.86 \\
\sum y_j &= \sum y_j + t_4.y = 1.86 \\
\sum x_i &= \sum x_i - t_4.x = 2.47 \\
\sum x_j &= \sum x_j + t_4.x = 1.27
\end{aligned}$$

Earlier we have shown how to calculate objective cost from aggregate values at $P_{(1,4)}$. Similarly objective cost for $P_{(1,3)}$ can be calculated using the aggregate values. If the aggregate values at $P_{(1,4)}$ is known, the aggregate values and objective cost at $P_{(1,3)}$ can be

calculated in $O(1)$ time and space. Although our illustration is in 2-dimension, our UPDATENEIGHBOR oracle works in any dimension.

4.3.2 UPDATENEIGHBOR oracle

Our first contribution is an Oracle which, when provided with a input vertex along with its optimization function value and a neighboring vertex, updates the optimization function value in $\mathcal{O}(d)$ time. The update operation relies on maintaining a few aggregate values when moving from one vertex to its neighboring vertex. To understand the motivation behind the aggregate values, we rewrite the optimization function into two summations.

$$\min_{\beta \in \mathbb{R}^d} \tau \sum_{i \in I^+} (y_i - X_i\beta) \; + \; (1-\tau) \sum_{j \in I^-} (X_j\beta - y_j)$$

First, we will show that if we are given certain aggregate values for a vertex, we can compute the optimization function in $\mathcal{O}(d)$ time. For all $i \in I^+$ and $j \in I^-$, consider the aggregate values $\sum y_i$, $\sum X_i[1], \ldots, \sum X_i[d]$, $\sum y_j$ and $\sum X_j[1], \ldots, \sum X_j[d]$ were known for a given vertex in the skeleton graph (hyperplane in primal space). Given these aggregate values, the optimization function can be computed for the corresponding primal hyperplane in $\mathcal{O}(d)$ time using the following formula.

$$\begin{aligned} \tau \sum_{i \in Y^+} y_i - \; \tau \sum_{1 \le m \le d} \big(\beta_m \sum_{i \in Y^+} X_i[m] \big) + \\ (1-\tau) \sum_{1 \le m \le d} \big(\beta_m \sum_{j \in Y^-} X_j[m] \big) - (1-\tau) \sum_{j \in Y^-} y_j \end{aligned} \tag{4.1}$$

Secondly, we show that given a vertex and the aggregate values for it, the aggregate values can be updated/computed for any neighboring vertex in $\mathcal{O}(d)$ time. An illustration to bring this important result is provided in Section 4.3.1.

The formal proof for this $\mathcal{O}(d)$ time update is presented in Theorem 3.

Theorem 3. *Given the aggregate values* $\forall_{1\leq m\leq d} \sum_{i\in I^+} X_i[m]$, $\sum_{i\in I^+} y_i$, $\sum_{j\in I^-} y_j$, $\forall_{1\leq m\leq d} \sum_{j\in I^-} X_j[m]$ *for a vertex in the complete skeleton graph, the aggregate values can be updated in* $\mathcal{O}(d)$ *time when we move to a neighboring vertex.*

Proof. Any vertex in the complete skeleton graph represents a hyperplane in the primal space which divides the set of points in $\mathcal{D}$ into two sets, I^+ and I^-. When we move from a vertex to its neighboring vertex the sets I^+ and I^- change in one of three ways,

- A point in primal space moves from above the hyperplane to below the hyperplane i.e. a point moves from I^+ to I^-.
- A point in primal space moves from below the hyperplane to above the hyperplane i.e. a point moves from I^- to I^+.
- The hyperplane moves such that a point from above is exchanged with a point from below. In such a case, the number of points above the hyperplane remain the same, i.e. a point from I^+ is swapped with a point in I^-. Note that the k in this case is may not be $\tau(1-n)$.

For each of the three cases, we prove that the aggregate values can be updated in $\mathcal{O}(d)$ time. For the rest of this proof, let the new sets after the update be represented by U^+ and U^-.

Let us consider the first case, where the hyperplane moves such that a point moves from the set I^+ to I^- i.e. a point in primal space which was above the hyperplane (vertex) now lies below the neighboring hyperplane (neighbor vertex). Let X_t be the point that is involved in the transition. As we know the details of point X_t, we can remove the

contribution of X_t towards I^+ and add the contribution to aggregates entities of I^-. The formulae for update are as below,

$$\sum_{i \in U^+} y_i = \sum_{i \in I^+} y_i - y_t$$
$$\forall_{1 \leq m \leq d} \quad \sum_{i \in U^+} X_i[m] = \sum_{i \in I^+} X_i[m] - X_t[m]$$
$$\sum_{i \in U^-} y_i = \sum_{i \in I^-} y_i + y_t$$
$$\forall_{1 \leq m \leq d} \quad \sum_{i \in U^-} X_i[m] = \sum_{i \in I^-} X_i[m] + X_t[m]$$

As there are $2d + 2$ equations which take $\mathcal{O}(1)$ time to update, the total time taken for the update of the aggregate values is $\mathcal{O}(d)$.

The second case, where a point in primal space has moved from below the hyperplane (vertex) to above the neighboring hyperplane (neighbor vertex), can be updated using a similar manner. Let X_t be the point that is involved in the transition. The formulae for the update operation for the second case are,

$$\sum_{i \in U^+} y_i = \sum_{i \in I^+} y_i + y_t$$
$$\forall_{1 \leq m \leq d} \quad \sum_{i \in U^+} X_i[m] = \sum_{i \in I^+} X_i[m] + X_t[m]$$
$$\sum_{i \in U^-} y_i = \sum_{i \in I^-} y_i - y_t$$
$$\forall_{1 \leq m \leq d} \quad \sum_{i \in U^-} X_i[m] = \sum_{i \in I^-} X_i[m] - X_t[m]$$

With $2d + 2$ equations each of which take $\mathcal{O}(1)$ the overall time complexity is $\mathcal{O}(d)$ time.

Let us consider the third case, where the hyperplane moves such that a point from above is exchanged with a point from below. In this case, a point from I^+ is swapped with a point in I^-. Let X_s be the point that is moved from I^+ to I^- and X_t be the point moved from I^- to I^+. As we know the details of point X_s and X_t, we can remove the contribution

of X_s towards I^+ and add the contribution of X_t to it. We perform vice verse operation to I^-. The formulae for update are as below,

$$\sum_{i \in U^+} y_i = \sum_{i \in I^+} y_i + y_t - y_s$$

$$\forall_{1 \leq m \leq d} \quad \sum_{i \in U^+} X_i[m] = \sum_{i \in I^+} X_i[m] + X_t[m] - X_s[m]$$

$$\sum_{i \in U^-} y_i = \sum_{i \in I^-} y_i - y_t + y_s$$

$$\forall_{1 \leq m \leq d} \quad \sum_{i \in U^-} X_i[m] = \sum_{i \in I^-} X_i[m] - X_t[m] + X_s[m]$$

With $2d + 2$ equations each of which take $\mathcal{O}(1)$ the overall time complexity is $\mathcal{O}(d)$ time. Hence, proved. □

4.4 Quantile Regression in 2 Dimension

In this section, we start with describing two approaches for the QR problem in 2 dimensions which will provide an intuition for our optimized two dimensional algorithm. We then propose our theoretical algorithm for 2 dimensional QR problem and prove its running time.

4.4.1 Algorithm based on neighbor exploration

In this subsection, we introduce an algorithm that utilizes the UPDATENEIGHBOR oracle.

Exploring the vertices in the complete skeleton graph presents us with an interesting algorithm to obtain the optimal *QR* line. Let us consider for simplicity that a GETNEIGHBORS oracle exists that can quickly provide us with the neighbours of a given vertex. We can start from any of the $\mathcal{O}(n^2)$ vertices and explore neighboring vertices around it in a Breadth First Search manner while updating the aggregate entities and objective cost.

While for simplicity of understanding we choose BFS for exploration, it does not change the analysis that follows. We stop when no new neighboring vertices are found with a lower optimization value. The optimal QR hyperplane is the vertex with the smallest optimization function among the explored intersection points.

Note that one naive way of constructing a GETNEIGHBORS Oracle is to compute the $\mathcal{O}(n^2)$ vertices of the complete skeleton graph beforehand. Computing and storing the intersections points consumes $\mathcal{O}(n^2)$ space and $\mathcal{O}(n^2)$ time. Often the $\mathcal{O}(n^2)$ space is prohibitive in nature/practice. As the dimensions grows beyond 2, this neighborhood based approach is prohibitive as there are a total of $\mathcal{O}(n^d)$ points. Exploring these intersection points is far more expensive than the LP approach for larger dimensions.

This algorithm provides us with an approach that explores the neighbors of vertices until reaches the optimal solution. While this approach produces similar complexity as that of exterior point method for 2 dimensions, it produces an intuition for our optimized 2D algorithm. We present a more efficient approach for 2 dimensions in the later part of this section.

4.4.2 k-set Exploration Algorithm

The seemingly disconnected concept of arrangement is connected to the problem of *QR* through the concept of k-sets. k-set is a subset containing k points that can be separated from the rest of the $n-k$ points by a hyperplane. k-sets have often been used in various computational geometry settings to solve numerous problem.

An interesting observation that can be made on the optimal solution QR hyperplane is that it separates the n points/tuples into τn and $(1-\tau)n$ points[**?**]. While there are many hyperplanes that separate the n points into two parts (with τn and $(1-\tau)n$ points), we are interested in that specific hyperplane which provides the lowest optimization score. As k-sets separate the points into k and $n-k$ points, the optimal hyperplane must be one

among of the k-set separating hyperplanes such that $k = \tau n$. The computational geometry technique of k-set can also be viewed as a walk along the k-level of an arrangement. Hence, the optimal hyperplane can be found along the vertices encountered during the walk in k-level of the arrangement.

The *QR* problem in 2 dimensions can utilize this property (the result of Theorem 2). The optimal hyperplane $\mathcal{H}^*$ in primal space is represented by a point $\mathcal{F}(\mathcal{H}^*)$ in dual space. This point is one of the vertices in the complete skeleton graph corresponding to the k-level arrangement in the dual space. One approach to solve this problem would be enumerate the vertices corresponding to the k-level of the arrangement and for each vertex compute the optimization function value.

We first present a k-set exploration approach that makes use of the concept of k-sets without the use of aggregate entities to solve the problem. Such an approach has to compute the optimization function value (error) for a vertex which corresponds to a dual point p in the k-level arrangement. Computing the optimization function for each of these vertices involves aggregating the error contribution of each of the n points in the primal space using the line $\mathcal{F}^{-1}(p)$. Note that each of the computation consumes $\mathcal{O}(n)$ time. For each of the vertices corresponding to the dual points in the k-level arrangement, the naive error computation is repeated and the optimal line l^* corresponds to that line $\mathcal{F}^{-1}(p)$ with the least error.

Time complexity: For a given vertex p in the k-level arrangement, a naive approach to compute the error without making use of UPDATENEIGHBOR oracle takes a total of $\mathcal{O}(n)$ time. The upper bound on the intersection points in k-level of an arrangement is bounded by $\mathcal{O}(nk^{1/3})$. Hence, the total time taken for the naive approach is $\mathcal{O}(n^2k^{1/3})$ for a given τ ($k = \tau n$). As $k = n/2$ when $\tau = 1/2$, the worst case time complexity is $\mathcal{O}(n^{7/3})$.

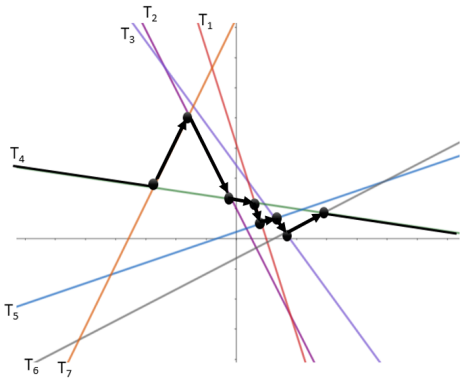

Figure 4.4: Traversing k-level of arrangement in order

4.4.3 Our 2D approach

The two approaches introduced so far set the base for our optimized approach. While the neighbor exploration algorithm that makes use of UPDATENEIGHBOR oracle has an optimized manner of exploration, the search space is not limited to k-sets. On the other hand, the main bottleneck in the k-set based approach without UPDATENEIGHBOR oracle is the error computation. For each vertex belonging to k-set, $\mathcal{O}(n)$ time is consumed for the computation of the optimization function.

Instead, we use a hybrid approach where the $\mathcal{O}(1)$ time update operation of computing optimization function when travelling between neighboring points in the k-level arrangement (k-set) overcomes the bottleneck in these approaches.

One property that our 2 dimensional algorithm relies upon is that the k-level arrangement enumerating algorithm explores the intersection points in order. That is for the Theorem 3 to be used we need to explore intersection points such that we move from one vertex in the complete skeleton to its neighbor vertex in the k-level arrangement. More specifically, we require the k-set enumerating algorithm to provide us vertices such that any vertex v_j obtained after v_i is its neighbor. Note that the neighboring vertices both be-

Algorithm 2 QREG2D

Input: Database $\mathcal{D}$, τ

Output: Parameters for optimal line $\mathcal{H}^*$

$k \leftarrow (1-\tau)n$

$Yp = Ym = Xp = Xm = 0$ {Initialize aggregate sums to 0. Yp: $\sum y_i$, Xp: $\sum X_i$ for $i \in I^+$
Ym: $\sum y_j$, Xm: $\sum X_j$ for $j \in I^-$}

$OptValue \leftarrow \infty$; $OptLine \leftarrow \emptyset$

$init \leftarrow False$

Obtain k-level arrangement points $\mathcal{P}$ using Randomized Incremental algorithm

for p in $\mathcal{P}$ **do**

  $l \leftarrow \mathcal{F}^{-1}(p)$ $\{y = mx + c \implies l \leftarrow [m, c]\}$

  if $init == False$ **then**

    $init = True$

    Calculate the aggregate values Ym, Xm, Yp, Xp using l

  else

    Find out X_s and X_t that swapped from I^+ and I^- respectively

    Update Ym, Xm, Yp, Xp using UPDATENEIGHBOR oracle

  $LineValue \leftarrow \tau(Yp - Xp \times l[1] - l[0] \times k) + (1-\tau)(Xm \times l[1] - l[0] \times (n-k) - Ym)$

  if $LineValue < OptValue$ **then**

    $OptValue \leftarrow LineValue$; $OptLine \leftarrow l$

return $OptLine$

long to the k-set. Figure 4.4 shows the ordered sequence of k-set in k level of arrangement for our running example.

Any k-set enumeration algorithm that satisfy this property can be used in our algorithm. The best deterministic approach to enumerate ordered k-set is using Sweep Line [132] algorithm by Edelsbrunner and Welzl using the dynamic data structure from T. Chan [133]. The runtime complexity of this deterministic approach is $O(n \log m + m \log^{1+a} n)$ where m is the number of k-set and $a > 0$ is an arbitrarily small constant. Additionally, T. Chan [134] has developed a Randomized Incremental algorithm to perform the enumeration in $O(m + n \log n)$.

Our approach is presented in QREG2D Algorithm 2. We initialize the aggregate values Xp, Yp, Xm and Ym to 0. Note that Xp and Yp represent the aggregate $\sum X_i$ and $\sum y_i$ for $i \in I^+$. Similarly, Xm and Ym represent the aggregate $\sum X_j$ and $\sum y_j$ for $j \in I^-$. We compute the aggregate values for the first point in the k-set as a special case. As we move from one point to its neighboring point in the k-set we call UPDATENEIGHBOR oracle to update the aggregate values. Objective cost can be computed using the aggregate values. We return the k-set with the minimum objective cost in the walk over the k level of arrangement. The runtime complexity of our algorithm with deterministic k-set enumeration is presented in Theorem 4. Theorem 5 proves the expected runtime of our algorithm.

Theorem 4. *Algorithm 2 finds the optimal solution for the 2 dimensional* QR *problem in* $\mathcal{O}(n^{4/3} \log^{1+a} n)$ *time where* $a > 0$ *is an arbitrarily small constant.*

Proof. For the first k-set obtained through the enumeration, the optimization function and aggregate values need to be calculated by a linear scan over the points which takes $\mathcal{O}(n)$ time. Updating the optimization function value and aggregate values as we explore neighboring k-set takes $\mathcal{O}(1)$ time. The points in the k-level of the arrangement can be computed using Sweep Line algorithm [132, 133, 134] in $O(n \log m + m \log^{1+a} n)$ where m is the number of k-set and $a > 0$ is an arbitrarily small constant. The upper bound on total

number of k-sets is given by Dey [135], $\mathcal{O}(nk^{1/3})$. This brings the overall time taken to $\mathcal{O}(nk^{1/3}\log^{1+a} n)$. As k is a percentage of n, the overall time complexity $\mathcal{O}(n^{4/3}\log^{1+a} n)$. Hence, proved. □

Theorem 5. *Algorithm 2 finds the optimal solution for the 2 dimensional* QR *problem in* $\mathcal{O}(n^{4/3})$ *expected time.*

Proof. Proof is similar to the proof of Theorem 4. However, if we use the Randomized Incremental algorithm instead of the deterministic algorithm, the enumeration can be done in $\mathcal{O}(n\log(n) + nk^{1/3})$ expected time (Corollary 4.4 [134]). The points in the k-level of the arrangement can be computed using Randomized Incremental algorithm[134] in $\mathcal{O}(n\log(n) + nk^{1/3})$ time (Corollary 4.4). This brings the overall time taken to $\mathcal{O}(n^{4/3})$. □

The run time complexity for IPM is $O(n^{3.5}\log 1/\epsilon)$ [121]. For exterior point based exact approach, the run time complexity for 2 dimension is $O(n^2)$ [124]. The worst case run time of complexity Algorithm 2 is, $\mathcal{O}(n^{4/3}\log^{1+a} n)$, which is better than the current exact approaches. However, the expected runtime our algorithm is $\mathcal{O}(n^{4/3})$ which is a lot better than the current state-of-the-art exact methods.

4.5 Challenges in Higher Dimension

Although our k-set based QR solution is extremely efficient in 2-dimension, the same idea does not flourish in higher dimension. Let us consider 3 dimension QR first. In 3 dimension, the upper bound on number of k-set is $O(nk^{\frac{3}{2}})$. As k is a fraction of n, the upper bound is $O(n^{\frac{5}{2}})$ for any quantile parameter. Given an Oracle which can enumerate the k-sets in 3 dimension, still the QR solution will take $O(n^{\frac{5}{2}})$ time which is theoretically better than QR-IPM. However, we do not know about any such enumeration technique for 3 dimensions which is an open problem for future research. Furthermore, in 4 dimensions,

the upper bound on number of k-set is $O(n^2(k+1)^{2-2/45})$. For any quantile, the upper bound becomes approximately $O(n^4)$ which is worse than QR-IPM. As number of k-set increases significantly in higher dimension, the k-set based solution for QR is not feasible.

4.6 Related work

The idea of quantile regression was introduced even before OLS regression. Around 1757, Boscovich proposed the idea of fitting a line for 2-dimensional data by minimizing the sum of absolute residuals under the assumption of mean of residuals has to be zero. Laplace gave an algebraic formulation of the problem in his Methode de Situation in 1789. In 1809 Gauss suggested to remove zero mean residual constraint. Later in 1823, he also proposed least square criterion (i.e., OLS). As OLS has more analytical and computational simplicity, it has been always popular. However, a criterion is a choice and there are different cases where one criterion will outperform others. In 1888, Edgeworth proposed a computational geometry based solution for the bi-variate median regression problem which helped the development of today's Simplex method. In 20th century, there has been extensive work on quantile regression which helped a wide range of applications. In 1974, Barrodale and Roberts [119] used simplex technique to solve median regression problem as a bounded dual problem. Bloomfield and Steiger [124] explored the simplex technique for median regression in depth and suggested exploring a normalized steepest edge direction instead of steepest edge direction.

Before 1987, all the quantile regression techniques were focusing on median regression. Koenker and d'Orey [136] generalized the criterion for any quantile which is now known as quantile regression. In median regression, an input point incurs a cost of absolute deviation. For quantile regression parameter τ, Koenker and d'Orey proposed to assign $1-\tau$ and τ weight to negative and positive residuals. After the development of gen-

eralized quantile regression for any quantile, there has been an extensive research in this area. We mention a few prominent work in this paper. There have been a various research directions in this area. The primary focus is on the development of a faster quantile regression algorithm [126]. Another research direction is finding quantile regression for multiple quantile[118, 126]. In 1997, Portnoy and Koenker [118] developed a primal dual interior point method where in each step the goal is to minimize the different between primal dual objective loss. It is extremely fast compared to the all previous methods. Later Koenkar also proposed a technique for sparse big data in 2011.

In the continuous quest of a faster quantile regression, the objective criterion has been often modified. As none of these are widely accepted, we only consider the original quantile regression problem in this paper. If we we consider the entire history of quantile regression, the most prominent two techniques are based on Barrodale and Roberts[119], and Portnoy and Koenkar [118]. The latest updated implementation based on these techniques can be found in quantreg package [131] in R maintained by Roger Koenkar which is a gold standard public library for quantile regression. However, these widely used and accepted implementations are quite memory hungry. Moreover, the run time complexity of these implementations is still far behind to face the challenges of big data. So it is extremely important to find out techniques that can take this everincreasing challenge.

bibliographyrefs

REFERENCES

[1] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann, "How good are query optimizers, really?" *PVLDB*, vol. 9, no. 3, 2015.

[2] V. Poosala, P. J. Haas, Y. E. Ioannidis, and E. J. Shekita, "Improved histograms for selectivity estimation of range predicates," in *ACM Sigmod Record*, vol. 25, no. 2, 1996.

[3] M. Germain, K. Gregor, I. Murray, and H. Larochelle, "Made: Masked autoencoder for distribution estimation," in *ICML*, 2015, pp. 881–889.

[4] M. Akdere, U. Çetintemel, M. Riondato, E. Upfal, and S. B. Zdonik, "Learning-based query performance modeling and prediction," in *ICDE*. IEEE, 2012, pp. 390–401.

[5] J. Ortiz, M. Balazinska, J. Gehrke, and S. S. Keerthi, "An empirical analysis of deep learning for cardinality estimation," *arXiv preprint arXiv:1905.06425*, 2019.

[6] B. Hilprecht, A. Schmidt, M. Kulessa, A. Molina, K. Kersting, and C. Binnig, "Deepdb: Learn from data, not from queries!" *PVLDB*, 2020.

[7] Z. Wang, D. Cashman, M. Li, J. Li, M. Berger, J. A. Levine, R. Chang, and C. Scheidegger, "Nncubes: Learned structures for visual data exploration," *arXiv preprint arXiv:1808.08983*, 2018.

[8] J. Sun and G. Li, "An end-to-end learning-based cost estimator," *arXiv preprint arXiv:1906.02560*, 2019.

[9] V. Leis, B. Radke, A. Gubichev, A. Kemper, and T. Neumann, "Cardinality estimation done right: Index-based join sampling." in *CIDR*, 2017.

[10] A. Kipf, T. Kipf, B. Radke, V. Leis, P. Boncz, and A. Kemper, "Learned cardinalities: Estimating correlated joins with deep learning," *CIDR*, 2019.

[11] M. Müller, G. Moerkotte, and O. Kolb, "Improved selectivity estimation by combining knowledge from sampling and synopses," *PVLDB*, vol. 11, no. 9, pp. 1016–1028, 2018.

[12] L. Getoor, B. Taskar, and D. Koller, "Selectivity estimation using probabilistic models," in *ACM SIGMOD Record*, vol. 30, no. 2, 2001.

[13] K. Tzoumas, A. Deshpande, and C. S. Jensen, "Lightweight graphical models for selectivity estimation without independence assumptions," *PVLDB*, vol. 4, no. 11, pp. 852–863, 2011.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[15] K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra, "Deep autoregressive networks," in *ICML*, 2014, pp. 1242–1250. [Online]. Available: http://jmlr.org/proceedings/papers/v32/gregor14.html

[16] B. Uria, I. Murray, and H. Larochelle, "NADE: the real-valued neural autoregressive density-estimator," *CoRR*, vol. abs/1306.0186, 2013. [Online]. Available: http://arxiv.org/abs/1306.0186

[17] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *ICML*, 2008. [Online]. Available: https://doi.org/10.1145/1390156.1390294

[18] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *ICML*, 2011, pp. 833–840.

[19] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[20] G. P. Lepage, "A new algorithm for adaptive multidimensional integration," *Journal of Computational Physics*, vol. 27, no. 2, pp. 192–203, 1978.

[21] V. Poosala and Y. E. Ioannidis, "Selectivity estimation without the attribute value independence assumption," in *VLDB*, vol. 97, 1997, pp. 486–495.

[22] Z. Yang, E. Liang, A. Kamsetty, C. Wu, Y. Duan, X. Chen, P. Abbeel, J. M. Hellerstein, S. Krishnan, and I. Stoica, "Deep unsupervised cardinality estimation," *Proceedings of the VLDB Endowment*, vol. 13, no. 3, pp. 279–292, 2019.

[23] B. Uria, I. Murray, and H. Larochelle, "A deep and tractable density estimator," in *International Conference on Machine Learning*, 2014, pp. 467–475.

[24] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *JMLR*, vol. 15, no. 1, pp. 1929–1958, 2014.

[26] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE TPAMI*, vol. 40, no. 12, pp. 2935–2947, 2018.

[27] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and unsupervised discretization of continuous features," in *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 194–202.

[28] F. Changyong, W. Hongyue, L. Naiji, C. Tian, H. Hua, L. Ying, *et al.*, "Log-transformation and its implications for data analysis," *Shanghai archives of psychiatry*, vol. 26, no. 2, p. 105, 2014.

[29] A. Dutt, C. Wang, A. Nazi, S. Kandula, V. Narasayya, and S. Chaudhuri, "Selectivity estimation for range predicates using lightweight models," *PVLDB*, vol. 12, no. 9, pp. 1044 – 1057, 2019.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, vol. abs/1412.6980, 2015.

[31] N. Thaper, S. Guha, P. Indyk, and N. Koudas, "Dynamic multidimensional histograms," in *SIGMOD*, 2002, pp. 428–439. [Online]. Available: https://doi.org/10.1145/564691.564741

[32] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel, "Optimal histograms with quality guarantees," in *VLDB*, 1998, pp. 275–286. [Online]. Available: http://www.vldb.org/conf/1998/p275.pdf

[33] Y. E. Ioannidis, "The history of histograms (abridged)," in *VLDB*, 2003. [Online]. Available: http://www.vldb.org/conf/2003/papers/S02P01.pdf

[34] N. Bruno and S. Chaudhuri, "Conditional selectivity for statistics on query expressions," in *SIGMOD*, 2004, pp. 311–322. [Online]. Available: https://doi.org/10.1145/1007568.1007604

[35] N. Bruno, S. Chaudhuri, and L. Gravano, "Stholes: A multidimensional workload-aware histogram," in *SIGMOD*, 2001, pp. 211–222. [Online]. Available: https://doi.org/10.1145/375663.375686

[36] Y. Matias, J. S. Vitter, and M. Wang, "Wavelet-based histograms for selectivity estimation," in *ACM SIGMOD Record*, vol. 27, no. 2, 1998.

[37] R. J. Lipton, J. F. Naughton, and D. A. Schneider, *Practical selectivity estimation through adaptive sampling*. ACM, 1990, vol. 19, no. 2.

[38] G. Cormode, M. Garofalakis, P. J. Haas, C. Jermaine, *et al.*, "Synopses for massive data: Samples, histograms, wavelets, sketches," *Foundations and Trends in Databases*, vol. 4, no. 1–3, pp. 1–294, 2011.

[39] J. Neter, W. Wasserman, and M. H. Kutner, "Applied linear regression models," 1989.

[40] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[41] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis, "The case for learned index structures," in *SIGMOD*. ACM, 2018, pp. 489–504.

[42] P. Chilinski and R. Silva, "Neural likelihoods via cumulative distribution functions," *arXiv preprint arXiv:1811.00974*, 2018.

[43] J. Ortiz, M. Balazinska, J. Gehrke, and S. S. Keerthi, "Learning state representations for query optimization with deep reinforcement learning," *arXiv preprint arXiv:1803.08604*, 2018.

[44] R. Marcus and O. Papaemmanouil, "Deep reinforcement learning for join order enumeration," *arXiv preprint arXiv:1803.00055*, 2018.

[45] K. Tzoumas, T. Sellis, and C. S. Jensen, "A reinforcement learning approach for adaptive query processing," *History*, 2008.

[46] S. Krishnan, Z. Yang, K. Goldberg, J. Hellerstein, and I. Stoica, "Learning to optimize join queries with deep reinforcement learning," *arXiv preprint arXiv:1808.03196*, 2018.

[47] T. Kraska, M. Alizadeh, A. Beutel, E. H. Chi, J. Ding, A. Kristo, G. Leclerc, S. Madden, H. Mao, and V. Nathan, "Sagedb: A learned database system," 2019.

[48] R. Marcus, P. Negi, H. Mao, C. Zhang, M. Alizadeh, T. Kraska, O. Papaemmanouil, and N. Tatbul, "Neo: A learned query optimizer," *Proceedings of the VLDB Endowment*, vol. 12, no. 11, pp. 1705–1718, 2019.

[49] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang, "Distributed representations of tuples for entity resolution," *PVLDB*, vol. 11, no. 11, pp. 1454–1467, 2018.

[50] R. Cappuzzo, P. Papotti, and S. Thirumuruganathan, "Creating embeddings of heterogeneous relational datasets for data integration tasks," *SIGMOD*, 2020.

[51] S. Thirumuruganathan, N. Tang, M. Ouzzani, and A. Doan, "Data curation with deep learning," *EDBT*, 2020.

[52] F. Korn, T. Johnson, and H. Jagadish, "Range selectivity estimation for continuous attributes," in *ssdbm*. IEEE, 1999, p. 244.

[53] M. Kiefer, M. Heimel, S. Breß, and V. Markl, "Estimating join selectivities using bandwidth-optimized kernel density models," *PVLDB*, vol. 10, no. 13, pp. 2085–2096, 2017. [Online]. Available: http://www.vldb.org/pvldb/vol10/p2085-kiefer.pdf

[54] D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi, "Selectivity estimators for multidimensional range queries over real attributes," *VLDB J.*, vol. 14, no. 2, pp. 137–154, 2005. [Online]. Available: https://doi.org/10.1007/s00778-003-0090-4

[55] S. Lakshmi and S. Zhou, "Selectivity estimation in extensible databases-a neural network approach," in *VLDB*, 1998, pp. 623–627.

[56] A. Kipf, D. Vorona, J. Müller, T. Kipf, B. Radke, V. Leis, P. Boncz, T. Neumann, and A. Kemper, "Estimating cardinalities with deep sketches," in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 1937–1940.

[57] A. Kipf, M. Freitag, D. Vorona, P. Boncz, T. Neumann, and A. Kemper, "Estimating filtered group-by queries is hard: Deep learning to the rescue," *1st International Workshop on Applied AI for Database Systems and Applications*, 2019.

[58] D. Vorona, A. Kipf, T. Neumann, and A. Kemper, "Deepspace: Approximate geospatial query processing with deep learning," in *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2019, pp. 500–503.

[59] S. Thirumuruganathan, S. Hasan, N. Koudas, and G. Das, "Approximate query processing for data exploration using deep generative models," *ICDE*, 2020.

[60] B. Mozafari and N. Niu, "A handbook for building an approximate query engine." *IEEE Data Eng. Bull.*, vol. 38, no. 3, pp. 3–29, 2015.

[61] B. Mozafari, "Approximate query engines: Commercial challenges and research opportunities," in *SIGMOD*, 2017, pp. 521–524. [Online]. Available: http://doi.acm.org/10.1145/3035918.3056098

[62] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[63] Y. Bengio, I. J. Goodfellow, and A. Courville, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[64] J. Altosaar, *Tutorial - What is a variational autoencoder?*, 2018. [Online]. Available: https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

[65] M. N. Garofalakis and P. B. Gibbons, "Approximate query processing: Taming the terabytes." in *VLDB*, 2001, pp. 343–352.

[66] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*. CRC press, 1994.

[67] Y. Burda, R. Grosse, and R. Salakhutdinov, "Importance weighted autoencoders," *arXiv preprint arXiv:1509.00519*, 2015.

[68] R. M. Neal, "Annealed importance sampling," *Statistics and computing*, vol. 11, no. 2, pp. 125–139, 2001.

[69] A. Grover, R. Gummadi, M. Lazaro-Gredilla, D. Schuurmans, and S. Ermon, "Variational rejection sampling," in *AISTATS*, 2018.

[70] W. G. Cochran, *Sampling techniques*. John Wiley & Sons, 2007.

[71] P. R. Rosenbaum, "An exact distribution-free test comparing two multivariate distributions based on adjacency," *JRSS: Series B*, vol. 67, no. 4, pp. 515–530, 2005.

[72] L. Wasserman, *Modern Two-Sample Tests*, 2012. [Online]. Available: https://normaldeviate.wordpress.com/2012/07/14/modern-two-sample-tests/

[73] J. Edmonds, "Paths, trees, and flowers," *Canadian Journal of mathematics*, vol. 17, pp. 449–467, 1965.

[74] R. G. Krishnan and M. Hoffman, "Inference and introspection in deep generative models of sparse data," *Advances in Approximate Bayesian Inference Workshop at NIPS*, 2016.

[75] M. Kulessa, A. Molina, C. Binnig, B. Hilprecht, and K. Kersting, "Model-based approximate query processing," *arXiv:1811.06224*, 2018.

[76] A. Galakatos, A. Crotty, E. Zgraggen, C. Binnig, and T. Kraska, "Revisiting reuse for approximate query processing," *PVLDB*, 2017. [Online]. Available: https://doi.org/10.14778/3115404.3115418

[77] F. Reiss, M. Garofalakis, and J. M. Hellerstein, "Compact histograms for hierarchical identifiers," in *Proceedings of the 32nd international conference on Very large data bases*. VLDB Endowment, 2006, pp. 870–881.

[78] S. Guha, N. Koudas, and D. Srivastava, "Fast algorithms for hierarchical range histogram construction," in *PODS*, 2002. [Online]. Available: http://doi.acm.org/10.1145/543613.543637

[79] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[80] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[81] M. Scutari and J.-B. Denis, *Bayesian Networks with Examples in R*. Boca Raton: Chapman and Hall, 2014.

[82] A. Molina, A. Vergari, N. Di Mauro, S. Natarajan, F. Esposito, and K. Kersting, "Mixed sum-product networks: A deep architecture for hybrid domains," in *AAAI*, 2018.

[83] "UCI Machine Learning Repository. Adult Data Set ," 2019. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/adult

[84] "Bureau of transportation statistics. Flights Data Set ," 2019. [Online]. Available: http://www.transtats.bts.gov/

[85] P. Eichmann, C. Binnig, T. Kraska, and E. Zgraggen, "Idebench: A benchmark for interactive data exploration," *arXiv:1804.02593*, 2018.

[86] I. J. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *CoRR*, vol. abs/1701.00160, 2017. [Online]. Available: http://arxiv.org/abs/1701.00160

[87] Q. Ma and P. Triantafillou, "Dbest: Revisiting approximate query processing engines with machine learning models," in *SIGMOD*, 2019.

[88] Z. Wang, D. Cashman, M. Li, J. Li, M. Berger, J. A. Levine, R. Chang, and C. Scheidegger, "Neuralcubes: Deep representations for visual data exploration," *CoRR*, vol. abs/1808.08983, 2018.

[89] A. Pavlo, G. Angulo, J. Arulraj, H. Lin, J. Lin, L. Ma, P. Menon, T. C. Mowry, M. Perron, I. Quah, *et al.*, "Self-driving database management systems." in *CIDR*, 2017.

[90] R. Marcus and O. Papaemmanouil, "Releasing cloud databases for the chains of performance prediction models." in *CIDR*, 2017.

[91] S. Venkataraman, Z. Yang, M. J. Franklin, B. Recht, and I. Stoica, "Ernest: Efficient performance prediction for large-scale advanced analytics." in *NSDI*, 2016, pp. 363–378.

[92] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy, "The aqua approximate query answering system," in *ACM Sigmod Record*, vol. 28, no. 2. ACM, 1999, pp. 574–576.

[93] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica, "Blinkdb: Queries with bounded errors and bounded response times on very large data," in *EuroSys*, 2013. [Online]. Available: http://doi.acm.org/10.1145/2465351.2465355

[94] Y. Park, B. Mozafari, J. Sorenson, and J. Wang, "Verdictdb: universalizing approximate query processing," in *SIGMOD*, 2018.

[95] S. Kandula, A. Shanbhag, A. Vitorovic, M. Olma, R. Grandl, S. Chaudhuri, and B. Ding, "Quickr: Lazily approximating complex adhoc queries in bigdata clusters," in *SIGMOD*. ACM, 2016, pp. 631–646.

[96] S. Chaudhuri, G. Das, and V. Narasayya, "Optimized stratified sampling for approximate query processing," *TODS*, vol. 32, no. 2, p. 9, 2007.

[97] J. M. Hellerstein, P. J. Haas, and H. J. Wang, "Online aggregation," in *Acm Sigmod Record*, vol. 26, no. 2. ACM, 1997, pp. 171–182.

[98] S. Wu, B. C. Ooi, and K.-L. Tan, "Continuous sampling for online aggregation over multiple queries," in *SIGMOD*, 2010.

[99] S. Chaudhuri, B. Ding, and S. Kandula, "Approximate query processing: No silver bullet," in *SIGMOD*, 2017. [Online]. Available: http://doi.acm.org/10.1145/3035918.3056097

[100] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim, "Data synthesis based on generative adversarial networks," *PVLDB*, vol. 11, no. 10, pp. 1071–1083, 2018. [Online]. Available: http://www.vldb.org/pvldb/vol11/p1071-park.pdf

[101] https://cacm.acm.org/news/241072-computing-power-needed-to-train-ai-is-rising-seven-times-faster-than before/fulltext.

[102] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," *Advances in neural information processing systems*, vol. 28, 2015.

[103] I. Goodfellow, P. McDaniel, and N. Papernot, "Making machine learning robust against adversarial inputs," *Communications of the ACM*, vol. 61, no. 7, pp. 56–66, 2018.

[104] D. Su, H. Zhang, H. Chen, J. Yi, P.-Y. Chen, and Y. Gao, "Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 631–648.

[105] J. Z. Li, "Principled approaches to robust machine learning and beyond," Ph.D. dissertation, Massachusetts Institute of Technology, 2018.

[106] M. Fink, Y. Liu, A. Engstle, and S.-A. Schneider, "Deep learning-based multi-scale multi-object detection and classification for autonomous driving," pp. 233–242, 2019.

[107] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean, "A guide to deep learning in healthcare," *Nature medicine*, vol. 25, no. 1, pp. 24–29, 2019.

[108] A. F. v. Veenstra and B. Kotterink, "Data-driven policy making: The policy lab approach," in *International conference on electronic participation*. Springer, 2017, pp. 100–111.

[109] A. Shapiro, "Reform predictive policing," *Nature*, vol. 541, no. 7638, pp. 458–460, 2017.

[110] S. Hasan, S. Thirumuruganathan, J. Augustine, N. Koudas, and G. Das, "Deep learning models for selectivity estimation of multi-attribute queries," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1035–1050.

[111] S. Thirumuruganathan, S. Hasan, N. Koudas, and G. Das, "Approximate query processing for data exploration using deep generative models," in *2020 IEEE 36th international conference on data engineering (ICDE)*. IEEE, 2020, pp. 1309–1320.

[112] M. A. Olsen, F. Tian, A. E. Wallace, K. B. Nickel, D. K. Warren, V. J. Fraser, N. Selvam, and B. H. Hamilton, "Use of quantile regression to determine the impact on

total health care costs of surgical site infections following common ambulatory procedures," *Annals of surgery*, vol. 265, no. 2, p. 331, 2017.

[113] B. S. Cade and B. R. Noon, "A gentle introduction to quantile regression for ecologists," *Frontiers in Ecology and the Environment*, vol. 1, no. 8, pp. 412–420, 2003.

[114] C. Davino, M. Furno, and D. Vistocco, *Quantile regression: theory and applications*. John Wiley & Sons, 2013, vol. 988.

[115] S. Thirumuruganathan, S. Shetiya, N. Koudas, and G. Das, "Prediction intervals for learned cardinality estimation: An experimental evaluation," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 3051–3064.

[116] F. Savva, C. Anagnostopoulos, and P. Triantafillou, "Ml-aqp: Query-driven approximate query processing based on machine learning," *arXiv preprint arXiv:2003.06613*, 2020.

[117] M. R. Chernick, "The elements of statistical learning: Data mining, inference and prediction," 2002.

[118] S. Portnoy and R. Koenker, "The gaussian hare and the laplacian tortoise: computability of squared-error versus absolute-error estimators," *Statistical Science*, vol. 12, no. 4, pp. 279–300, 1997.

[119] I. Barrodale and F. D. Roberts, "An improved algorithm for discrete l_1 linear approximation," *SIAM Journal on Numerical Analysis*, vol. 10, no. 5, pp. 839–848, 1973.

[120] F. Y. Edgeworth, "Xxii. on a new method of reducing observations relating to several quantities," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 25, no. 154, pp. 184–191, 1888.

[121] S. J. Wright, *Primal-dual interior-point methods*. SIAM, 1997.

[122] H. Edelsbrunner, *Algorithms in combinatorial geometry*. Springer Science & Business Media, 1987, vol. 10.

[123] R. Koenker and G. Bassett Jr, "Regression quantiles," *Econometrica: journal of the Econometric Society*, pp. 33–50, 1978.

[124] P. Bloomfield and W. L. Steiger, *Least absolute deviations: theory, applications, and algorithms*. Springer, 1983.

[125] H. M. Wagner, "Linear programming techniques for regression analysis," *Journal of the American Statistical Association*, vol. 54, no. 285, pp. 206–212, 1959.

[126] V. Chernozhukov, I. Fernández-Val, and B. Melly, "Fast algorithms for the quantile regression process," *Empirical economics*, pp. 1–27, 2020.

[127] J. Yang, X. Meng, and M. Mahoney, "Quantile regression for large-scale applications," in *International Conference on Machine Learning*. PMLR, 2013, pp. 881–887.

[128] N. Meinshausen and G. Ridgeway, "Quantile regression forests." *Journal of machine learning research*, vol. 7, no. 6, 2006.

[129] Y. Feng, Y. Chen, and X. He, "Bayesian quantile regression with approximate likelihood," *Bernoulli*, vol. 21, no. 2, pp. 832–850, 2015.

[130] S. Zheng, "Gradient descent algorithms for quantile regression with smooth approximation," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 3, pp. 191–207, 2011.

[131] https://cran.r project.org/web/packages/quantreg/index.html.

[132] H. Edelsbrunner and E. Welzl, "Constructing belts in two-dimensional arrangements with applications," *SIAM Journal on Computing*, vol. 15, no. 1, pp. 271–284, 1986.

[133] T. M. Chan, "Dynamic planar convex hull operations in near-logarithmic amortized time," *Journal of the ACM (JACM)*, vol. 48, no. 1, pp. 1–12, 2001.

[134] ——, "Remarks on k-level algorithms in the plane," 1999.

[135] T. K. Dey, "Improved bounds on planar k-sets and k-levels," in *Proceedings 38th Annual Symposium on Foundations of Computer Science*. IEEE, 1997, pp. 156–161.

[136] R. W. Koenker and V. d'Orey, "Algorithm as 229: Computing regression quantiles," *Applied statistics*, pp. 383–393, 1987.

BIOGRAPHICAL STATEMENT

Shohedul Hasan was born in Bangladesh in 1989. He received his B.S. degree in Computer Science and Engineering from the Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh, in 2014. After graduating from BUET, he started a remote Software Engineering job in a U.S. startup named Q.I. Analysis Inc. After two years working at Q.I., he came to the USA for his Ph.D. in the Fall 2016 at The University of Texas at Arlington. His current research interests are Databases, Algorithms, Data Mining, Machine Learning, and Data Science.