Explaining Aggregates for Exploratory Analytics

Fotis Savva
University of Glasgow, UK
f.savva.1@research.gla.ac.uk

Christos Anagnostopoulos
University of Glasgow, UK
christos.anagnostopoulos@glasgow.ac.uk

Peter Triantafillou
University of Warwick, UK
p.triantafillou@warwick.ac.uk

***Abstract*—Analysts wishing to explore (multivariate) data spaces, typically pose queries involving selection operators (i.e., range or radius queries, which in essence define data subspaces of possible interest) and then use aggregation functions, the results of which determine their interesting-ness. However, such aggregate query (AQ) results are simple scalar values. As such, they convey limited information about the queried spaces for exploratory analysis. We aim to address this shortcoming, aiding analysts to explore and better understand data (sub)spaces by contributing a novel type of explanations (coined XAXA: eXplaining Aggregates for eXploratory Analytics). XAXA's novel AQ explanations are represented using functions, which are obtained by solving a three-fold joint optimization problem. Specifically, explanations assume the form of a set of parametric piecewise-linear functions acquired through statistical/Machine Learning (ML) models and algorithms. A key feature of the proposed solution is that model training is performed by only monitoring AQs and their answers on-line. Using our models, explanations for future AQs can be computed without any DB access and can be used to further explore/understand the queried data (sub)spaces, without issuing any more queries to the DB. This greatly simplifies and expedites exploratory data analysis. We evaluate the explanation accuracy and efficiency of XAXA by applying theoretically grounded metrics over real-world and synthetic datasets and query workloads.**

***Index Terms*—Exploratory analytics, aggregates, explanations, PLR regression, vector quantization, hetero-associative statistical learning, query-driven analytics.**

I. Introduction

In the era of big data, analysts must be able to explore and understand huge data spaces and derive new knowledge in an efficient manner. The typical procedure followed by analysts is rather ad hoc and domain specific, but invariantly includes the fundamental step of *exploratory analysis* [20].

Aggregate Queries (AQs), e.g., `COUNT`, `SUM`, `AVG` (`AVERAGE`), play a key role in exploratory analysis as they summarize regions of data. Using these aggregates, analysts decide whether a region is of importance, depending on the task at hand. However, AQs return single values possibly conveying little information. For example, imagine checking whether a particular range of zip codes is of interest, where the latter depends on the count of persons enjoying a 'high' income: if the count of a particular subspace is 273, what does that mean? If the selection (range) predicate was less or more selective, how would this count change? What is the region size with the minimum/maximum count of persons? Similarly, how do the various subregions of the queried subspace contribute to the total count value? To answer such exploratory questions and enhance the analysts understanding of the queried subspace, they need to issue more queries. The analyst has no understanding of the space that would steer them in the right direction w.r.t. which queries to pose next; as such, further exploration becomes ad hoc, unsystematic, and uninformed. To this end, the main objective of this paper is to find ways to assist the analysts understanding such subspaces. A convenient way to represent how something is derived (in terms of compactly and succinctly conveying rich information) is adopting a regression function. In turn, this would lead to fewer queries issued against the system saving system resources and drastically reducing the time needed for exploratory analysis.

A. Motivations and Uses

We focus on AQs with a *center-radius selection* operator (CRS) because of their wide-applicability in data analytics tasks and their easy extension to 1-d range queries. A CRS operator is defined by a multi-dimensional point (center) and a radius. Such operator is evident in many applications including: location-based search, e.g searching for spatially-close (within a radius) objects, such as astronomical objects, objects within a geographical region etc. In addition, the same operator is found within social network analytics, e.g when looking for points within a similarity distance in a social graph. Finally, please note that such operator can easily be extended to formulate an AQ over a specific attribute, e.g a user can issue an AQ with `VAR` and a CRS to retrieve the variance for a specific attribute, over the whole range or a portion of it.

Crimes Data: Consider analyzing crimes' datasets, containing recorded crimes, along with their location, the type of crime (homicide, burglary, etc.) and other information. One such dataset is the Chicago Crimes Dataset [1]. A typical exploration technique is to issue the AQ with a CRS operator:

```
SELECT COUNT(*) AS y FROM Crimes AS C
WHERE $theta>sqrt(power($X-C.X,2)
+power($Y-C.Y,2));
```

The multi-dimensional center is defined by (*$X*, *$Y*) and radius by *$theta*. Such AQ returns the number of crimes in a specific area of interest located around the center. With the given parameters defining a data subspace corresponding to an arbitrary neighborhood. Having a function as an explanation for this AQ, the analyst could use it to further understand how the queried subspace and its subregions contribute to the aggregate value. For instance, Figure 1 (left) depicts the specified AQ as the out-most colored circle. Where the x,y

axes are the *(Lat, Lon)* and the blue points are the locations of incidents reported from the Chicago Crimes Dataset. The different colors inside the circle denote the different rates at which the number of crimes increase as the region gets larger. Thus, consulting this visualisation the analyst can infer the contributions of the different subspaces defined as the separate concentric circles. Starting from the center of the circle we see that the rate is small, then varies as parameter *$theta* gets larger. The different concentric circles and the different rates (different colors at different radii) are obtained using a Piecewise-Linear Regression function. Finally, the availability of an explanation function will facilitate the further exploration of the aggregate value for subregions of smaller or greater sizes without issuing additional SQL queries, but instead simply plugging different parameter values in the explanation function.

Telecommunication Calls Data: Consider a scientist tasked with identifying time-frames having high average call times. She needs to issue AQs for varying-sized ranges over time, such as this SQL range-AQ.

```
SELECT AVG(call_time) AS y FROM TCD AS C
WHERE C.time BETWEEN $X-$theta AND $X+$theta
```

(We can immediately see the usefulness to more than just spatial queries using a 1-d range selection operator on any attribute.) Discovering the aforementioned time-frames, without our proposed explanations, can be a daunting task as multiple queries have to be issued, overflowing the system with a number of AQs. Beyond that, analysts could formulate an optimization problem that could be solved using our function. Given a function, the maxima and minima points can be inferred. Thus the analyst can easily discover the parameter at which the AQ result is maximized or minimized. Again, such functionalities are very much lacking and are crucial for exploratory analytics.

B. Problem Definition

We consider the result of an AQ stemming from a function, defined from this point onwards as the *true* function. With that true function fluctuating as the parameter values of an AQ (e.g *$X, $Y, $theta*) change. Thus, we are faced with the problem of being able to approximate the true function with high accuracy. Using the aforementioned facts we construct an optimization problem to which its solution is an optimal approximation to the *true* function. In turn, this problem is further deconstructed into two optimization problems. The first being the need to find optimal query parameters representing a query space. The described AQs often form clusters, as a number of AQs are issued with similar parameters. This is evident from Figure 1 (right) displaying AQs (from real-world analytical query workload [31]) forming clusters in the query space. AQs clustered together will have similar results, thus similar true functions. Consequently, we are interested in finding a set of optimal parameters to represent queries in each one of those clusters, as this will be more accurate than one set

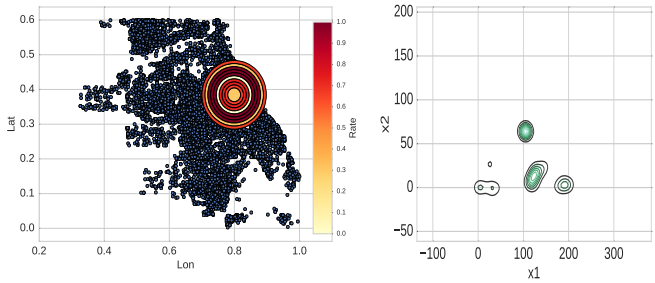


Fig. 1. (Left) Use case of explanation function; x-axis is Longitude, y-axis is Latitude; blue points are the locations of reported incidents; Circle on right is the AQ with a CRS operator with the varying rate-of-increase shown as color coded concentric circles; (right) Real workload cluster analysis (Source SDSS [31]); x_1 and x_2 axes are parameters of a CRS operator.

of global parameters[1]. The second part of the deconstructed optimization problem is being able to approximate the true function(s) using a known class of functions over the clustered query space. To this end we use piecewise-linear functions that are fitted over the clustered query space. In addition, as we consider both historically observed queries and incoming queries we form a third joint optimization problem which is solved in an on-line manner by adapting both the obtained optimal parameters and the fitted function.

II. Related Work & Contribution

XAXA aims to provide explanations for AQ queries. AQs and their efficient computation has been a major research interest [3], [5], [6], [11], [13], [14], [14], [18], [19], [27], [34], [36], with methods applying sampling techniques, synopses and ML models to compute such AQs. Compared to XAXA, the above works are largely complementary. However, the key salient and distinguishing feature of XAXA is that its primary task is to *explain the AQ results* and do so efficiently, accurately, and scalably w.r.t. increasing data sizes.

Explanation techniques have emerged in multiple contexts within the data management and ML communities. Their origin stems from data provenance [12], but have since departed from such notions, and focus on explanations in varying contexts and with different objectives. One of such contexts, is to find explanations, represented as predicates, for simple query answers as in [15], [25], [29].

Explanations have been used in domains such as interpreting outliers in both *in-situ* data [35] and in streaming data [7]. In [26], the authors build a system to provide interpretations for service errors and present the explanations visually to assist debugging large scale systems. In addition, the authors of PerfXPlain [22] created a tool to assist users while debugging performance issues in Map-Reduce jobs. Other frameworks provide explanations that in turn can be utilized by the users to locate any discrepancies found in their data [32], [9], [33]. A recent trend is in explaining ML models for debugging

[1]This basically represents all queries in a cluster by one query.

purposes [23] or to understand how a model makes predictions [30] and conveys trust to users using these predictions [28].

In this work, we particularly focus on explanations for AQs because of the AQs' wide use in exploratory analytics [34]. Both [35] and [4] focused on explaining aggregate queries. Our major difference is that we wish to explain these AQs solely based on the input parameters and results of previous and incoming queries, thus not having to rely on the underlying data, which makes generating explanations slow and inefficient for large-scale datasets. That being said, our objectives and criteria are vastly different than the aforementioned related work.

Scalability/efficiency is particularly important: Computing explanations is proved to be an NP-Hard problem [32] and generating them can take a long time [15], [29], [35] even with modest datasets. An exponential increase in data size will imply a dramatic increase in the time to generate explanations. XAXA does not suffer from such limitations and is able to construct explanations in a matter of milliseconds even with massive data volumes. To do so, it relies on two pillars: First, on workload characteristics: workloads contain a large number of overlapping queried data subspaces. This characteristic has been acknowledged and exploited also by recent research e.g., [34], [27], STRAT [10], and SciBORQ [24] and has been found to hold in real-world workloads involving exploratory/statistical analysis, such as in the Sloan Digital Sky Survey and in the workload of SQLShare [21]. Second, XAXA relies on a novel ML model that exploit the above workload characteristics to perform efficient and accurate (approximate) AQ explanations by only monitoring queries in an on-line manner.

Our **contributions** are:

1) Novel explanations for AQs useful for exploratory analytics.
2) Definition of finding the optimal AQ explanation functions as a three-fold joint optimization problem.
3) A novel hetero-associative statistical learning model, for formalizing and solving the joint optimization problem.
4) Model training, AQ processing and explanation generation algorithms requiring *zero* data access.
5) Use of principled metrics for evaluating the accuracy of AQ explanations, their efficiency and scalability, and their sensitivity to key parameters.

III. Representing Explanations

We formally model data items (tuples) and AQs with a CRS operator, as vectors in a vectorial data space, $\mathbb{D} \subset \mathbb{R}^d$, and vectorial query space, $\mathbb{Q} \subset \mathbb{R}^{d+1}$.

A. Vectorial Representation of Queries

Let $\mathbf{x} = [x_1, \ldots, x_d] \in \mathbb{R}^d$ denote a random row vector in data space $\mathbb{D}$.

Definition 1: (Vector Norm) The p-norm (L_p) distance between two vectors $\mathbf{x}$ and $\mathbf{x}'$ from $\mathbb{R}^d$ for $1 \leq p < \infty$, is $\|\mathbf{x} - \mathbf{x}'\|_p = (\sum_{i=1}^{d} |x_i - x'_i|^p)^{\frac{1}{p}}$ and for $p = \infty$, is $\|\mathbf{x} - \mathbf{x}'\|_\infty = \max_{i=1,\ldots,d}\{|x_i - x'_i|\}$.

Consider now a scalar $\theta > 0$, hereinafter referred to as *radius*, and a dataset $\mathcal{B}$ consisting of N vectors $\{\mathbf{x}_i\}_{i=1}^{N}$.

Definition 2: (Data Subspace) Given $\mathbf{x} \in \mathbb{R}^d$ and scalar θ, being the parameters of an AQ with a CRS operator, a data subspace $\mathbb{D}(\mathbf{x}, \theta)$ is the convex subspace of $\mathbb{R}^d$, which includes vectors $\mathbf{x}_i : \|\mathbf{x}_i - \mathbf{x}\|_p \leq \theta$ with $\mathbf{x}_i \in \mathcal{B}$. Thus the region enclosed by a CRS operator is the referred data subspace.

Definition 3: (Aggregate Query) Given a data subspace $\mathbb{D}(\mathbf{x}, \theta)$ an AQ is a function over $\mathbb{D}(\mathbf{x}, \theta)$, that produces a response variable $y = f(\mathbb{D}(\mathbf{x}, \theta))$ which is the result of an AQ. So for instance, in the case of `AVG` the function could be $f(\mathbb{D}) = \mathbb{E}[y|\mathbb{D}(\mathbf{x}, \theta)]$, where y is the attribute we are interested in e.g *call_time*.

Definition 4: (Query Similarity) The L_2^2 distance or similarity measure between AQs $\mathbf{q}, \mathbf{q}' \in \mathbb{Q}$ is $\|\mathbf{q} - \mathbf{q}'\|_2^2 = \|\mathbf{x} - \mathbf{x}'\|_2^2 + (\theta - \theta')^2$.

B. Functional Representation of Explanations

The AQs defined in Section III-A return a single scalar value y, to the analysts. We seek to explain how such values are generated by finding a *function* $f : \mathbb{R} \times \mathbb{R}^d \rightarrow \mathbb{R}$ that can describe how y, is produced given an ad-hoc query $\mathbf{q} = [\mathbf{x}, \theta]$ with $\mathbf{x} \in \mathbb{R}^d$. Specifically, given our interest in parameter $\mathbf{x}$, we desire explaining the evolution/variability of output y of query $\mathbf{q}$ as the radius θ varies. Therefore, given a parameter of interest $\mathbf{x}$ we define a parametric function $f(\theta; \mathbf{x})$ to which its input is the radius θ. Thus, our produced explanation function(s) approximates the *true* function w.r.t radius θ conditioned on $\mathbf{x}$.

An explanation function can be linear or polynomial w.r.t θ given a fixed center $\mathbf{x}$. However, when using high-order polynomial functions as our explanations, we assume that none of the true AQ functions will be monotonically increasing, and that we know the order of the polynomial. A monotonically increasing AQ function w.r.t θ is `COUNT`. However, for aggregates such as `AVG` the previous assumption does not hold: Hence, representing explanations with high-order polynomial functions will be problematic. Therefore, we choose to employ linear functions. But, linear functions will still have trouble representing explanations accurately. For instance, if the result y is given by `AVG` or `CORR`, then the output of the function might increase or decrease for a changing θ. Even for `COUNT`, a single linear function might be an incorrect representation as the output y might remain constant within an interval θ and then increase abruptly.

Figure 2 (left) shows that the *true* function for this AQ (eg `COUNT`), is monotonically increasing w.r.t θ given $\mathbf{x}$. The resulting *linear* and *polynomial* fitting/approximation functions fail to be representative. The same holds for Figure 2 (right) in which the *true* function is non-linear for this AQ (eg `AVG`). Hence, we need to find an adjustable linear function that is able to capture such irregularities and conditional non-linearities. A more appropriate solution is to approximate the explanation function f with a Piecewise Linear Regression(PLR) function a.k.a. *segmented regression*. A PLR approximation of f is able to address the above shortcomings by finding and fitting the

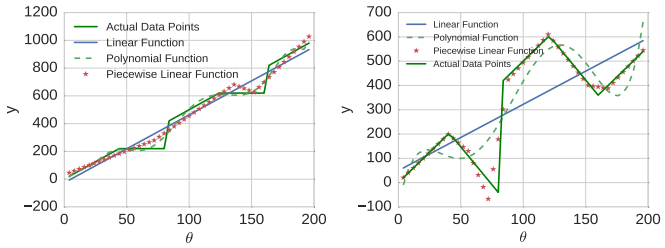


Fig. 2. (Left) Actual and approximate explanation functions for a monotonically increasing result, where x-axis is the radius θ of a query and y-axis is the result y given $\mathbf{x}$; (right) Actual and approximate explanation functions for non-linear function vs radius θ of a query given $\mathbf{x}$.

best multiple linear functions, as illustrated in Figure 2. We now provide the definition of an explanation:

Definition 5: (Explanation Function) Given an AQ $\mathbf{q} = [\mathbf{x}, \theta]$, an explanation function $f(\theta; \mathbf{x})$ is defined as the fusion of piecewise linear functions, $f \approx \sum \hat{f}(\theta; \mathbf{x})$, derived by the fitting of $\hat{f}$ over AQs $\mathbf{q}' = [\mathbf{x}', \theta']$ with radii θ'.

IV. Explanation Fundamentals

The challenge in approximating f, as defined in Definition 5 lies in seeking functions to explain the way result y varies as radius θ changes without access to base data, as this would harm efficiency. Specifically, we seek to find explanation functions by only leveraging previous and incoming AQs.

A. Explanation Approximation

XAXA utilizes AQs to train statistical learning models that are able to accurately approximate the *true* explanation function f for any center of interest $\mathbf{x}$. Given these models, we no longer need access to the underlying data, thus, guaranteeing efficiency. This follows since the models learn the various query patterns to approximate explanation functions $f(\theta; \mathbf{x}), \forall \mathbf{x} \in \mathbb{R}^d$.

Formally, given a well defined explanation loss $\mathcal{L}(f, \hat{f})$ (defined later) between the *true explanation function* $f(\theta; \mathbf{x})$ and an *approximated* function $\hat{f}(\theta; \mathbf{x}) \in \mathcal{F}$, we seek the optimal approximation function $\hat{f}^*$ that minimizes the Expected Explanation Loss (EEL) for *all possible* queries (all locations and radii):

$$\hat{f}^* = \arg\min_{\hat{f} \in \mathcal{F}} \int_{\mathbf{x} \in \mathbb{R}^d} \int_{\theta \in \mathbb{R}_+} \mathcal{L}(f(\theta; \mathbf{x}), \hat{f}(\theta; \mathbf{x})) p(\theta, \mathbf{x}) d\theta d\mathbf{x}, \quad (1)$$

where $p(\theta, \mathbf{x})$ is the probability density function of the queries $\mathbf{q} = [\mathbf{x}, \theta]$ over the query workload. Eq(1) is the objective minimization loss function given that we use the optimal model approximation function to explain the relation between y and radius θ for any given/fixed center $\mathbf{x}$.

However, accuracy will be problematic as it seems intuitively wrong that one such function can explain *all* the queries at any location $\mathbf{x} \in \mathbb{R}^d$ with any radius $\theta \in \mathbb{R}$. Such an explanation is not accurate because: (i) The radius θ can generally be different for different queries. For instance, crime data analysts will issue AQs with a different radius to compare if crimes centered within a small radius at a given location increase at a larger/smaller radius; (ii) At different locations $\mathbf{x}$, the result y will almost surely be different.

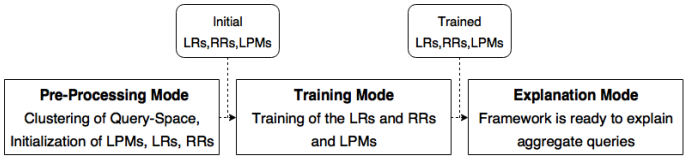


Fig. 3. The XAXA framework modes overview.

Therefore we introduce *local* approximation functions $\hat{f}_1, \ldots, \hat{f}_K$ that collectively minimize the objective in (1). Thus, we no longer wish to find one global approximation to the true explanation function for all possible queries. Instead, we fit a number of local optimal functions that can explain a *subset* of possible queries. We refer to those models as Local PLR Models (LPM).

The LPMs are fitted using AQs forming a cluster (have similar query parameters). Thus for each uncovered cluster of AQs we fit an LPM. Therefore, if K clusters are obtained from clustering the Query space $\mathbb{Q}$, K LPMs are fitted.

The obtained K LPMs are essentially multiple possible explanations. Intuitively, this approach makes more sense as queries that are issued at relatively close proximity to each other, location-wise, and that have similar radii will tend to have similar results and in turn will behave more or less the same way. The resulting fused explanation has more accuracy as the clusters have less variance for y and θ. This was empirically shown from our experimental workload.

Formally, to minimize EEL, we seek K local approximation model functions $\hat{f}_k \in \mathcal{F}, k = 1 \in [K]$, such that for each query $\mathbf{q}$ belonging to a partition/cluster k, $\mathbb{Q}_k$, the summation of the local EELs is minimized:

$$\mathcal{J}_0(\{\hat{f}_k\}) = \sum_{\hat{f}_k \in \mathcal{F}} \int_{\mathbf{q} \in \mathbb{Q}_k \subset \mathbb{R}^{d+1}} \mathcal{L}(f(\theta; \mathbf{x}), \hat{f}_k(\theta; \mathbf{x})) p_k(\mathbf{q}) d\mathbf{q} \quad (2)$$

where $p_k(\mathbf{q})$ is the probability density function of the query vectors belonging to a query sub-space $\mathbb{Q}_k$. Thus, $\mathcal{J}_0$ forms our *generic* optimization problem mentioned in Section I-B. We define *explanation loss*, $\mathcal{L}(f, \hat{f})$ as the discrepancy of the actual explanation function f due to the approximation of explanation $\hat{f}$. For evaluating $\mathcal{L}$, we propose two different aspects to measure *explanation loss*: *(1) the statistical aspect*, where the goodness of fit of our explanation function is measured and (2) the *predictive accuracy* denoting how well the results from the *true* function can be approximated using our explanation function. The proposed metrics are covered in Section VIII.

B. A Bird's Eye View of the Solution

The proposed methodology for computing explanations is split into three *modes* (Figure 3). The *Pre-Processing Mode* aims to identify the optimal number of LPMs and an initial approximation of their parameters using previously executed

queries. The purpose of this mode is basically to jump-start our framework. In the *Training Mode*, the LPMs' parameters are incrementally optimized to minimize the objective function (2) as incoming queries are processed in an on-line manner. In the *Explanation Mode*, the framework is ready to explain AQ results.

1) Pre-Processing Mode: A training set $\mathcal{T}$ of m previously executed queries $\mathbf{q}$ and their corresponding aggregate results is used as input to the *Pre-Processing Mode*. The central task is to partition (quantize) the query space $\mathbb{Q}$ based on the observed (previous) queries $\mathbf{q} \in \mathcal{T}$ into K clusters, sub-spaces $\mathbb{Q}_k$, in which queries with similar $\mathbf{x}$ are grouped together. Each cluster is then further quantized into L sub-clusters, as queries with similar $\mathbf{x}$ are separated with regards to their θ parameter values. Therefore, this is a hierarchical query space quantization (first level partition w.r.t. $\mathbf{x}$ and second level partition w.r.t. θ), where each Level-1 (L1) cluster $\mathbb{Q}_k, k = 1, \ldots, K$ is associated with a number of Level-2 (L2) sub-clusters $\mathbb{U}_{kl}, l = 1, \ldots, L$ in the θ space. For each L1 cluster $\mathbb{Q}_k$ and L2 sub-cluster $\mathbb{U}_{kl}$, we assign a L1 representative, hereinafter referred to as Location Representative (LR) and a L2 representative, hereinafter referred to as Radius Representative (RR). LR converges to the mean vector of the centers of all queries belonging to L1 cluster $\mathbb{Q}_k$, while the associated RR converges to the mean radius value of the radii of all queries, whose radius values belong to $\mathbb{U}_{kl}$.

After this hierarchical quantization of the query space, the task is to *associate an LPM* $\hat{f}_{kl}(\theta; \mathbf{x})$ with *each* L2 sub-cluster $\mathbb{U}_{kl}$, given that the center parameter $\mathbf{x}$ is a member of the corresponding L1 cluster $\mathbb{Q}_k$. This process is nicely summarized in Figure 4.

2) Training Mode: This mode optimizes the clustering parameters (LR and RR representatives) of the pre-processing mode in order to minimize the objective function (2). This optimization process is achieved incrementally by processing each new pair $(\mathbf{q}_i, y_i)$ in an on-line manner. Consulting Figure 4, in *Training* mode, each incoming query $\mathbf{q}_i$ is projected/mapped to the closest LR corresponding to a L1 cluster. Since, the closest LR is associated with a number of L2 RRs, the query is then assigned to one of those RRs (closest u to θ), and then the associated representatives are adapted. After a pre-specified number of processed queries, the corresponding LPM $\hat{f}_{kl}(\theta; \mathbf{x})$ is re-trained to associate the result y with the θ values in $\mathbb{U}_{kl}$ and account for the newly associated queries.

3) Explanation Mode: In this mode no more modifications to LRs, RRs, and the associated LPMs are made. Based on the L1/2 representatives and their associated approximation explanation functions LPMs, the model is now ready to provide explanations for AQs. Figure 4 sums up the result of all three modes and how an explanation is given to the user. For a given query $\mathbf{q}$, the system finds the closest LR;$\mathbf{w}_k$ and then, based on a combination of the RRs;$(u_{k,1}, \ldots, u_{k,3})$ and their associated LPMs;$(\hat{f}_{k,1} \ldots, \hat{f}_{k,3})$, returns an explanation as a fusion of diverse PLR functions derived by the L2 level. We elaborate on this fusion of L2 LPMs in Section VII[2].

V. Optimization Problems Formulations

We initially defined the generic optimization problem in (2), which identifies the need for local approximations of the true explanation function. Subsequently, we defined a high level solution that allows us to tackle such problem. In this section, we deconstruct the generic problem into more specific optimization problems, where XAXA minimizes the EEL.

A. Optimization Problem 1: Query Space Clustering

The first part of the deconstructed generic problem identifies the need to find *optimal* LRs and RRs, as such optimal parameters guarantee better grouping of queries thus better approximation of *true* function, during the *Pre-Processing* phase. The LRs are initially random location vectors $\mathbf{w}_k \in \mathbb{R}^d, k = 1, \ldots, K$, and are then refined by a clustering algorithm as the mean vectors of the query locations that are assigned to each LR. Formally, this phase finds the optimal mean vectors $\mathcal{W} = \{\mathbf{w}_k\}_{k=1}^K$, which minimize the L1 Expected Quantization Error (L1-EQE):

$$\mathcal{J}_1(\{\mathbf{w}_k\}) = \mathbb{E}[\|\mathbf{x} - \mathbf{w}^*\|^2; \mathbf{w}^* = \arg\min_{k=1,\ldots,K} \|\mathbf{x} - \mathbf{w}_k\|^2], \quad (3)$$

where $\mathbf{x}$ is the location of query $\mathbf{q} = [\mathbf{x}, \theta] \in \mathcal{T}$ and $\mathbf{w}_k$ is the mean center vector of all queries $\mathbf{q} \in \mathbb{Q}_k$ associated with $\mathbf{w}_k$. We adopt the K-Means [17] clustering algorithm to identify the L1 LRs based on the queries' centers $\mathbf{x}$[3]. This phase yields the K L1 cluster representatives, LRs. A limitation of the K-Means algorithm is that we need to specify K number of LRs in advance. Therefore, we devised a simple strategy to find a near-optimal number K. By running the clustering algorithm in a loop, each time increasing the input parameter K for the K-Means algorithm, we are able to find a K that is near-optimal. In this case, an optimal K would *sufficiently* minimize the Sum of Squared Quantization Errors (SSQE), which is equal to the summation of distances, of all queries from their respective LRs[4].

$$\text{SSQE} = \sum_{i}^{n} \min_{\mathbf{w}_k \in \mathcal{W}} (||\mathbf{x}_i - \mathbf{w}_k||_2^2) \quad (4)$$

This algorithm starts with an initial K value and gradually increments it until SSQE yields an improvement no more than a predefined threshold $\epsilon > 0$.

We utilize K-Means again, but this time over each L1 cluster of queries created by the L1 query quantization phase. Formally, we minimize the conditional L2 Expected Quantization Error (L2-EQE):

$$\mathcal{J}_{1.1}(u_{k,l}) = \mathbb{E}[(\theta - u_{k,*})^2]; u_{k,*} = \arg\min_{l \in L} (\theta - u_{k,l})^2 \quad (5)$$

conditioned on $\mathbf{w}_k = \mathbf{w}^*$. Therefore, for each LR $\mathbf{w}_1, \ldots, \mathbf{w}_K$, we *locally* run the K-Means algorithm with L

[2]Note that LPMs are also referred to as PLRs.

[3]The framework can employ any clustering algorithm.

[4]The algorithm is available in the extended version of the paper at: https://github.com/Skeftical/xaq-reproducibility/tree/master/Extended\%20Version

number of RRs, where a near optimal value for L is obtained following the same near-optimal strategy. With SSQE being computed using u and θ instead of $(\mathbf{w}, \mathbf{x})$. Specifically, we identify the L2 RRs over the radii of those queries from $\mathcal{T}$ whose closest LR is $\mathbf{w}_i$. Then, by executing the L-Means over the radius values from those queries we derive the corresponding set of radius representatives $\mathcal{U}_i = \{u_{i1}, \ldots, u_{iL}\}$, where each u_{il} is the mean radius of all radii in the l-th L2 sub-cluster of the i-th L1 cluster. Thus the first part of the deconstructed optimization problem can be considered as two-fold, as we wish to find optimal parameters for both LRs and RRs that minimize (3) and (5).

B. Optimization Problem 2: Fitting PLRs per Query Cluster

The second part of the deconstructed generic optimization problem has to do with fitting *optimal* approximation functions such that the local EEL is minimized given the optimal parameters obtained from the first part of the deconstructed problem. We fit (approximate) L PLR functions $\hat{f}_{kl}(\theta; \mathbf{w}_k)$ for each L2 sub-cluster $\mathbb{U}_{kl}$ for those θ values that belong to the L2 RR u_{kl}. The fitted PLR captures the *local* statistical dependency of θ around its mean radius u_{kl} given that $\mathbf{x}$ is a member of the L1 cluster represented by $\mathbf{w}_k$. Given the objective in (2), for each local L2 sub-cluster, the approximate function $\hat{f}_{kl}$ minimizes the conditional Local EEL:

$$\begin{aligned} \mathcal{J}_2(\{\beta_{kl}, \lambda_{kl}\}) &= \mathbb{E}_{\theta,\mathbf{x}}[\mathcal{L}(f_{kl}(\theta; \mathbf{w}_k), \hat{f}_{kl}(\theta; \mathbf{w}_k))] \quad (6) \\ \text{s.t.} \quad & \mathbf{w}_k = \arg\min_{j \in [K]} \|\mathbf{x} - \mathbf{w}_j\|^2, \\ & u_{kl} = \arg\min_{j \in [L]} |\theta - u_{kl}| \end{aligned}$$

conditioned on the closeness of the query's $\mathbf{x}$ and θ to the L1 and L2 quantized query space $\mathbb{Q}_k$ and $\mathbb{U}_{kl}$, respectively, where $\{\beta_{kl}, \lambda_{kl}\}$ are the parameters for the PLR local approximation explanation functions $\hat{f}_{kl}$ defined as follows in (7).

Remark 1: Minimizing objective $\mathcal{J}_2$ in (6) is not trivial due to the double conditional expectation over each query center and radius. To *initially* minimize this local objective, we use Multivariate Adaptive Regression Splines (MARS) [16] as the approximate model explanation function $\hat{f}_{kl}$. Thus, our approximate $\hat{f}_{kl}$ has the following form:

$$\hat{f}_{kl}(\theta; \mathbf{w}_k) = \beta_0 + \sum_{i=1}^{M} \beta_i h_i(\theta), \quad (7)$$

where the basis function $h_i(\theta) = \max\{0, \theta - \lambda_i\}$, with λ_i being the *hinge* points. Essentially, this creates M linear regression functions. The number M of linear regression functions is automatically derived by MARS using a threshold for convergence w.r.t R^2(*coefficient-of-determination*, later defined); which optimize fitting. Thus, guaranteeing an optimal number of M linear regression functions. For each L2 sub-cluster, we fit L MARS functions $\hat{f}_{kl}, l = 1, \ldots, L$, each one associated with an LR and an RR, thus, in this phase we initially fit $K \times L$ MARS functions for providing explanations for the whole query space. Figure 4 illustrates the two levels L1 and L2 of our explanation methodology, where each LR and RR are associated with a MARS model.

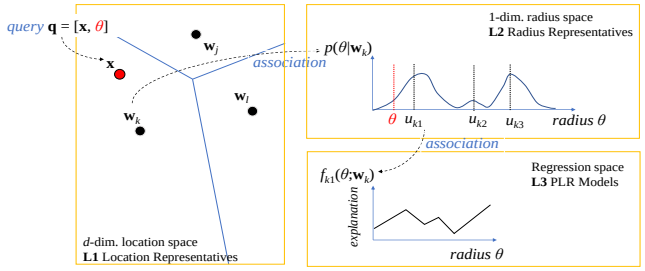


Fig. 4. The XAXA framework rationale: Query mapping to L1 cluster, then mapping to L2 sub-cluster, and association to L3 PLR regression space. The explanation is provided by the associated set of PLR functions f_{kl}.

C. Optimization Problem 3: Putting it All Together

The optimization objective functions in (3), (5) and (6) should be combined to establish our generic optimization function $\mathcal{J}_0$, which involves the estimation of the *optimal* parameters that minimize the EQE in $\mathcal{J}_1$ (L1) and $\mathcal{J}_{1.1}$ (L2), and then the conditional optimization of the parameters in $\mathcal{J}_2$. In this context, we need to estimate these values of the parameters in $\mathcal{W} = \{\mathbf{w}_k\}$ and $\mathcal{U} = \{u_{kl}\}$ that minimize the EEL given that our explanation comprises a set of regression functions $\hat{f}_{kl}$. Formally, our joint optimization is optimizing *all* the parameters from $\mathcal{J}_1$, $\mathcal{J}_{1.1}$, and $\mathcal{J}_2$ from Problem 1 and Problem 2, respectively:

$$\mathcal{J}_3(\mathcal{W}, \mathcal{U}, \mathcal{M}) = \mathcal{J}_1(\mathcal{W}) + \mathcal{J}_{1.1}(\mathcal{U}) + \mathcal{J}_2(\mathcal{M}) \quad (8)$$

with parameters:

$$\mathcal{W} = \{\mathbf{w}_k\}, \mathcal{U} = \{u_{kl}\}, \mathcal{M} = \{(\beta_i, \lambda_i)_{kl}\} \quad (9)$$

with $k \in [K]$, $l \in [L]$, $i \in [M]$, which are stored for fine tuning and explanation/prediction.

Remark 2: The optimization function $\mathcal{J}_3$ approximates the generic objective function $\mathcal{J}_0$ in (2) via L1 and L2 query quantization (referring to the integral part of (2)) and via the estimation of the local PLR functions referring to the family of function space $\mathcal{F}$. Hence, we hereinafter contribute to an algorithmic/computing solution to the optimization function $\mathcal{J}_3$ approximating the *theoretical* objective function $\mathcal{J}_0$.

VI. XAXA: Hetero-Associative Learning

The framework contributes a new hetero-associative learning model that associates the (hierarchically) clustered query space with (PLR-based explanation) functions. Given the hierarchical query space quantization and local PLR fitting, the training mode fine-tunes the parameters in (9) to optimize both $\mathcal{J}_1$, $\mathcal{J}_{1.1}$ in (3), (5) and $\mathcal{J}_2$ in (6). The three main sets of parameters $\mathcal{W}, \mathcal{U}$, and $\mathcal{M}$ of the framework are *incrementally* trained in *parallel* using queries issued against the DBMS. Therefore, (i) the analyst issues an AQ $\mathbf{q} = [\mathbf{x}, \theta]$; (ii) the

DBMS answers with result y ; (iii) our framework exploits pairs $(\mathbf{q}, y)$ to train its new statistical learning model.

Training follows three steps. First, in the *assignment* step, the incoming query $\mathbf{q}$ is associated with an L1, L2 and PLR, by finding the closest representatives at each level. The second step is the *on-line adjustment* step, where the LR, RR and PLR parameters are gradually modified w.r.t. the associated incoming query in the direction of minimizing the said objectives. Finally, the *off-line adjustment* step conditionally retrains any PLR fitting model associated with any incoming query so far. This step is triggered when a predefined retraining value is reached, or when the number of queries exceeds a threshold .

Query Assignment Step. For each executed query-answer pair $(\mathbf{q}, y)$, we project $\mathbf{q}$ to its closest L1 LR using only the query center $\mathbf{x}$ based on (3). Obtaining the projection $\mathbf{w}_k^*$ allows us to *directly* retrieve the associated RRs $\mathcal{U}_k^* = \{u_{1k}^*, \ldots, u_{Lk}^*\}$. Finding the best L2 RR in $\mathcal{U}_k^*$ is, however, more complex than locating the best LR. In choosing one of the L2 RRs, we consider both *distance* and the associated *prediction error*. Specifically, the *prediction error* is obtained by the PLR fitting models of each RR from the set $\mathcal{U}_k^*$. Hence, in this context, we first need to consider the distance of our query $\mathbf{q} = [\mathbf{x}, \theta]$ to all of the RRs in $\mathcal{U}_k^*$ focused on the absolute distance in the radius space:

$$|\theta - u_{kl}|, \forall u_{kl} \in \mathcal{U}_k^*, \tag{10}$$

and, also, the *prediction error* given by each RR's associated PLR fitting model $\hat{f}_{kl}$. The prediction error is obtained by the squared difference of the actual result y of the query $\mathbf{q}$ and the predicted outcome of the local PLR fitting model $\hat{y} = \hat{f}_{kl}(\theta; \mathbf{w}_k^*)$:

$$(y - \hat{f}_{kl}(\theta; \mathbf{w}_k^*))^2, l = 1, \ldots, L \tag{11}$$

Therefore, for assigning a query $\mathbf{q}$ to a L2 RR, we combine both distances in (10) and (11) to get the assignment distance in (12), which returns the RR in $\mathcal{U}_k^*$ which minimizes:

$$l^* = \arg\min_{l \in [L]} (z|\theta - u_{kl}| + (1 - z)(y - \hat{f}_{kl}(\theta; \mathbf{w}_k^*))^2) \tag{12}$$

The parameter z tilts our decision towards the *distance*-wise metric or the *prediction*-wise metric, depending on which aspect we wish to attach greater significance.

Remark 3: *Why incorporate prediction error?* We could associate an incoming query with the closest L2 RR as is being done with L1 LR. However, note that an explanation function may have lower prediction error even though is not the closest (w.r.t to RR). Intuitively, this holds true, as some function might be able to make better generalizations even if their RRs are further apart. Therefore, we introduce the weighted-distance in (12) to account for this and make more sophisticated selections.

On-line Representatives Adjustment Step. This step optimally adjusts the positions of the chosen LR and RR so that training is informed by the new query. Their positions are shifted using Stochastic Gradient Descent (SGD) [8] over the $\mathcal{J}_1$ and $\mathcal{J}_2$ w.r.t. $\mathbf{w}$ and θ variables in the negative direction of their gradients, respectively. This ensures the *optimization of both objective functions*. Theorems 1 and 2 present the update rule for the RR selected in (12) to minimize the EEL given that a query is projected to its L1 LR and its convergence to the median value of the radii of those queries.

Theorem 1: Given a query $\mathbf{q} = [\mathbf{x}, \theta]$ projected onto the closest L1 $\mathbf{w}_{k^*}$ and L2 u_{k^*,l^*}, the update rule for u_{k^*,l^*} that minimizes $\mathcal{J}_2$ is:

$$\Delta u_{k^*,l^*} \leftarrow \alpha z \text{sgn}(\theta - u_{k^*,l^*}) \tag{13}$$

Proof 1: See the proof in extended version of the paper.

$\alpha \in (0, 1)$ is the learning rate defining the shift of θ ensuring convergence to optimal position and $sgn(x) = \frac{d|x|}{dx}, x \neq 0$ is the signum function. Given that query $\mathbf{q}$ is projected on L1 $\mathbf{w}_k^*$ and on L2 u_{k^*,l^*}, the corresponding RR converges to the local median of all radius values of those queries.

Theorem 2: Given the optimal update rule in (13) for L2 RR $u_{k,l}$, it converges to the median of the θ values of those queries projected onto the L1 query subspace $\mathbb{Q}_k$ and the L2 sub-cluster $\mathbb{U}_{kl}$, i.e., for each query $\mathbf{q} = [\mathbf{x}, \theta]$ with $\mathbf{x} \in \mathbb{Q}_k$, it holds true for $u_{kl} \in \mathbb{U}_{kl}$ that: $\int_0^{u_{kl}} p(\theta|\mathbf{w}_k) d\theta = \frac{1}{2}$.

Proof 2: See the proof in extended version of the paper.

Using SGD, θ_{k^*,l^*} converges to the median of all radius values of all queries in the local L2 sub-cluster in an on-line manner.

Off-line PLR Adjustment Step. The mini-batch adjustment step is used to conditionally re-train the PLRs to reflect the changes by (13) in $\mathcal{U}_k$ parameters. As witnessed earlier, representatives are incrementally adjusted based on the projection of the incoming query-answer pair onto L1 and L2 levels. For the PLR/MARS functions, the adjustment of hinge points and parameters (β_i, λ_i) needs to happen in mini-batch mode taking into consideration the projected incoming queries onto the L2 level. In order to achieve this, we keep track of the number of projected queries on each L2 sub-cluster $\mathcal{U}_k$ and re-train the corresponding parameters $(\beta_{kli}, \lambda_{kli})$ PLR/MARS of the fitting $\hat{f}_{kl}$ given a conditionally optimal L2 RR u_{kl} and for every processed query we increment a counter. Once we reach a predefined number of projected queries-answers, we re-train every PLR/MARS model that was affected by projected training pairs. Once the fitting models are retrained, then a new era of on-line adjustment begins until the end of the training pairs or the convergence of the L2 RRs.

VII. XAXA: Explanation Serving

After *Pre-Processing* and *Training*, explanations can be provided for *unseen* AQs. The explanations are *predicted* by the fusion of the trained/fitted PLRs based on the incoming queries. The process is as follows. The analyst issues a query $\mathbf{q}$, $q \notin \mathcal{T}$. To obtain the explanation of the result for the given query $\mathbf{q} = [\mathbf{x}, \theta]$, XAXA finds the closest $\mathbf{w}_k^*$ LR and then directly locates all of the associated RRs of the $\mathcal{U}_k^*$. The provided explanation utilizes all *selected* associated L PLRs fitting models $\hat{f}_{k^*,l}$. The selection of the most relevant PLR functions for explaining $\mathbf{q}$ is based on the boolean indicator

$I(\theta, u_{k^*,l})$, where $I(\theta, u_{k^*,l}) = 1$ if we select to explain AQ based on the area around θ represented by the L2 RR $u_{k^*,l}$; 0 otherwise. Specifically this indicator is used to denote which is the domain value for the θ radius in order to deliver the dependency of the answer y within this domain as reflected by the corresponding PLR $\hat{f}_{k^*,l}(\theta; \mathbf{w}_{k^*})$. In other words, for different query radius values, XAXA selects different PLR models to represent the AQ, which is associated with the RR that is closer to the given query's θ. Using this method, L possible explanations for $\mathbf{q}$ exist and the selection of the most relevant PLR fitting model for the current radius domain is determined when $I(\theta, u_{k^*,l}) = 1$ and, simultaneously, when:

$$\begin{cases} 0 \leq \theta < u_{k^*,1} + \frac{1}{2}|u_{k^*,1} - u_{k^*,2}| \text{ and } l = 1 \\ u_{k^*,l-1} + \frac{1}{2}|u_{k^*,l-1} + u_{k^*,l}| \leq \theta < u_{k^*,l} + \frac{1}{2}|u_{k^*,l} + u_{k^*,l+1}| \\ \qquad \text{and } l < L \\ u_{k^*,l-1} + \frac{1}{2}|u_{k^*,l-1} + u_{k^*,l}| \leq \theta \text{ and } l = L \end{cases} \tag{14}$$

Interpretation. The domain radius in (14) shows the radius interval boundaries for selecting the most relevant PLR fitting model for explanation in the corresponding radius domain. In other words, we switch between PLR models according to the indicator function. For instance, from radius 0 up to the point where the first RR would still be the closest representative, indicated as $\leq u_1 + \frac{1}{2}|u_1 - u_2|$, we use the first PLR fitting model. Using this selection method, we can derive an accurate explanation for the AQ. The returned explanation is then represented by (15):

$$\sum_{l=1}^{L} I(\theta, u_{k^*,l}) \hat{f}_{k^*,l} \tag{15}$$

which is a PLR function where the indicator $I(\cdot, \cdot)$ returns 1 only for the selected PLR.

VIII. Experimental Evaluation

A. Experimental Setup & Metrics

Data Sets and Query Workloads: The real dataset $\mathcal{B}_1 = \{\mathbf{x}_i\}_{i=1}^{N}, \mathbf{x} \in \mathbb{R}^2$ with cardinality $|\mathcal{B}_1| = N = 6 \cdot 10^6$ contains crimes for the city of Chicago [1]. We also used another real dataset $\mathcal{B}_2 = \{\mathbf{x}\}$ where $\mathbf{x} \in \mathbb{R}^2$ and $|\mathcal{B}_2| = 5.6 \cdot 10^6$ contains records of calls issued from different locations in Milan [2][5]. We create a synthetic workload $\mathcal{T}$ containing $m = 5 \cdot 10^4$ queries and their answers, i.e., $\{(\mathbf{q}, y)_i\}_{i=1}^{m} = \mathcal{T}$. Each query is a 3-d vector $\mathbf{q} = [\mathbf{x}, \theta]$ with answer y where $\mathbf{x} = [x_1, x_2] \in \mathbb{R}^2$ is the center, $\theta \in \mathbb{R}$ is its radius and $y \in \mathbb{R}$ the result against real dataset $\mathcal{B}_1$. We scale $\mathcal{B}_1$ and $\mathcal{T}$ in all their dimensions, restricting their values in $[0, 1]$. We use workload $\mathcal{T}$ for *Pre-Processing* and *Training* and create a *separate* evaluation set $\mathcal{V}$ containing $|\mathcal{V}| = 0.2 \cdot m$ new query-answer pairs.

Query Workload $(\mathcal{T}, \mathcal{V})$ Distributions: It is important to mention that no real-world benchmarks exist for exploratory analytics in general; hence we resort to synthetic workloads.

[5]The complete set of results were omitted due to space restrictions, we note that the results were similar to real dataset $\mathcal{B}_1$

TABLE I
Synthetic Query Workloads $\mathcal{T}, \mathcal{V}$.

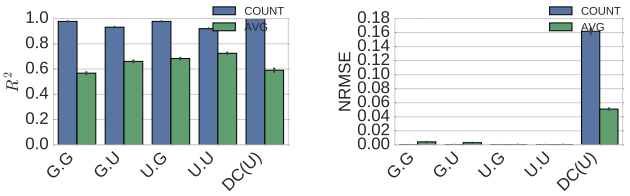
	Location $\mathbf{x}$	**Radius θ**
Gauss-Gauss	$\sum_{i=1}^{C} \mathcal{N}(\mu_i, \Sigma_i)$	$\sum_{i=1}^{J} \mathcal{N}(\mu_i, \sigma_i^2)$
Gauss-Uni	$\sum_{i=1}^{C} \mathcal{N}(\mu_i, \Sigma_i)$	$\sum_{i=1}^{J} U(v_i, v_i + 0.02)$
Uni-Gauss	$\sum_{i=1}^{C} U(v_i, v_i + 0.04)$	$\sum_{i=1}^{J} \mathcal{N}(\mu_i, \sigma_i^2)$
Uni-Uni	$\sum_{i=1}^{C} U(v_i, v_i + 0.04)$	$\sum_{i=1}^{J} U(v_i, v_i + 0.02)$
DC (U)	$U(0, \lvert D \rvert)$	$U(0, \lvert D \rvert)$

The same fact is recognized in [34]. For completeness, therefore, we will also use their workload as well in our evaluation. Please note that the workload from [34] depicts different exploration scenarios, based on zooming-in queries. As such, they represent a particular type of exploration, where queried spaces subsume each other, representing a special-case scenario for the overlaps in the workloads for which XAXA was designed. However, as it follows the same general assumptions we include it for completeness and as a sensitivity-check for XAXA workloads. We first generated $\mathcal{T}, \mathcal{V}$ with our case studies in mind. We used 2-dim./1-dim. Gaussian and 2-dim./1-dim. Uniform distributions for query locations $\mathbf{x}$ and radius θ of our queries, respectively. Hence, we obtain four variants of workloads, as shown in Table I. We also list the workload borrowed from [34] as DC(U).[6]

The parameters C and J signify the number of mixture distributions within each variant. Multiple distributions comprise our workload as we desire to simulate the analysts' behavior issuing queries against the real dataset $\mathcal{B}_1$. For instance, given a number $C \times J$ of analysts, each one of them might be assigned to different geo-locations, hence parameter C, issuing queries with different radii, hence J, given their objectives. For the parameters μ (mean) and v of the location distributions in Table I, we select points uniformly at random ranging in [0,1]. The covariance Σ_i of each Gaussian distribution, for location $\mathbf{x}$, is set to 0.0001. The number of distributions/query-spaces was set to $C \times J$, where $C \in \mathbb{N}$ and $J \in \mathbb{N}$ with $C = 5$ and $J = 3$, thus, a mixture of 15 query distributions. The radii covered $2\% - 20\%$ of the data space, i.e., μ_i for θ was randomly selected in $[0.02, 0.2]$ for Gaussian distributions, with small σ^2 for the Gaussian distributions, and v_i was in $[0.02, 0.18]$ for Uniform distributions. Further increasing that number would mean that the queries generated would cover a much greater region of the space, thus, making the learning task much easier. The variance σ_i^2 for Gaussian distributions of θ was set to 0.0009 to leave no overlap with the rest of the $J - 1$ distributions. We deliberately leave no overlapping within θ distributions as we assume there is a multi-modal mixture of distributions with different radii used by analysts. Parameters z and α were set to $z = 0.5$ to ensure equal importance during the query assignment step and stochastic gradient learning schedule $\alpha = 0.01$.

Our implementation was in Python 2.7. Experiments ran single threaded on a Linux Ubuntu 16.04 using an i7 CPU at 2.20GHz with 6GB RAM. For every query in the evaluation set $\mathcal{V}$, we take its radius and generate n evenly spaced radii over

[6]$|D|$ is the cardinality of the data-set generated by [34].



(a) Results for R^2. (b) Results for *NRMSE*.

Fig. 5. Goodness of fit and predictive accuracy results.

the interval $[0.02, \theta']$, where 0.02 is the minimum radius used. For query $q = [\mathbf{x}, \theta]$ we generate and find the answer for n sub-queries, $ST = \{([\mathbf{x}, \theta_1], y_1), \cdots ([\mathbf{x}, \theta_n], y_n)\}$. The results of these queries constitute the *Actual Explanation* (AE) and hence the *true* function is represented as a collection of n pairs of sub-radii and y, $\{(\theta_i, y_i)\}_{i=1}^n$. The approximated function is the collection of n sub-radii and predicted $\hat{y}$, $\{(\theta_i, \hat{y}_i)\}_{i=1}^n$.

Evaluation & Performance Metrics:

Goodness of Fit: The EEL is measured here using the coefficient of determination R^2. This metric indicates how much of the *variance* generated by $f(\theta; \mathbf{x})$ can be explained using the approximation $\hat{f}(\theta; \mathbf{x})$. This represents the goodness-of-fit of an approximated explanation function over the actual one. It is computed using : $R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$, in which the denominator is proportional to the *variance* of the *true* function and the numerator are the residuals of our approximation. The EEL between f and $\hat{f}$ explanations can then be computed as $\mathcal{L}(f(\theta; \mathbf{x}), \hat{f}(\theta; \mathbf{x})) = 1 - R^2$.

Predictive Accuracy: To completely appreciate the accuracy and usefulness of XAXA, we also quantify the predictive accuracy associated with explanation (regression) functions. We employ the *Normalized-Root-Mean-Squared-Error(NRMSE)* for this purpose: NRMSE $= \frac{1}{y_{max} - y_{min}} (\frac{1}{n} \sum_i^n (y_i - \hat{y}_i)^2)$. Essentially, this shows how accurate the results would be if an analyst used the explanation (regression) functions for further data exploration, without issuing DB queries.

B. Experimental Results: Accuracy

For our experiments we chose to show performance and accuracy results over two representative aggregate functions: `COUNT` and `AVG` due to their extensive use and because of their properties, with `COUNT` being monotonically increasing and `AVG` being non-linear. As baselines do not exist for this kind of problem, we demonstrate the accuracy of our framework with the bounds of the given metrics as our reference. Due to space limitations, we show only representative results– results with additional datasets and experiments are given in the extended version of the paper along with the scripts used, for *reproducibility* at [7].

Figure 5(a) shows the results for R^2 over all workloads for `AVG` and `COUNT`. We report on the *average* R^2 found by evaluating the explanation functions given for the evaluation set $\mathcal{V}$. Overall, we observe high accuracy across all workloads and aggregate functions. Meaning that our approximate explanation function can explain most of the variation observed by the *true* function. Figure 5(a) also shows the *standard-deviation* for R^2 within the evaluation set, which appears to be minimal. Note that accuracy for `COUNT` is higher than for `AVG`; this is expected as `AVG` fluctuates more than `COUNT`, the latter being monotonically increasing. If the *true* function is highly non-linear, the score for this metric deteriorates as witnessed by the decreased accuracy for `AVG`. Hence, it should be consulted with care for non-linear functions this is why we also provide measurements for *NRMSE*. Also, it is important to note that if the *true* function is constant, meaning the rate of change is 0, then the score returned by R^2 is 0. Thus, in cases where such *true* functions exist, R^2 should be avoided as it would incorrectly label our approximation as inaccurate. We have encountered many such *true* functions especially for small radii in which no further change is detected.

As statistical measures for model fitness can be hard to interpret, Figure 5(b) provides results for *NRMSE* when using XAXA explanation (regression) functions for predictions to AQ queries (avoiding accessing the DBMS). Overall, the predictive accuracy is shown to be excellent for all combinations of Gaussian-Uniform distributed workloads and for both aggregate functions. Even for the worst-case workload of DC(U), accuracy (NRMSE) is below 4% for `AVG` and below 18% for `COUNT`. As such, an analyst can consult this information to decide on whether to use the approximated function for subsequent computation of AQs to gain speed, instead of waiting idle for an AQ to finish execution.

C. Experimental Results: Efficiency and Scalability

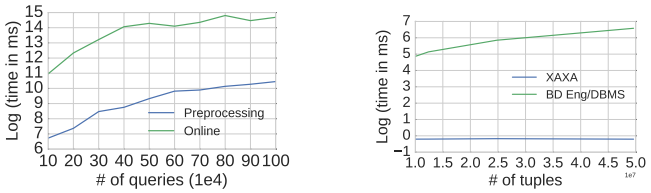
1) XAXA is invariant to data size changes: XAXA remains invariant to data size changes as shown in Figure 6 (b). Dataset $\mathcal{B}_1$ was scaled up by increasing the number of data points in $\mathcal{B}_1$ respecting the original distribution. As a reference we show the time needed to compute aggregates for $\mathcal{B}_1$ using a DBMS/Big Data engine.

2) Scalability of XAXA: We show that XAXA scales linearly with some minor fluctuations attributed to the inner workings of the system and different libraries used. Evidently, both *Pre-Processing* stage and *Training* stage do not require more than a few minutes to complete. We note that *Training* stage happens on-line, however we test the stage off-line to demonstrate its scalability; results are shown in Figure 6(a).

IX. Conclusions

We have defined a novel class of explanations for Aggregate Queries, which are of particular importance for exploratory analytics. The proposed AQ explanations are succinct, assuming the form of functions. As such, they convey rich information to analysts about the queried data subspaces and as to how the aggregate value depends on key parameters of the queried space, helping to guide analysts in their further explorations. Furthermore, they allow analysts to utilize these explanation functions for their explorations without the need to issue more

[7] https://github.com/Skeftical/xaq-reproducibility/tree/master/Extended\%20Version



(a) *Pre-Processing & Training*. (b) Time vs. $|\mathcal{B}_1|$ increase.

Fig. 6. Scalability of XAXA in terms of explanation time and data set size.

(potentially very) expensive queries to the DBMS. Thus, given the importance of statistical analyses and exploratory analytics, these explanations can significantly empower in-DBMS analytics for data exploration tasks. We have formulated the problem of deriving AQ explanations as a joint optimization problem and have provided novel statistical/machine learning models for its solution. The proposed scheme for computing explanations does not require DBMS data accesses (after model training) ensuring efficiency and scalability.

REFERENCES

[1] Crimes - 2001 to present. URL: https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2, 2016. Accessed: 2016-12-01.

[2] Telecommunications - sms, call, internet - mi. URL: https://dandelion.eu/datagems/SpazioDati/telecom-sms-call-internet-mi/, 2017. Accessed: 2016-12-01.

[3] 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 *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 29–42. ACM, 2013.

[4] Y. Amsterdamer, D. Deutch, and V. Tannen. Provenance for aggregate queries. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 153–164. ACM, 2011.

[5] C. Anagnostopoulos and P. Triantafillou. Learning set cardinality in distance nearest neighbours. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 691–696. IEEE, 2015.

[6] C. Anagnostopoulos and P. Triantafillou. Efficient scalable accurate regression queries in in-dbms analytics. In *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*, pages 559–570. IEEE, 2017.

[7] P. Bailis, E. Gan, S. Madden, D. Narayanan, K. Rong, and S. Suri. Macrobase: Analytic monitoring for the internet of things. *arXiv preprint arXiv:1603.00567*, 2016.

[8] L. Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

[9] A. Chalamalla, I. F. Ilyas, M. Ouzzani, and P. Papotti. Descriptive and prescriptive data cleaning. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 445–456. ACM, 2014.

[10] S. Chaudhuri, G. Das, and V. Narasayya. Optimized stratified sampling for approximate query processing. *ACM Transactions on Database Systems (TODS)*, 32(2):9, 2007.

[11] S. Chaudhuri, G. Das, and U. Srivastava. Effective use of block-level sampling in statistics estimation. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 287–298. ACM, 2004.

[12] J. Cheney, L. Chiticariu, W.-C. Tan, et al. Provenance in databases: Why, how, and where. *Foundations and Trends® in Databases*, 1(4):379–474, 2009.

[13] G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1–3):1–294, 2012.

[14] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[15] K. El Gebaly, P. Agrawal, L. Golab, F. Korn, and D. Srivastava. Interpretable and informative explanations of outcomes. *Proceedings of the VLDB Endowment*, 8(1):61–72, 2014.

[16] J. H. Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

[17] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[18] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *ACM SIGMOD Record*, volume 26, pages 171–182. ACM, 1997.

[19] B. Huang, S. Babu, and J. Yang. Cumulon: Optimizing statistical data analysis in the cloud. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 1–12. ACM, 2013.

[20] S. Idreos, O. Papaemmanouil, and S. Chaudhuri. Overview of data exploration techniques. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 277–281. ACM, 2015.

[21] S. Jain, D. Moritz, D. Halperin, B. Howe, and E. Lazowska. Sqlshare: Results from a multi-year sql-as-a-service experiment. In *Proceedings of the 2016 International Conference on Management of Data*, pages 281–293. ACM, 2016.

[22] N. Khoussainova, M. Balazinska, and D. Suciu. Perfxplain: debugging mapreduce job performance. *Proceedings of the VLDB Endowment*, 5(7):598–609, 2012.

[23] S. Krishnan and E. Wu. Palm: Machine learning explanations for iterative debugging. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*, page 4. ACM, 2017.

[24] M. L. K. L. Sidirourgos and P. A. Boncz. Sciborq: Scientific data management with bounds on runtime and quality. In *Proceedings of Conference on Innovative Data Systems, (CIDR)*, 2011.

[25] A. Meliou, S. Roy, and D. Suciu. Causality and explanations in databases. *Proceedings of the VLDB Endowment*, 7(13):1715–1716, 2014.

[26] V. Nair, A. Raul, S. Khanduja, V. Bahirwani, Q. Shao, S. Sellamanickam, S. Keerthi, S. Herbert, and S. Dhulipalla. Learning a hierarchical monitoring system for detecting and diagnosing service issues. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2029–2038. ACM, 2015.

[27] Y. Park, A. S. Tajik, M. Cafarella, and B. Mozafari. Database learning: Toward a database that becomes smarter every time. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 587–602. ACM, 2017.

[28] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

[29] S. Roy, L. Orr, and D. Suciu. Explaining query answers with explanation-ready databases. *Proceedings of the VLDB Endowment*, 9(4):348–359, 2015.

[30] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*, 2017.

[31] A. S. Szalay, J. Gray, A. R. Thakar, P. Z. Kunszt, T. Malik, J. Raddick, C. Stoughton, and J. vandenBerg. The sdss skyserver: public access to the sloan digital sky server data. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 570–581. ACM, 2002.

[32] X. Wang, X. L. Dong, and A. Meliou. Data x-ray: A diagnostic tool for data errors. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1231–1245. ACM, 2015.

[33] X. Wang, A. Meliou, and E. Wu. Qfix: Diagnosing errors through query histories. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1369–1384. ACM, 2017.

[34] A. Wasay, X. Wei, N. Dayan, and S. Idreos. Data canopy: Accelerating exploratory statistical analysis. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 557–572. ACM, 2017.

[35] E. Wu and S. Madden. Scorpion: Explaining away outliers in aggregate queries. *Proceedings of the VLDB Endowment*, 6(8):553–564, 2013.

[36] S. Wu, B. C. Ooi, and K.-L. Tan. Continuous sampling for online aggregation over multiple queries. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 651–662. ACM, 2010.