Query-Driven Learning for Predictive Analytics of Data Subspace Cardinality

CHRISTOS ANAGNOSTOPOULOS and PETER TRIANTAFILLOU, University of Glasgow

Fundamental to many predictive analytics tasks is the ability to estimate the cardinality (number of data items) of multi-dimensional data subspaces, defined by query selections over datasets. This is crucial for data analysts dealing with, e.g., interactive data subspace explorations, data subspace visualizations, and in query processing optimization. However, in many modern data systems, predictive analytics may be (i) too costly money-wise, e.g., in clouds, (ii) unreliable, e.g., in modern Big Data query engines, where accurate statistics are difficult to obtain/maintain, or (iii) infeasible, e.g., for privacy issues. We contribute a novel, query-driven, function estimation model of analyst-defined data subspace cardinality. The proposed estimation model is highly accurate in terms of prediction and accommodating the well-known selection queries: multi-dimensional range and distance-nearest neighbors (radius) queries. Our function estimation model: (i) quantizes the vectorial query space, by learning the analysts' access patterns over a data space, (ii) associates query vectors with their corresponding cardinalities of the analyst-defined data subspaces, (iii) abstracts and employs query vectorial similarity to predict the cardinality of an unseen/unexplored data subspace, and (iv) identifies and adapts to possible changes of the query subspaces based on the theory of optimal stopping. The proposed model is decentralized, facilitating the scaling-out of such predictive analytics queries. The research significance of the model lies in that (i) it is an attractive solution when data-driven statistical techniques are undesirable or infeasible, (ii) it offers a scale-out, decentralized training solution, (iii) it is applicable to different selection query types, and (iv) it offers a performance that is superior to that of data-driven approaches.

CCS Concepts: • **Information systems** → **Data management systems**; • **Computing methodologies** → **Supervised learning**; **Unsupervised learning**; • **Mathematics of computing** → *Mathematical optimization;*

Additional Key Words and Phrases: Predictive analytics, predictive learning, data subspace exploration, analytics selection queries, vector regression quantization, optimal stopping theory

ACM Reference Format:
Christos Anagnostopoulos and Peter Triantafillou. 2017. Query-driven learning for predictive analytics of data subspace cardinality. ACM Trans. Knowl. Discov. Data 11, 4, Article 47 (June 2017), 46 pages.
DOI: http://dx.doi.org/10.1145/3059177

1. INTRODUCTION

Recent R&D efforts have been dominated by efforts to accommodate large datasets with frameworks that enable highly scalable distributed data exploration and predictive analytics. Platforms such as Hadoop/MapReduce [White 2009], Yarn [Vavilapalli et al. 2013], Spark [Zaharia et al. 2012], Stratosphere [Alexandrov et al. 2014], and AsterixDB [Alsubaiee et al. 2014] are nowadays the infrastructures of choice, where

Authors' addresses: C. Anagnostopoulos (corresponding author) and P. Triantafillou, School of Computing Science, Sir Alwyn Williams Building, University of Glasgow, Glasgow G12 8RZ, Scotland, United Kingdom; emails: {christos.anagnostopoulos, peter.triantafillou}@glasgow.ac.uk.

DOI: http://dx.doi.org/10.1145/3059177

large datasets are partitioned across large numbers of data nodes, each working in parallel. In predictive analytics [Lin et al. 2014], data exploration is commonly based on aggregation operators (e.g., count, average, and median) over the results of analytics queries. Such queries typically involve datasets (which may themselves be the result of linking of other different datasets) and a number of selection predicates (e.g., range predicates over multi-dimensional vectors), which are important for predictive analytics and exploratory analysis [Atanasov et al. 2012; Balac et al. 2013]. As (i) query-driven data exploration is becoming increasingly important in the presence of large-scale data [Gosink et al. 2011], and (ii) answering (aggregations over) analytics queries is a fundamental exploration task [Chaudhuri et al. 2014], it is important to study how to process them in modern scale-out data systems. In this context, it is important to consider both query types used by analysts: multi-dimensional *range queries* and *distance nearest neighbors* (radius) queries. In these analytics queries, the analyst defines a data subspace of interest with very specific boundaries, e.g., hyper-rectangles in the case of range queries and hyper-spheres in the case of radius queries.

The current data-driven approaches for this problem are undesirable in several emerging environments: In cloud deployments, continuous data access to derive and maintain accurate statistics and related structures may be too costly in terms of financial costs. Furthermore, in several modern computational environments accurate data statistics over distributed datasets are impossible to obtain/maintain since the query processing system does not "own" the data. This applies, for instance, in the SQL(-like) query engines built on top of Spark or Hadoop, such as Cloudera's Impala, IBM's Big SQL, Teradata's Hadapt, SparkSQL, and the like. Finally, raw data accesses may be completely infeasible, e.g., in environments of data federations with sensitive data and with multiple data owners who refuse access to raw data and only agree to provide analytics and/or aggregated answers. As a case in point, the National Institute of Statistical Sciences (NISS) has developed a methodology to perform statistical analyses that, in effect, integrate data in multiple distributed databases, but without literally bringing the data together in one place, especially for data confidentiality and access restriction reasons [Karr 2010]. This of course includes inability to perform globally (across all data) exploratory analyses and visualizations. Many government, industrial, and academic investigations require exploratory analyses based on multiple distributed databases, often each with a different owner. But, confidentiality, data transfer cost, and data heterogeneity barriers to the actual data integration and exploratory analytics are numerous. Therefore, predictive aggregate-based analytics solutions, which are widely applicable in all environments are highly desirable.

1.1. Definitions

Consider a d-dimensional data space $\mathbb{R}^d$ and a dataset $\mathcal{B}$ of data points (vectors) $\mathbf{x} \in \mathbb{R}^d$.

Definition 1.1 (*L_p Norm*). Given two d-dimensional row vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$, the p-norm (L_p) distance is $\|\mathbf{x} - \mathbf{x}'\|_p = (\sum_{i=1}^{d} |x_i - x_i'|^p)^{\frac{1}{p}}$ for $1 \leq p < \infty$, and $\|\mathbf{x} - \mathbf{x}'\|_\infty = \max_{i=1,\ldots,d}\{|x_i - x_i'|\}$ for $p = \infty$. For $p = 2$, L_2 is the Euclidean distance.

Definition 1.2 (*Range Data Subspace*). A range data subspace $\mathbb{D}(\mathbf{a}, \mathbf{b}) \subset \mathbb{R}^d$ is defined by an inclusive subset of the dataset $\mathcal{B}$ such that $\mathbb{D}(\mathbf{a}, \mathbf{b}) = \{\mathbf{x} \in \mathcal{B} : a_i \leq x_i \leq b_i\}$ with two row boundary vectors $\mathbf{a} = [a_1, \ldots, a_d]$ and $\mathbf{b} = [b_1, \ldots, b_d]$ with $a_i \leq b_i$, $a_i, b_i \in \mathbb{R}$.

The geometric representation of the $\mathbb{D}(\mathbf{a}, \mathbf{b})$ is a hyper-rectangle defined by the $\mathbf{a}$ and $\mathbf{b}$ vectors with phases parallel to the axes, as shown in Figure 1 (left).

Definition 1.3 (*Range Query*). A range query $\mathbf{q} \in \mathbb{R}^{2d}$ over a dataset $\mathcal{B}$ is defined by the $2d$-dimensional row vector $\mathbf{q} = [\mathbf{a}, \mathbf{b}] = [a_1, \ldots, a_d, b_1, \ldots, b_d]$, where a_i and b_i

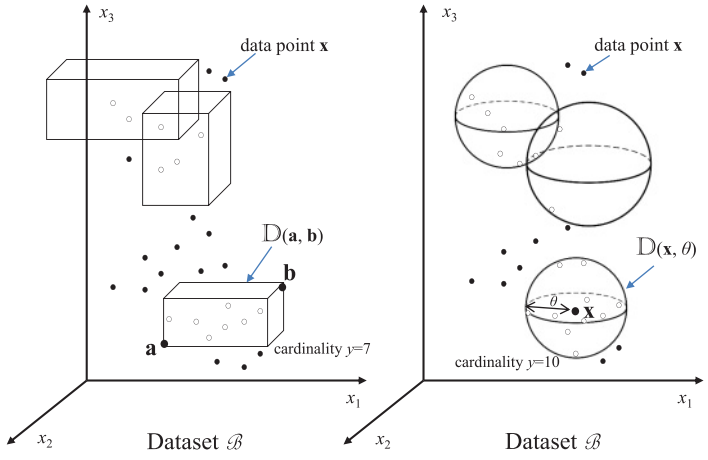


Fig. 1. (Left) A dataset $\mathcal{B}$ in a 3-dimensional space with three 6-dimensional range queries $\mathbf{q} \in \mathbb{R}^6$ defining the subsets $\mathbb{D}(\mathbf{a}, \mathbf{b})$ and $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$ (represented as 3D rectangles). The data points in white dots correspond to the data points returned by the range queries. (Right) A dataset $\mathcal{B}$ in a 3-dimensional space with three 4-dimensional radius queries $\mathbf{q} \in \mathbb{R}^4$ defining the subsets $\mathbb{D}(\mathbf{x}, \theta)$ and $\mathbf{x} \in \mathbb{R}^3$ (represented as spheres). The data points in white dots correspond to the data points returned by the radius queries.

are the lower and higher values, respectively, for attribute i. The range query retrieves the subset of the range data subspace $\mathbb{D}(\mathbf{a}, \mathbf{b})$. We notate by $\mathbb{Q} \subseteq \mathbb{R}^{2d}$ the range query vectorial space.

Definition 1.4 (*Range Query Overlap*). The range queries $\mathbf{q}, \mathbf{q}' \in \mathbb{R}^{2d}$, which define the range data subspaces $\mathbb{D}(\mathbf{a}, \mathbf{b})$ and $\mathbb{D}(\mathbf{a}', \mathbf{b}')$, respectively, overlap if for the Boolean indicator $\mathcal{A}(\mathbf{q}, \mathbf{q}') \in \{\texttt{TRUE}, \texttt{FALSE}\}$ it holds true that the logical operation $\mathcal{A}(\mathbf{q}, \mathbf{q}') = \bigwedge_{1 \leq i \leq d} ((a_i \leq b'_i) \wedge (b_i \geq a'_i)) = \texttt{TRUE}$.

Definition 1.5 (*Radius Data Subspace*). A radius data subspace $\mathbb{D}(\mathbf{x}, \theta) \subset \mathbb{R}^d$ is defined by an inclusive subset of the dataset $\mathcal{B}$ such that $\mathbb{D}(\mathbf{x}, \theta) = \{\mathbf{x}' \in \mathcal{B} : \|\mathbf{x} - \mathbf{x}'\|_p \leq \theta\}$ under L_p.

The geometrical representation of a radius data subspace $\mathbb{D}(\mathbf{x}, \theta)$ is a hyper-sphere in the d-dimensional space defined by a center row vector $\mathbf{x} \in \mathbb{R}^d$ and radius $\theta \in \mathbb{R}$, $\theta > 0$, as shown in Figure 1 (right).

Definition 1.6 (*Radius Query*). A radius query $\mathbf{q} \in \mathbb{R}^{d+1}$ over dataset $\mathcal{B}$ is represented by the $(d+1)$-dimensional row vector $\mathbf{q} = [\mathbf{x}, \theta]$, which retrieves the subset of the radius data subspace $\mathbb{D}(\mathbf{x}, \theta)$ under L_p.

We notate by $\mathbb{Q} \subseteq \mathbb{R}^{d+1}$ the radius query vectorial space.

If $p = \infty$, a radius query $\mathbf{q} = [\mathbf{x}, \theta]$ is represented by a hyper-rectangle centered at $\mathbf{x}$ with extent 2θ on each dimension. If $p = 2$, the radius query region is represented by a hyper-sphere centered at $\mathbf{x}$ with radius θ.

Definition 1.7 (*Radius Query Overlap*). The radius queries $\mathbf{q}, \mathbf{q}' \in \mathbb{R}^{d+1}$, which define the radius data subspaces $\mathbb{D}(\mathbf{x}, \theta)$ and $\mathbb{D}(\mathbf{x}', \theta')$, respectively, overlap if for the Boolean indicator $\mathcal{A}(\mathbf{q}, \mathbf{q}') \in \{\texttt{TRUE}, \texttt{FALSE}\}$ it holds true that the logical operation $\mathcal{A}(\mathbf{q}, \mathbf{q}') = (\|\mathbf{x} - \mathbf{x}'\|_p \leq \theta + \theta') = \texttt{TRUE}$.

Definition 1.8 (*Query L_2 Distance*). The L_2^2 distance/dissimilarity between range queries $\mathbf{q}$ and $\mathbf{q}'$ is defined as $\|\mathbf{q}-\mathbf{q}'\|_2^2 = \sum_{i=1}^{d}(a_i - a_i')^2 + (b_i - b_i')^2$. The L_2^2 distance between between radius queries $\mathbf{q}$ and $\mathbf{q}'$ is defined as $\|\mathbf{q}-\mathbf{q}'\|_2^2 = \sum_{i=1}^{d}(x_i - x_i')^2 + (\theta - \theta')^2$.

Given a range query $\mathbf{q} = [\mathbf{a}, \mathbf{b}]$ and a radius query $\mathbf{q} = [\mathbf{x}, \theta]$ over dataset $\mathcal{B}$, the cardinality $y \in \mathbb{N}$ is $y = |\{\mathbf{x} : \mathbf{x} \in \mathbb{D}(\mathbf{a}, \mathbf{b})\}|$ and $y = |\{\mathbf{x}' : \mathbf{x}' \in \mathbb{D}(\mathbf{x}, \theta\}|$, respectively. Note that $|\mathcal{B}|$ denotes the cardinality of the set $\mathcal{B}$.

The reader could refer to Table III in Appendix G for nomenclature. Figure 1 illustrates three range and radius queries in a $d = 3$-dimensional space over a dataset $\mathcal{B}$.

1.2. Motivation

We focus on the cardinality prediction of an analyst-defined data subspace through both range and radius queries. This is a fundamental task in data management and predictive analytics, e.g., for data mining, query-driven data exploration, time series analysis, and visualization tasks. In this context, analysts define specific data regions of a large dataset that are worth exploring by issuing (range/radius) queries and derive significant statistics metrics on the populations of these regions like the cardinality of these populations. In addition to being an important aggregation operator, in database systems, cardinality estimation can empower query optimizers to choose, for instance, the access plan that produces the smaller intermediate-query results (which have to be retrieved from disks and communicated over the network) saving time, resource waste, and money (e.g., in clouds).

Cardinality prediction is of high importance for resource utilization under budget constraints in distributed data systems. Consider K data servers, S_k, each one storing a big (partition of a) dataset $\mathcal{B}_k, k = 1, \ldots, K$, or in more compact notation $k \in [K]$. Assume also that the cost for communicating (e.g., to access and transfer data) with each data node is known, say c_k, expressed as monetary units per amount of data in a cloud setting. Given a budget of at most $\mathcal{C}$ monetary units per query, we wish to obtain the largest possible subset of the query's answer from the S_k nodes without exceeding our budget $\mathcal{C}$. That is, we have to identify which data nodes to engage in executing the incoming analytics query given $\mathcal{C}$. Hence, we desire to maximize $\sum_{k=1}^{K} y_k o_k$ subject to $\sum_{k=1}^{K} c_k o_k \leq \mathcal{C}$ and $o_i \in \{0, 1\}$ represents the engagement of node S_k to execute the query and return its data that satisfy the query. This problem (a.k.a. the 0–1 knapsack problem) can be solved (in $O(K\mathcal{C})$ time using dynamic programming) once we have predicted the cardinality y_k for each data node given a analytics query. Then, we are able to determine which data nodes to engage.

Well-established and widely adopted techniques for Approximate aggregation-Query Processing (AQP) based on sampling, histograms, self-tuning histograms, wavelets, and sketches [Cormode et al. 2012] have been proposed and can be used for cardinality prediction. Their fundamental and naturally acceptable assumption is that the underlying data are always accessible, available, and, thus, it is feasible to create and maintain their statistical structures. For instance, histograms [Gunopulos et al. 2005] require scanning of all data to be constructed and be kept up-to-date. The self-tuning histograms [Srivastava et al. 2006] require additionally the execution of queries to fine tune their statistical structures. The sampling methods [Olken and Rotem 1990] execute the queries over the sample to extrapolate the cardinality prediction result. But as aforementioned, such data-driven approaches are undesirable or infeasible in several environments. There are cases where data access to these nodes' data are restricted, e.g., distributed government medical and DNA databases and demographic and neighborhood statistic datasets. Another case is when it is not possible to provide

exact answers because only summarized data is available for reasons of privacy and confidentiality, e.g., patient record, customer profile and pattern. Furthermore, many real-world distributed data systems may limit the number of queries that can be issued and/or charge for excessive data accesses. For example, there may exist per-IP limits (for web interface queries) or per developer key limits (for Application Programming Interface (API)-based queries). Even when the (daily) limit is high, repeated executions incur a high monetary cost (e.g., in cloud deployments) and waste computational and communication resources. The accessed data nodes can either fully execute the queries (to produce exact results) or locally deploy an AQP technique to produce estimates. In the latter case, we must rely upon the cardinality prediction accuracy provided by the applied AQP technique.

1.3. Rationale and Challenges

The above motivation discussion raises the following desiderata: Develop AQP techniques that:

—(D1) are applicable to all data-environment scenarios (restricted-access or not),
—(D2) are inexpensive (i.e., avoid relying on excessive querying of and communication with the data nodes), while
—(D3) offering high prediction accuracy, and
—(D4) being prudent in terms of computer-network-store resource utilization.

Let us consider an indicative baseline solution for AQP in our environment. One approach is to store, e.g., locally to a central node, all the AQP structures (e.g., histograms, samples, and sketches) from each data node. Hence, we can simply locally access this node for cardinality prediction. First, this violates our desideratum D1, as privacy issues emerge. Obviously, retaining all AQP structures provides one with the whole information about the underlying data distribution. For instance, in the case of histograms, we obtain the underlying probability data distribution, while in sampling methods we retain actual samples from the remote datasets. Even in cases where the local accesses to AQP structures were secured (which is again subject to major security concerns), we would have to cope with the problem of AQP structure updates. The maintenance of those structures in the face of updates demands high network bandwidth overhead and latency for communicating with all nodes during updates of the underlying datasets at these nodes, and scalability and performance bottleneck problems arise at the central node. Therefore, this approach does not scale well and can be expensive, violating our D2 and D3 criteria above.

An alternative baseline solution would be to do away with the central node and send the analytics query to all data nodes, each of which maintains traditional AQP statistical structures and sends back its results to the querying node which then aggregates the per-node results. As before, this violates many of our desiderata. It is not applicable to restricted-access scenarios (violating D1) and involves heavy querying of data nodes (violating D2 and D4).

These facts help expose the formidable challenges to the problem at hand. In this work, we study cardinality prediction in distributed systems taking into consideration the above-mentioned desiderata. Although significant data-centric AQP approaches (like multi-dimensional histograms, self-tuning histograms, sampling methods, please refer to Section 2.2) for cardinality prediction have been proposed for centralized and unrestricted systems, a solution for our environment is currently not available.

There are three fundamental pressures at play here. The first pertains to the development of a solution for cardinality prediction that is efficient and scalable, especially for distributed scale-out environments, where the datasets and computation are

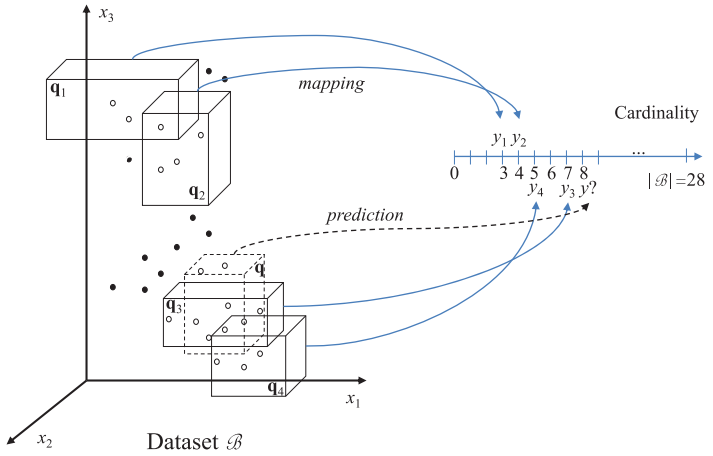


Fig. 2. A dataset $\mathcal{B}$ of 28 3-dimensional points with four 6-dimensional range queries $\mathbf{q}_1, \ldots, \mathbf{q}_4$ represented as 3D rectangles. The data points in white dots correspond to the data points returned by the range queries. Each query $\mathbf{q}_i$ is mapped to a cardinality value y_i. The cardinality prediction for the query $\mathbf{q}$ is achieved by analyzing the similarity and the overlapping of query $\mathbf{q}$ with the four range queries.

distributed. The second pertains to the prediction accuracy, where as we shall see traditional solutions fall short. The third concerns the wide applicability of a proposed method, taking into account environments where data accesses may be restricted.

We propose a solution that addresses these tensions. Conceptually, its fundamental difference from related work is that it is query-driven, based on a novel function estimation model, trained by a small number of queries and later used to predict answers to unseen analytics queries. The challenging aim of our approach is to swiftly provide cardinality prediction of ad-hoc, unseen queries while *avoiding executing them over data nodes*, saving communication and computational resources and money.

Example: Let us provide an example on the cardinality prediction problem over a specific dataset $\mathcal{B}$ illustrated in Figure 2. Liaise with the 3-dimensional data space in Figure 1 (left), where there are four 6-dimensional range queries $\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3$, and $\mathbf{q}_4$ represented as 3-dimensional rectangles over the data space. The data set $\mathcal{B}$ contains 28 data points (black and white dots) and the points that are contained in every query are illustrated with white dots. Each query $\mathbf{q}_i$ is associated with the cardinality y_i of the corresponding subset, which contains the white dots. Figure 2 shows the cardinalities $y_1, \ldots, y_4$ of each query, such that $y_i \in \{0, |\mathcal{B}|$. The overall idea for predicting the cardinality y of an *unseen* query $\mathbf{q}$ defined over the dataset $\mathcal{B}$ is to make use of the actual mappings (query, cardinality), thus extrapolating this knowledge to provide an estimate $\hat{y}$ without executing the query $\mathbf{q}$, i.e., without accessing and scanning the data points in $\mathcal{B}$. The exploitation of these mappings is achieved by locating the most similar queries $\mathbf{q}_i$ with the unseen one $\mathbf{q}$ and then, based on this similarity, to reason about an estimate $\hat{y}$ of the actual cardinality y. In our example, the unseen query $\mathbf{q}$, which is illustrated as a dotted 3-dimensional rectangle, overlaps with the two 3-dimensional rectangles that correspond to the queries $\mathbf{q}_3$ and $\mathbf{q}_4$. Based on this overlapping, we can infer that the actual cardinality y of the query $\mathbf{q}$ can be estimated based on the known cardinalities y_3 and y_4 to a certain extent driven by the similarity and overlapping of $\mathbf{q}$ with the queries $\mathbf{q}_3$ and $\mathbf{q}_4$, respectively. In this work, we provide specific statistical learning algorithms for extracting knowledge from the mapping $\mathbf{q} \rightarrow y$ and how to use the similarity and overlapping context between queries to proceed with estimating the cardinality of unseen queries. Similar reasoning holds for radius queries.

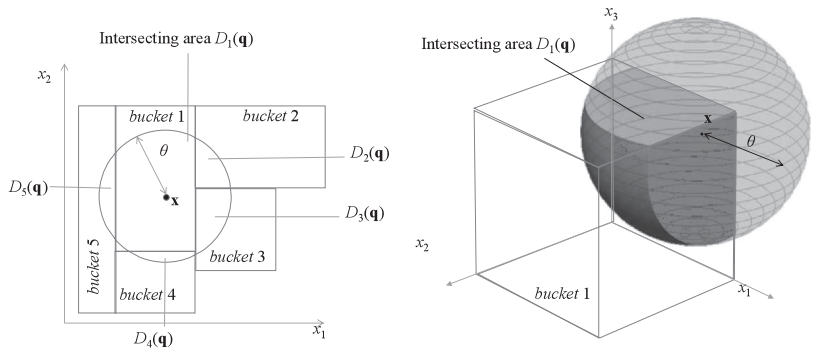


Fig. 3. (Left) Five intersecting areas $\mathbb{D}_b(\mathbf{q})$ of 2-dimensional bucket b with radius query $\mathbf{q} = (\mathbf{x}, \theta)$ under L_2. (Right) Intersecting area of 3-dimensional bucket and radius query under L_2.

2. RELATED WORK

We report on data-centric and query-driven approaches for cardinality prediction for range and radius queries given a data space of $\mathbb{R}^d$. Although data-centric approaches fail our desiderata, we discuss them here for reasons of comprehensiveness.

2.1. Data-Centric Approaches

Most approaches for cardinality prediction use some form of multi-dimensional histograms. Histograms partition the data space into buckets by inspecting the (possibly huge) underlying set $\mathcal{B}$ and then estimate the probability density function (PDF) $p(\mathbf{x})$. In histograms, the estimation of $p(\mathbf{x})$ is highly exploited especially for cardinality prediction of range queries like in Gunopulos et al. [2005]. In the case of a radius query $\mathbf{q}$, a histogram achieves cardinality prediction by first computing for each bucket b the intersecting area $\mathbb{D}_b(\mathbf{q}) \subset \mathbb{R}^d$ with the query as shown in Figure 3, where the query intersects with (left) five buckets/rectangles, and (right) with one bucket/cube. Assuming that $p(\mathbf{x})$ is uniform in each bucket b (local uniformity assumption), the number of points that satisfy the query $\mathbf{q}$ is estimated by the number of points in bucket b multiplied by the d-dimensional volume fraction $\frac{\mathbb{D}_b(\mathbf{q})}{\mathbb{D}_b}$, where $\mathbb{D}_b \subset \mathbb{R}^d$ is the area of bucket b. The answer set cardinality y is obtained by summing the *local* cardinalities from each intersecting bucket.

However, histograms do not scale well with big datasets. Histograms need be periodically rebuilt to incorporate updates (i.e., $p(\mathbf{x})$ is updated), increasing the overhead of this approach. Further, the local uniformity assumption rarely holds in real datasets. Moreover, cardinality prediction dealing with radius queries costs dearly as it requires the expensive computation of $\mathbb{D}_b(\mathbf{q})$. For instance, the intersection of a radius query under L_2 with a bucket b (rectangle in a 2-dimensional histogram) results in an irregular shape whose $\mathbb{D}_b(\mathbf{q})$ area is difficult to compute; see Figure 3 (left). Central to our thinking is that histograms are constructed solely from data, thus obviously being not applicable to our problem for the above-mentioned reasons, e.g., access restricted data.

Although methods for calculating $\mathbb{D}_b(\mathbf{q})$, like Berchtold et al. [1997] by adopting the Monte-Carlo method, cope with this problem, especially for the radius queries, they (i) require a large number of random points for accurate estimation (especially in high dimensional data), (ii) assume uniformity in each bucket ($p(\mathbf{x})$ is constant within $\mathbb{D}_b(\mathbf{q})$), and (iii) repeat this process for every partially intersecting bucket, leading to significant overhead.

Self-tuning histograms (STHs) [Srivastava et al. 2006] were introduced to alleviate some of the problems with histograms. STHs estimate $p(\mathbf{x})$ from scratch, starting with

no buckets and relying only on the cardinality provided by the execution of queries, referred to as Query Feedback Records (QFR): Given a range query $\mathbf{q}$ with cardinality y, STHs learn the conditional probability density $p(\mathbf{x}|y, \mathbf{q})$. Unfortunately, guarantees on cardinality prediction accuracy are hard to obtain and most methods for STHs lack relevant theoretical analyses. Fundamentally, the limitations in STHs in our problem stem from the fact that they estimate $p(\mathbf{x}|y, \mathbf{q})$, thus having to access data (in multi-dimensional STHs at least one scan of the set $\mathcal{B}$ is required), deals with the data distribution and make assumptions of statistical dependencies in data.

Other histogram-based cardinality prediction methods adopt wavelets [Chakrabarti et al. 2000] and entropy [To et al. 2013]. Briefly, the idea is to apply wavelet decomposition to the dataset to obtain a compact data synopsis based on the wavelet coefficients. By nature, wavelets-based AQP relies on the synopsis construction over data, thus could not be applied to our problem. Overall, STHs and the other advanced histogram-based approaches are associated with data access for estimating $p(\mathbf{x})$ or any other $p(\mathbf{x}|\mathbf{q}, \ldots)$, thus not applicable in our problem.

There are also methods for predicting the cardinality over data subspaces defined by radius queries under any L_p. In Tao et al. [2003], the methods use the $\mathcal{B}$ set's fractal dimensions. They rely on certain assumptions on the density of data points in $\mathcal{B}$ and require data access to construct their structures. Sampling methods, e.g., Olken and Rotem [1990], have been also proposed. They share the common idea to evaluate the query over a small subset of $\mathcal{B}$ and extrapolate the observed cardinality. Finally, another approach for AQP answering to cardinality estimation is data sketching; we refer the reader to Cormode et al. [2012] for a survey of sketching techniques. Sketching algorithms construct estimators from the raw data and yield a function of these estimators as the answer to the query.

2.2. Query-Driven Approaches

The idea of predictive analytics based only on the knowledge from previously issued queries and their results are only now emerging, based on our previous work on query-driven learning for centralized cardinality prediction of range queries in Anagnostopoulos and Triantafillou [2015b] and for radius queries in Anagnostopoulos and Triantafillou [2015a]. The model proposed in this article is fundamentally different from previous approaches in the following ways: (i) We deal with both types of (range and radius) queries. This is achieved by introducing the idea of unsupervised function estimation/regression for cardinality, which is based on a probabilistic framework over query distances, thus being independent on the types of queries. (ii) Based on that framework, the proposed model can *detect changes* in query patterns (distributions), which is a fundamental requirement for query-driven approaches. (iii) Then, the model is capable of being incrementally updated, by adjusting its structures to reflect the *changes in query pattern distributions*. (iv) Moreover, the model is decentralized and can be applied in modern scale-out data systems over distributed data stores. This also provides for elasticity (adding/removing data nodes) in distributed environments since the model training phase (as will be shown) is purely decentralized (local) and thus independent among data nodes. (v) Finally, our function estimation model for cardinality prediction is significantly different than the models in Anagnostopoulos and Triantafillou [2015a, 2015b].

The fundamental idea of previous methods [Anagnostopoulos and Triantafillou 2015a, 2015b] and our approach is to partition (cluster) the query patterns, identify query prototypes, and then associate with each query prototype a cardinality prototype. Learning and adapting the mapping between the query space and cardinality domain is most significant in cluster-based prediction [Kohonen 1989].

Both previous approaches [Anagnostopoulos and Triantafillou 2015a, 2015b] focus only on utilizing the prediction error[1] as feedback to adapt the cardinality prototypes only – degrading query space partitioning to a mere clustering. Instead, we further propagate this feedback to inform/*supervise* query clustering as well (Section 4.1). This leads to *a more powerful function estimation model, which performs joint minimization* (in the query space and cardinality domain) of the prediction error. In addition and more interestingly, the change detection of the query space, reflecting the way analysts change their interests in exploring and analyze data subspaces, is provided in our model and treated as an optimal stopping time problem. Through this time-optimized stochastic framework, we are able to securely decide when a query subspace is *novel* reflecting the analysts' interest in this query subspace with the ultimate purpose to minimize the risk of low prediction accuracy. In this context, we also propose an adaptation mechanism to continuously follow the trend of the analytics queries issued by the analysts to novelty query subspaces.

3. FUNDAMENTALS AND PROBLEM FORMULATION

Consider a user-defined data subspace $\mathbb{D}$, which is either a range data subspace $\mathbb{D}(\mathbf{a}, \mathbf{b})$ or a radius data subspace $\mathbb{D}(\mathbf{x}, \theta)$, from range $\mathbf{q} = [\mathbf{a}, \mathbf{b}]$ or radius $\mathbf{q} = [\mathbf{x}, \theta]$ query, respectively, over a dataset $\mathcal{B}$. Let also $y \in \mathbb{N}$ denote the *actual* cardinality for the data subspace $\mathbb{D}$.

We seek to estimate the unknown function $f : \mathbb{D} \to \{0, \ldots, |\mathcal{B}|\} \subset \mathbb{N}$ for each query type, i.e., the mapping $\mathbf{q} \mapsto y$ that predicts the cardinality $\hat{y}$ of the data subspace $\mathbb{D}$ of a query $\mathbf{q}$. We notate $y = f(\mathbf{a}, \mathbf{b})$ and $y = f(\mathbf{x}, \theta)$ for the range and radius queries, respectively, or to simplify the notation, $y = f(\mathbf{q})$.

The prediction error for estimating $\hat{y}$ is defined as the absolute-deviation loss $|y - \hat{y}|$. Note that other prediction errors can be also adopted, e.g., the λ-insensitive loss $\max\{|y-\hat{y}|-\lambda, 0\}, \lambda > 0$ or the 0–1 loss $I(y \neq \hat{y}) \in \{0, 1\}$ with I be the indicator function. We focus on the absolute-deviation loss, because (i) it is widely used for evaluating the cardinality prediction error as in Gunopulos et al. [2005] and Srivastava et al. [2006] and (ii) it provides us the (robust statistic) median of the data subspaces cardinalities, which is used to accurately approximate the actual cardinalities, as elaborated later.

In statistical learning theory, the function f estimation is achieved by a regression function $\hat{f}(\mathbf{q}; \alpha)$ with parameter $\alpha \in \mathcal{A}$, where $\mathcal{A}$ is a parameter space defined later. In this context, the problem is reduced to choosing from a set of regression functions $\mathcal{F} = \{\hat{f}(\mathbf{q}; \alpha) | \alpha \in \mathcal{A}\}$ the regression function $\hat{f}(\mathbf{q}; \alpha^*)$, which minimizes the Expected Prediction Error (EPE) with respect to the absolute-deviation loss $|y - \hat{y}|$, i.e.,

$$\hat{f}(\mathbf{q}; \alpha^*) = \arg\min_{\hat{f} \in \mathcal{F}} \mathbb{E}[|y - \hat{f}(\mathbf{q}; \alpha)|]. \quad (1)$$

Consider now K distributed data nodes $S_1, \ldots, S_K$, each one storing a dataset $\mathcal{B}_k$, $k \in [K]$, consisting of d-dimensional vectors $\mathbf{x} = [x_1, \ldots, x_d] \in \mathbb{R}^d$. We again stress that we wish to reduce base data access during query processing the datasets $\mathcal{B}_k, \forall k$. We only have access to query-cardinality responses, that is, the cardinality y_k of the data subspace defined by a random query $\mathbf{q}$ executed on node S_k. Assume also a Central Node (CN), which can communicate with all data nodes. The aim of the CN is, given an incoming query $\mathbf{q}$, to predict the cardinality $\hat{y}_k$ for each node S_k, without accessing the nodes.

In the remainder, when we refer to a query $\mathbf{q}$ we imply both a range and a radius query, unless otherwise stated. Let us focus on the set $\mathcal{T} = \{(\mathbf{q}_i, y_i)\}_{i=1}^n$ of *training pairs*

[1]Kernel regression error in Anagnostopoulos and Triantafillou [2015b], linear regression error in Anagnostopoulos and Triantafillou [2015a].

of (query, cardinality) corresponding to a node S_k. The CN receives a new (unseen) analytics query $\mathbf{q}$ and predicts the cardinality $\hat{y}$ of the corresponding data subspace using only the set $\mathcal{T}$ without actually executing $\mathbf{q}$ at S_k. In this context, we discuss some ideas on how to exploit the set $\mathcal{T}$ to estimate the regression function $\hat{f}$, which minimizes the EPE as defined in (1).

Idea 1: An idea is for the CN to keep all pairs $(\mathbf{q}_i, y_i)$ and, given an unseen query $\mathbf{q}$, it finds the most *similar* query $\mathbf{q}_j$ under L_2. In this case, CN predicts the corresponding cardinality of the data subspace as

$$\hat{y} = \hat{f}(\mathbf{q}) = y_j : \mathbf{q}_j = \arg\min_{i \in [n]} \|\mathbf{q} - \mathbf{q}_i\|_2, (\mathbf{q}_j, y_j) \in \mathcal{T}. \tag{2}$$

The CN can also involve the κ closest (most similar) queries to $\mathbf{q}$ under L_2 and average their cardinalities, i.e., perform a κ-nearest neighbors regression, i.e.,

$$\hat{y} = \hat{f}(\mathbf{q}; \kappa) = \frac{1}{\kappa} \sum_{\mathbf{q}_j \in \mathcal{N}(\mathbf{q}; \kappa)} y_j : (\mathbf{q}_j, y_j) \in \mathcal{T}, \tag{3}$$

where $\mathcal{N}(\mathbf{q}; \kappa)$ is the L_2 neighborhood of $\mathbf{q}$ defined by the κ closest queries $\mathbf{q}_j$ in $\mathcal{T}$.

The major problems here are: (i) we must store and search all previous pairs for each unseen query $\mathbf{q}$; $\mathcal{T}$ can be huge and we have to search the $\mathcal{T}$ sets for all K data nodes. Deciding which pairs to discard is not a trivial task (a new pair might convey useful information, while another new one might be a redundant/repeated query); (ii) with data updates at node S_k, which impact the cardinality of query results, it is not trivial to determine which pairs from $\mathcal{T}$ and how many of them to update. Specifically, in the κ-nearest neighbors regression: (1) the CN can only add actual pairs of $(\mathbf{q}, y)$ to support regression, which obviously does not scale with the number of stored pairs in $\mathcal{T}$; (2) since there is no other information stored for pinpointing the impact of the regression accuracy of any arbitrary neighborhood of the closest κ queries, then any update policy for discarding or updating some of the κ closest actual pairs $(\mathbf{q}, y) \in \mathcal{T}$ cannot guarantee that this will minimize the prediction error for *future* queries. This is the major disadvantage of the non-parametric regression predictive modeling, where there is no support for scaling with the number of the training pairs $\mathcal{T}$ and no parametric information is coded to represent the very specific neighborhood of an input query $\mathbf{q}$ for updating any internal structure in light of minimizing the prediction error. (iii) Even worse, all pairs may need updating in order to decrease the current prediction error, which evidently does not scale, due to the fact that when query patterns change (new analysts' interests), there may be many pairs in $\mathcal{T}$ that will not contribute to the prediction result (the new queries are actually distant to the previous ones) or even negatively impact the final result. To avoid such problems, we should extract knowledge from each, per node, $\mathcal{T}$ as to *how query and data subspace cardinality depend on each other*.

Idea 2: Moving forward, focusing again on the set $\mathcal{T}$ of a data node, we could cluster similar queries under L_2, thus, forming a much smaller set $\mathcal{W}$ of representative (prototype) queries $\mathbf{w}$ with $|\mathcal{W}| \ll |\mathcal{T}|$. For instance, $\mathbf{w} \in \mathcal{W}$ can be the centroid of those queries from $\mathcal{T}_\mathbf{w} \subset \mathcal{T}$ with L_2 distances from $\mathbf{w}$ being the smallest among all other candidate centroids. This may ameliorate the above problems. However, *we are not just interested in clustering* $\mathcal{T}$. We cluster $\mathcal{T}$, i.e., performing unsupervised learning, with the aim at cardinality prediction, i.e., performing *regression* (supervised learning) of cardinality against query prototypes. We, thus, pursue a clustering and regression model, which exploits (i) the statistical structure emerging from query clustering by generating query prototypes and (ii) the prediction error to guide both the regression and clustering processes. This leads us on the idea of *unsupervised regression*, which

interprets that we associate each $\mathbf{w}_i \in \mathcal{W}$ with a "representative" cardinality of the corresponding data subspace defined by $\mathbf{w}_i$. This representative cardinality should refer to a statistic, e.g., the average cardinality of those queries that belong to $\mathcal{T}_{\mathbf{w}_i}$, or, to a robust statistic like the median cardinality of the queries belonging to $\mathcal{T}_{\mathbf{w}_i}$.

The mapping $\hat{f}$ from representative queries to representative cardinalities of the underlying data subspaces defined by analysts' queries approximates the $f(\mathbf{q})$ regression function:

$$f(\mathbf{q}) \approx \hat{f}(\mathbf{w}), \mathbf{q} \in \mathcal{T}_{\mathbf{w}}, \mathbf{w} \in \mathcal{W}. \tag{4}$$

Once this approximation is achieved, the CN only keeps $\mathcal{W}$ and discards $\mathcal{T}$. Nonetheless, we want to move beyond "off-line" processing of the $\mathcal{W}$ sets of nodes for each query, as this would require high memory resources. Hence, we need *incremental* clustering of the query space and *incremental* regression each time a new query arrives at the CN. In other words, we require an adaptive, joint *clustering and regression algorithm* that incrementally quantizes $\mathcal{T}$ by minimizing the EPE.

3.1. Problem Formulation

The adoption of any known algorithm for on-line clustering (e.g., on-line κ-means), is not directly applicable, as we explicitly focus on the quantization of the query space for the purpose of regression and prediction. Also, on-line regression methods, such as incremental regression trees [Ikonomovska et al. 2010] and on-line support vector regression [Ma et al. 2003], could not solve the problem at hand: We need to deal with the fact that queries are continuously observed, conveying analysts' (changing) interests during data exploration and predictive analytics tasks. Query distributions in many application domains are known to be non-uniform – actually, considerably skewed – with specific portions of the data subspaces being more popular, e.g., an instance of the *hubness* phenomenon [Radovanovic et al. 2010]. Furthermore, query patterns change with time, reflecting changes of analysts' interests to exploring different data subspaces. A model's capability to detect and adapt to such changes requires explicit information on very specific subspaces of the query space. This is neither easily provided nor supported by incremental quantization and regression methods. Moreover, the problem here is not only to detect changes on the query spaces, but also determine which query representative(s) of the query subspaces and which cardinality representatives to update and how, upon query updates.

The fundamental problems are given in the following.

PROBLEM 1. *Given a data node S_k that locally stores a dataset $\mathcal{B}_k$, incrementally estimate $f(\mathbf{q})$ by a function estimation model $\hat{f}_k(\mathbf{q};\alpha_k)$ with parameter α_k, which minimizes the EPE in (1).*

Moreover, the function estimation model $\hat{f}_k$ should be up-to-date as query patterns are subject to change. This is because of the fact the analysts change their interest in or generate new interests in other unexplored data subspaces. In this context, the estimation model should first detect changes in query patterns and then proceed with adaptation keeping itself up-to-date.

PROBLEM 2. *Given a function estimation model $\hat{f}_k$ for a data node S_k and changes in the query patterns over a dataset $\mathcal{B}_k$, incrementally detect such changes and adapt model's parameter α_k such that the updated model minimizes the EPE in (1).*

3.2. Our Approach and Contribution

We address Problems 1 and 2 bearing in mind the requirements of (i) incremental quantization of the query space, corresponding to analyst-defined data subspaces of

interest, (ii) incremental regression of cardinality on query prototypes, and (iii) on-line detection of *novel* data subspaces of interest accompanied with incremental adaptation of the *current* function estimation model. To this end, we combine adaptive vector quantization of the query space by taking into account the objective to minimize the EPE. We propose an incremental *unsupervised regression vector quantization* model to approximate f. Fundamentally, we rest on the fact that similar queries under L_2 correspond to close medians of the cardinalities of the corresponding data subspaces, as it will be proved later. Based on that, the query prototypes are quantized with respect to the EPE in (1). Then, we interpolate $\hat{f}$ over a weighted average of the cardinality medians of those data subspaces that overlap with the data subspace defined by an unseen query. This median-based interpolation is a robust measure of the central tendency of the cardinalities that fall within those overlapping data subspaces.

Once we have approximated f by training a $\hat{f}_k$ on each data node S_k, we predict the cardinality y_k over a random analyst-defined data subspace for each S_k without executing the query over S_k. Our model also swiftly detects and adapts to changes in query patterns. The major technical contributions are given as follows:

—An unsupervised regression function estimation model of the cardinality of a data subspace defined by radius and range queries, with its convergence analysis;
—A stochastic time-optimized method for detecting query patterns changes based on Optimal Stopping Theory and Adaptive Resonance Theory;
—Comprehensive experimental results analyzing the performance of our model and showcasing its benefits vis-à-vis data-centric sampling [Vitter 1985], histograms [Gunopulos et al. 2005] and STHs [Srivastava et al. 2006; Viswanathan et al. 2011].

4. FUNCTION ESTIMATION MODEL

As we are after decentralized function estimation, each approximation model $\hat{f}_k$ corresponds to a data node S_k. Due to the assumed undesirable access to $\mathcal{B}_k$, the regression function $\hat{f}_k$ can only approximate the unknown function f based only the available training pairs $(\mathbf{q}, y)$ for data node S_k. As highlighted above, we perform two major training tasks: The first task (T1) is adaptive vector quantization of the query space $\mathbb{Q}$ by incrementally updating a set of query prototypes. The second task (T2) is an adaptive (probabilistic) regression of associated cardinality prototypes on query prototypes.

4.1. Unsupervised Regression Function Estimation

4.1.1. Query Prototypes. A key task for T1 is to represent random queries $\mathbf{q} \in \mathbb{Q}$ by a set of M prototypes $\mathcal{W} = \{\mathbf{w}_j \in \mathbb{Q}, j \in [M]\}$. A query $\mathbf{q}$ is represented by its winner (closest under L_2) prototype:

$$\mathbf{w}^* = \arg\min_{j \in [M]} \|\mathbf{q} - \mathbf{w}_j\|_2. \tag{5}$$

Range query prototype: In the case of a range query $\mathbf{q} = [\mathbf{a}, \mathbf{b}]$, its representative query refers to the prototype $\mathbf{w}^* = [\mathbf{a}^*, \mathbf{b}^*] \in \mathcal{W}$, where the sum of the L_2 distances of the lower bound vectors $\|\mathbf{a} - \mathbf{a}^*\|_2$ and the upper bound vectors $\|\mathbf{b} - \mathbf{b}^*\|_2$ is the smallest compared to the sum of the corresponding L_2 distances from all $\mathbf{w} \in \mathcal{W}$. The similarity of a range prototype $\mathbf{w}^*$ and a range query $\mathbf{q}$ refers to the similarity of their corresponding hyper-rectangles, i.e., the closeness of their lower and upper bound vectors under L_2. Statistically, the range data subspace $\mathbb{D}(\mathbf{a}^*, \mathbf{b}^*)$ defined by the query prototype $\mathbf{w}^*$ is a *representative* range data subspace of all analyst-defined data subspaces $\mathbb{D}(\mathbf{a}, \mathbf{b})$ defined by past issued range queries over the dataset $\mathcal{B}$. The physical interpretation of the representative query, as it will be proved later in Theorem 4.6, is that the lower bound vector $\mathbf{a}^*$ and upper bound vector $\mathbf{b}^*$ of the prototype $\mathbf{w}^*$ refer to the mean vectors of *all* lower bound vectors $\mathbf{a}$ and upper bound vectors $\mathbf{b}$ of those

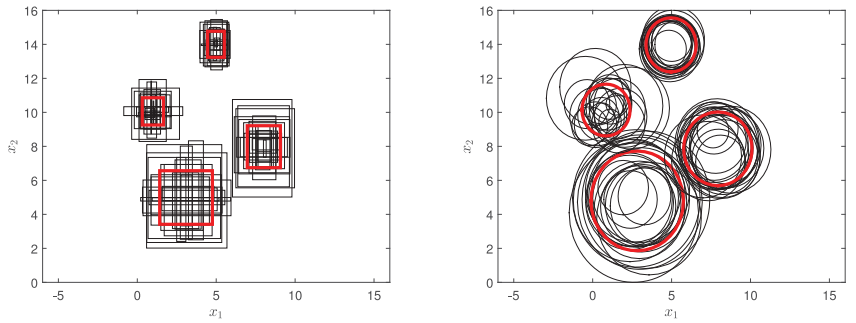


Fig. 4. (Left) Example of four prototype range queries $\mathbf{w}$ (red rectangles) projected onto a 2-dimensional data space (x_1, x_2) representing 100 range queries issued over four data subspaces of interest. (Right) Example of four prototype radius queries $\mathbf{w}$ (red circles) projected onto a 2-dimensional data space (x_1, x_2) representing 100 radius queries issued over four data subspaces of interest.

issued range queries $\mathbf{q} = [\mathbf{a}, \mathbf{b}]$, which were closest to $\mathbf{q}^*$. Figure 4 (left) shows the four prototype range queries $\mathbf{w}$ as *average* red rectangles projected onto a 2-dimensional data space (x_1, x_2) representing 100 range queries issued over four data subspaces of interest.

Radius query prototype: In the case of a radius query $\mathbf{q} = [\mathbf{x}, \theta]$, two radius queries are similar when their corresponding centers and their radii are similar, too. The representative radius query $\mathbf{w}^* = [\mathbf{x}^*, \theta^*] \in \mathcal{W}$ is the closest to a radius query $\mathbf{q} = [\mathbf{x}, \theta]$, when the sum of the distances of their centers and radii $\|\mathbf{x} - \mathbf{x}^*\|_2^2 + (\theta - \theta^*)^2$ is the smallest among all corresponding distances from all $\mathbf{w} \in \mathcal{W}$. The prototype radius data subspace $\mathbb{D}(\mathbf{x}^*, \theta^*)$ is represented as the *average* hyper-sphere from all past analyst-defined radius data subspaces $\mathbb{D}(\mathbf{x}, \theta)$, where center $\mathbf{x}^*$ and θ^* are the mean vector of all centers and the average radius of all radii, respectively, of past issued radius queries with closest prototype $\mathbf{w}^*$; see Theorem 4.7. Figure 4 (right) shows the four prototype radius queries $\mathbf{w}$ as *average* red circles projected onto a 2-dimensional data space (x_1, x_2) representing 100 radius queries issued over four data subspaces of interest.

The Expected Quantization Error (EQE) under L_2^2 incurred by this representation, i.e., by projecting a random query $\mathbf{q}$ to its closest prototype $\mathbf{w} \in \mathcal{W}$, is given by

$$\mathcal{E}_1(\mathbf{w}_1, \ldots, \mathbf{w}_M) = \mathbb{E}\left[\min_{j \in [M]} \|\mathbf{q} - \mathbf{w}_j\|_2^2\right]. \tag{6}$$

The task T1 incrementally minimizes (6) with the presence of random queries one at a time by updating their winner prototypes.

The task T2 incrementally estimates the unknown conditional probability density of cardinality y given a query $\mathbf{q}$, $p(y|\mathbf{q})$, with the presence of a random pair $(\mathbf{q}, y)$ provided once by node S_k. While task T1 progressively optimizes the EQE $\mathcal{E}_1$, at the same time tasks T2 trains a probabilistic regression model, which importantly affects/feedbacks also the quantization of $\mathbb{Q}$ as will be discussed later.

To introduce the idea of quantization of $\mathbb{Q}$ driven by the current EPE, thus obtaining the insights of the mapping $\mathbf{q} \rightarrow y$ association, we rest on the assignment conditional probability density of $\mathbf{w}_j$ given a random query $\mathbf{q}$, $p(\mathbf{w}_j|\mathbf{q})$. This is the probability of assigning a random query $\mathbf{q}$ to a prototype $\mathbf{w}_j \in \mathcal{W}$. We require this probability to be dependent only on the L_2 distance between a prototype $\mathbf{w}_j$ and a random query $\mathbf{q}$, thus applicable to both types of radius and range queries. Define also (i) $p(\mathbf{w}_j)$ as the prior probability that $\mathbf{w}_j$ is assigned to a query and (ii) $p(\mathbf{q}|\mathbf{w}_j)$ as the probability density of *generating* a query $\mathbf{q}$ in a query subspace of $\mathbb{Q}$ around the prototype $\mathbf{w}_j$. In that case,

the probability density $p(\mathbf{q})$ is expressed by a mixture model over all prototypes of $\mathcal{W}$:

$$p(\mathbf{q}) = \sum_{j \in [M]} p(\mathbf{q}|\mathbf{w}_j) p(\mathbf{w}_j). \tag{7}$$

4.1.2. Cardinality Prototypes. The key task T2 estimates the cardinality prototypes in $\mathcal{U} = \{u_j \in \mathbb{R}, j \in [M]\}$, where each one is associated with a query prototype $\mathbf{w}_j$, $j \in [M]$. The cardinality prototypes u_j of the (range/radius) data subspaces of the query prototypes $\mathbf{w}_j$ comprise the fundamental model parameters used for minimizing the EPE in (1) in terms of function estimation and for regression in terms of prediction.

Consider the case of a range query. Given a range query $\mathbf{q} = [\mathbf{a}, \mathbf{b}]$, let $y = |\mathbb{D}(\mathbf{a}, \mathbf{b})|$ be the cardinality of the corresponding range data subspace $\mathbb{D}(\mathbf{a}, \mathbf{b})$, and $\mathbf{w}^* = [\mathbf{a}^*, \mathbf{b}^*] \in \mathcal{W}$ be its closest prototype given (5). As it will be proved in Theorem 4.7, the cardinality prototype u^*, which corresponds to the cardinality of the *average* range data subspace $\mathbb{D}(\mathbf{a}^*, \mathbf{b}^*)$, refers to the median of all cardinalities y of all those past range queries with closest prototype $\mathbf{w}^*$. The same holds true for the cardinality prototypes associated with radius queries, i.e., for range queries:

$$u^* = |\mathbb{D}(\mathbf{a}^*, \mathbf{b}^*)| = \text{median}\{y\} : \mathbf{w}^* = \arg\min_{j \in [M]} \|\mathbf{q} - \mathbf{w}_j\|_2 \text{ and } y = |\mathbb{D}(\mathbf{a}, \mathbf{b})|$$

and for radius queries:

$$u^* = |\mathbb{D}(\mathbf{x}^*, \theta^*)| = \text{median}\{y\} : \mathbf{w}^* = \arg\min_{j \in [M]} \|\mathbf{q} - \mathbf{w}_j\|_2 \text{ and } y = |\mathbb{D}(\mathbf{x}, \theta)|.$$

The cardinality prototype u^*, as a median, refers to a robust statistic over the cardinalities of those data subspaces from queries similar to $\mathbf{w}^*$. The cardinality prototype u^* is the most resistant statistic, having a breakdown point of 50%, i.e., robust in the presence of outlier values, is independent of any distance metric, and provides us a measure of location when the distribution of the cardinality values of $|\mathbb{D}(\mathbf{a}, \mathbf{b})|$ (range query) or $|\mathbb{D}(\mathbf{x}, \theta)|$ (radius query) is skewed. These cardinality prototypes are used for *instantaneous predictions* that are given as feedback to the entire training process for minimizing the EPE, as discussed later.

The major problem for the task T2 is to approximate the conditional density $p(y|\mathbf{q})$ through the prototype pairs $(\mathbf{w}_j, u_j)$. Specifically, we use those cardinality prototypes in $\mathcal{U}$ by interpolation to predict the cardinality of an analyst-defined data subspace given an unseen query $\mathbf{q}$. Based on the approximation of $p(y|\mathbf{q})$ through the prototype pairs $(\mathbf{w}_j, u_j)$, task T2 progressively uses the current, instantaneous prediction error derived from the cardinality prototypes u_j as feedback in quantizing $\mathbb{Q}$. During training of the $\hat{f}_k$, given an unseen query $\mathbf{q}$, we first find its winner (closest) query prototype $\mathbf{w}_j$. This winner prototype, representing a subspace of $\mathbb{Q}$, is associated with a cardinality prototype u_j. Then, the instantaneous predicted cardinality $\hat{y}$ for query $\mathbf{q}$ is obtained by the cardinality prototype, i.e., $\hat{y} = u_j$, and the induced current prediction error is $|y - u_j|$.

To overall minimize the EPE in (1), we require T2 to minimize the expected probabilistic cost based on cardinality prototypes u_j weighted by the probability assignment $p(\mathbf{w}_j|\mathbf{q})$ of their query prototypes. That is, task T2, given the set of query prototypes $\mathcal{W}$, attempts to incrementally minimize the Expected Conditional Cost (ECC):

$$\mathcal{E}_2(u_1, \ldots, u_M|\mathcal{W}) = \mathbb{E}\left[\sum_{j \in [M]} p(\mathbf{w}_j|\mathbf{q})|y - u_j|\right]. \tag{8}$$

Remark 4.1. The function $\mathcal{E}_2$ is a cost function by definition of $p(\mathbf{w}_j|\mathbf{q})$ and is suitable for unsupervised regression. That is because, when $p(\mathbf{w}_j|\mathbf{q}) = 1$ for the nearest

prototype given a fixed set of prototypes $\mathcal{W}$, the assignment probabilities for other prototypes are zero[2]. In that case, the ECC $\mathcal{E}_2$ in (8) reduces to the EPE in (1), which is our main purpose to minimize it, thus having approximated function f by a regression function $\hat{f}_k$.

Remark 4.2. The objective optimization function in (8) is based on a *mixture of experts* where the output u_j is associated with the quantization of the input vector space, which is achieved by the query prototypes $\mathbf{w}_j$, i.e., the j-th expert is represented by the pair $(\mathbf{w}_j, u_j)$. The proposed cost function attempts to minimize the weighted prediction error by estimating the optimal position of the cardinality prototypes u_j in the output space associated with the optimal position of the query prototypes $\mathbf{w}_j$ in the input space. The cardinality prototypes are real numbers in $[0, |\mathcal{B}|]$ that will be closed to optimal positions for minimizing the prediction error with the actual cardinality y. Note also that the actual cardinality y refers to a non-negative integer, by dentition of the cardinality, $y \in \{0, 1, 2, \ldots, |\mathcal{B}|\}$. Our idea is to combine both supervised learning for estimating the best positions of the cardinality prototypes u_j and unsupervised learning for estimating the best positions of the query prototypes $\mathbf{w}_j$. The concept of supervision is based on providing feedback to both classes of prototypes to move around the input–output space. This derives from the mixtures of experts' methodology used for (i) prediction, i.e., regress y on the quantized input vectorial space by $\mathbf{w}_j$, and (ii) for query space quantization (unsupervised learning). As it will be shown in Theorem 4.4, the movement of the query prototypes $\mathbf{w}_j$ is based on both the query input $\mathbf{q}$, which attempts to quantize the input space in an unsupervised manner, and the output y predicted by the network of the experts u_j, which attempts to minimize the absolute difference $|y-u_j|$. Both methodologies are combined to provide the concept of unsupervised regression analyzed in Section 4.2.

Evidently, we do not have any prior knowledge about $p(\mathbf{w}_j|\mathbf{q})$ in order to proceed with the incremental minimization of (8), which corresponds to the minimization of the EPE. Hence, we apply the *principle of maximum entropy*: Among all possible probability distributions that yield a given expected probabilistic cost, we choose the one that maximizes the entropy [Rogers and McClelland 2014]. A key factor to ensure convergence of the T2 minimization of (8) (and thus of the EPE in (1)) and obtain the global minimum is that density $p(\mathbf{w}_j|\mathbf{q})$ conforms to the Gibbs distribution and expressed via the query prototypes in $\mathcal{W}$, i.e.,

$$p(\mathbf{q}|\mathbf{w}_j) \propto \exp\big(-\beta\|\mathbf{q}-\mathbf{w}_j\|_2^2\big), \tag{9}$$

where the parameter $\beta \geq 0$ will be explained later. If we assume that each prototype $\mathbf{w}_j$ has the same prior $p(\mathbf{w}_j) = \frac{1}{M}$, then through Bayes' rule $p(\mathbf{w}_j|\mathbf{q}) = \frac{p(\mathbf{q}|\mathbf{w}_j)p(\mathbf{w}_j)}{p(\mathbf{q})}$, we obtain that

$$p(\mathbf{w}_j|\mathbf{q}) = \frac{\exp(-\beta\|\mathbf{q}-\mathbf{w}_j\|_2^2)}{\sum_{i=1}^{M}\exp(-\beta\|\mathbf{q}-\mathbf{w}_i\|_2^2)}. \tag{10}$$

Remark 4.3. Note that $p(\mathbf{w}_j|\mathbf{q})$ explicitly depends on the L_2^2 distance of queries with the query prototypes. Hence, this renders our model applicable to both range and radius query types.

Based on (10), the task T2 converges to the global minimum of $\mathcal{E}_2$ if parameter β is increased sufficiently slower at each update step in the presence of a pattern pair

[2]The reason why $p(\mathbf{w}_j|\mathbf{q})$ is, at the beginning, non-unity has to do with the training procedure, as will be described.

[Rogers and McClelland 2014]. By varying the parameter β, the probability assignment $p(\mathbf{w}_j|\mathbf{q})$ can be completely *fuzzy* ($\beta = 0$, each random query belongs equally to all prototypes) and *crisp* ($\beta \to \infty$, each random query belongs to only one prototype, or more precisely uniformly distributed over the set of equidistant closest prototypes). As $\beta \to \infty$, this probability becomes a delta function around the winner, i.e., the prototype closer to $\mathbf{q}$. In this case, the expected cost function in (8) is strictly decreasing. As $\beta < \infty$, at each step all prototypes are simultaneously updated towards the pair $(\mathbf{q}, y)$, but for each of them the step size is scaled by the assignment probability in (10). That is, the j-th query prototype and its corresponding cardinality prototype are shifted a fraction of $p(\mathbf{w}_j|\mathbf{q})$ at each step.

4.2. Unsupervised Regression Optimization

4.2.1. Update Rules. Our problem is to approximate the *optimal* query prototypes $\mathcal{W}$ and cardinality prototypes $\mathcal{U}$ that minimize the EQE in (6) and the ECC in (8), respectively. Once those prototypes have approximated for a data node S_k, $\hat{f}_k(\mathbf{q}; \alpha_k)$ with parameter $\alpha_k = \mathcal{W} \cup \mathcal{U}$ approximates $f(\mathbf{q})$ with respect to the EPE . In our case, we deal with (simultaneous) incremental estimation of tasks T1 and T2 upon the presence of a random pair $(\mathbf{q}, y)$ at node S_k. These tasks jointly minimize the objective optimization function:

$$\mathcal{E}_0(\mathcal{W}, \mathcal{U}) = \mathcal{E}_1(\mathcal{W}) + \mathcal{E}_2(\mathcal{U}|\mathcal{W}), \tag{11}$$

with $\mathcal{W} = \{\mathbf{w}_j\}_{j=1}^M, \mathcal{U} = \{u_j\}_{j=1}^M$. The query space $\mathbb{Q}$ quantization is driven by the instantaneous prediction error generated by the current prototypes in $\mathcal{U}$.

In the parameter $\alpha_k = \{\mathbf{w}_j\}_{j=1}^M \cup \{u_j\}_{j=1}^M$, all query and cardinality prototypes represent the associations $\mathbf{w}_j \to u_j$. These prototypes are incrementally updated through Stochastic Gradient Descent (SGD). The rationale of the adoption of SGD is that it approaches the optimal α_k parameter by following the negative gradient of the objective optimization function $\mathcal{E}_0$ after seeing only a single pair $(\mathbf{q}, y)$. SGD computes the gradient $\nabla\mathcal{E}_0(\alpha_k)$ with respect to α_k at each step, thus updating α_k by $\Delta\alpha_k \propto -\eta\nabla\mathcal{E}_0(\alpha_k)$. In SGD, the learning rate $\eta \in (0, 1)$ is a relatively small scalar that drives the movement of the α_k parameter toward the optimal solution based on the current pair. Choosing the proper learning rate and schedule, i.e., changing the value of the learning rate as training progresses, is fairly difficult (this primarily depends on the *convexity* and *smoothness* of the objective optimization function). To ensure convergence of the parameter to the optimal solution, scalar $\eta_j(t)$ at each step t must satisfy: $\sum_{t=0}^{\infty} \eta_j(t) = \infty$ and $\sum_{t=0}^{\infty} \eta_j^2(t) < \infty$ [Bousquet and Bottou 2008]. In practice, a standard strategy is to use a small enough learning rate that gives stable convergence through a hyperbolic-based function, i.e., $\eta(t) \propto \frac{1}{t+1}$.

Based on SGD, the approximation of α_k to its optimum w.r.t. (11) is provided by a set of update rules for each component of the α_k parameter, as stated in Theorem 4.4.

THEOREM 4.4. *Given a pair $(\mathbf{q}, y)$, the estimation model's parameter α_k of data node S_k converges if query prototypes $\mathbf{w}_j$ and cardinality prototypes u_j are updated as*

$$\Delta\mathbf{w}_j = -\eta_j \left(\left(|y - u_j| - \sum_{i \in [M]} p(\mathbf{w}_i|\mathbf{q})|y - u_i| \right) \cdot p(\mathbf{w}_j|\mathbf{q}) + Y_j \right) (\mathbf{q} - \mathbf{w}_j), \tag{12}$$

$$\Delta u_j = \eta_j p(\mathbf{w}_j|\mathbf{q}) \mathrm{sgn}(y - u_j), \tag{13}$$

where $\text{sgn}(\cdot)$ *is the signum function,* Y_j *is the winner indicator:* $Y_j = 1$ *if* $\mathbf{w}_j = \arg\min_{i\in[M]} \|\mathbf{q} - \mathbf{w}_i\|_2$*; 0, otherwise, and* $\eta_j \in (0, 1)$ *is a learning rate of the current update step.*

PROOF. See Appendix A. □

Remark 4.5. The update rule of a prototype $\Delta\mathbf{w}_j$ in a component-wise format for a range query prototype and a radius query prototype is: $\Delta\mathbf{w}_j = [\Delta\mathbf{a}_j, \Delta\mathbf{b}_j]$ and $\Delta\mathbf{w}_j = [\Delta\mathbf{x}_j, \Delta\theta_j]$, respectively. Hence, based on (12) and upon the presence of a range query $\mathbf{q} = [\mathbf{a}, \mathbf{b}]$ and of a radius query $\mathbf{q} = [\mathbf{x}, \theta]$, we obtain, respectively,

$$\Delta\mathbf{w}_j \propto [\Delta\mathbf{a}_j, \Delta\mathbf{b}_j] = [\mathbf{a} - \mathbf{a}_j, \mathbf{b} - \mathbf{b}_j],$$
$$\Delta\mathbf{w}_j \propto [\Delta\mathbf{x}_j, \Delta\theta_j] = [\mathbf{x} - \mathbf{x}_j, \theta - \theta_j].$$

4.2.2. Learning Rate. All prototypes are updated at each step $t = 0, 1, \ldots$ upon presence of a pair $(\mathbf{q}, y)$. Let n_j be the counter indicating the number of times the query prototype $\mathbf{w}_j$ is assigned a query $\mathbf{q}$ as a winner. Furthermore, a closer look at $\mathcal{E}_1$ refers to the M-means quantization algorithm over the query space achieved by SGD as analyzed in Bottou and Bengio [1994]. In that case, the learning rate $\eta_j(t)$ at each time step t should follow the hyperbolic schedule of $\eta_j(t) \propto \frac{1}{n_j(t)}$, which is proved to converge [MacQueen 1967], and, interestingly, it refers to the optimal learning schedule. However, since we are further interested in quantizing the query space in light of prediction (i.e., minimizing also the ECC $\mathcal{E}_2$), we also update the query prototypes that are not winners of incoming pairs. This *weighted* update of all prototypes driven by the current prediction error (discussed later) results in the update of all counters $n_j(t)$ of the M prototypes at step t based on their probability of assignment. In that case, we obtain that

$$n_j(t) \leftarrow n_j(t-1) + p(\mathbf{w}_j|\mathbf{q}), j \in [M]. \tag{14}$$

It is worth noting here that the update rule in (14) is close to the Kohonen Learning Algorithm (KLA) [Kohonen 2013], which reflects the importance of each prototype in the learning process with respect to its probability assignment $p(\mathbf{w}_j|\mathbf{q})$. The hyperbolic schedule of the learning rates $\eta(t)$, $t = 0, 1, \ldots$, for the prototypes in (13) and (14) is then updated by

$$\eta_j(t) = \frac{1}{n_j(t)}, \tag{15}$$

which reflects the weight of the probability assignment $p(\mathbf{w}_j|\mathbf{q})$.

4.2.3. Convergence of Assignment Probability. During the optimization process, the probability assignment $p(\mathbf{w}_j|\mathbf{q})$ in (10) starts with a low value of β, (e.g., $\beta(0) = 0.01$) for which $p(\mathbf{w}_j|\mathbf{q})$ is approximately uniform. That is, for low values of β, the query prototypes are not yet attracted to a certain partition, and they all migrate toward the pairs presented. As time progresses, β gradually increases and the query prototypes are separated from each other since $p(\mathbf{w}_j|\mathbf{q})$ begins peaking around the winner. In time, the query prototypes closer to pairs will take relatively larger steps, while the query prototypes that are far away will be increasingly less affected. In the limit as $t \to \infty$, we obtain that

$$p(\mathbf{w}_j|\mathbf{q}) = \begin{cases} 1, & \text{if } \|\mathbf{q} - \mathbf{w}_j\|_2 < \|\mathbf{q} - \mathbf{w}_i\|_2, \forall j \neq i, \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

The fact that the prototypes in $\mathcal{W}$ and $\mathcal{U}$ being updated are not the only winners is the key capability of the proposed optimization process to avoid local minima, thus requiring $\lim_{t\to\infty} \beta(t) = \infty$ [Rose et al. 1992]. As in simulated annealing [Lee et al.

2013], we adopt $\beta(t) = \beta_0 \ln(t + 1)$ to achieve global minimum for (11) with $0.1 \leq \beta_0 \leq 0.4$. It is worth noting that gradually as β increases, $p(\mathbf{w}_j|\mathbf{q}) \to 1$; thus, $\hat{f}_k$ with the query and cardinality prototypes from optimal α_k^* approximates the function f w.r.t. (1):

$$f(\mathbf{q}) \approx \hat{f}_k(\mathbf{q}; \alpha_k^*). \tag{17}$$

4.3. Instantaneous Prediction Feedback

In this section, we discuss how the update rule in (12), which corresponds to the quantization of the query space, depends on the instantaneous prediction error (feedback) in cardinality. (This reflects our principle in clustering the queries patterns in light of cardinality prediction and function estimation.) The update rule in (12) makes the amount of update of the query prototype $\mathbf{w}_j$ dependent on the absolute difference $|y - u_j|$ between the actual cardinality y of a data subspace defined by a query $\mathbf{q}$ and the associated cardinality u_j of the prototype $\mathbf{w}_j$. In this context, prototype $\mathbf{w}_j$ can either move toward query $\mathbf{q}$ or move away from it based on the instantaneous prediction u_j.

To further demonstrate the behavior of (12), consider the case where only the two closest prototypes to $\mathbf{q}$ are used for instantaneous prediction, say $\mathbf{w}_j$ being the closest and $\mathbf{w}_i$ being the second closest, and neglect the remaining prototypes. In this case, our (generic) model coarsely would reduce to the regression LVQ [Grbovic and Vucetic 2009], where parameter β does not follow the rules of simulated annealing and the minimization of $\mathcal{E}_1$ is not pursued. If both prototypes are in similar proximity to $\mathbf{q}$, their assignment probabilities will be approximately the same, $p(\mathbf{w}_j|\mathbf{q}) = p(\mathbf{w}_i|\mathbf{q}) = 0.5$. We then obtain the update rules for those prototypes and the corresponding cases:

$$\begin{cases} \Delta\mathbf{w}_j \propto -(|y - u_j| - |y - u_i|) \\ \Delta\mathbf{w}_i \propto +(|y - u_j| - |y - u_i|) \end{cases} \tag{18}$$

—If instantaneous prediction errors $e_j = |y - u_j|$ and $e_i = |y - u_i|$ of those prototypes are similar, then the second closest prototype $\mathbf{w}_i$ will not be updated, i.e., it will not follow the pattern of the query $\mathbf{q}$. On the other hand, in this case, the winner (closest) prototype $\mathbf{w}_j$ will follow the pattern of the query $\mathbf{q}$ only through the rule $\Delta\mathbf{w}_j \propto -(\mathbf{q} - \mathbf{w}_j)$ (since in this case $Y_j = 1$), which reduces to the "standard" adaptive vector quantization rule in Rose et al. [1992] minimizing $\mathcal{E}_1$.
—If the instantaneous prediction error of $\mathbf{w}_j$ is larger than that of $\mathbf{w}_i$, it moves away from the query pattern $\mathbf{q}$ and, on the other hand, the prototype $\mathbf{w}_i$ moves toward the query pattern $\mathbf{q}$. This is somewhat similar to the intuition adopted in LVQ2.1 [Kohonen 2013] (but dealing with binary classification).
—In the case the instantaneous prediction error of $\mathbf{w}_j$ is smaller than that of $\mathbf{w}_i$, the former moves toward the pattern query $\mathbf{q}$ to follow the trend in the query space in a competitive manner (as a reward of a "good" instantaneous prediction plus a unity factor Y_j since $\mathbf{w}_j$ minimizes also $\mathcal{E}_1$), while the latter moves away from the query pattern $\mathbf{q}$.

The update rule for the cardinality prototypes in (13) always forces u_j to follow the actual cardinality y scaled by an amount of how close winner $\mathbf{w}_j$ is to query $\mathbf{q}$.

Figure 5 shows how the query prototypes are updated based on the feedback from the instantaneous cardinality prediction and the competitive-learning quantization of the query space.

4.4. Convergence Analysis

In this section, we discuss the fundamental features of the proposed function estimation model in terms of convergence of both types of prototypes: query and cardinality

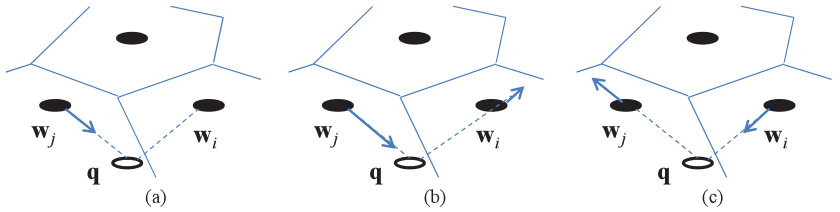


Fig. 5. Movements (toward/away from a query $\mathbf{q}$) due to feedback of the closest $\mathbf{w}_j$ and the second closest $\mathbf{w}_i$ query prototypes when instantaneous prediction errors (a) $e_j = e_i$, (b) $e_j < e_i$, and (c) $e_j > e_i$.

prototypes. As briefly reported in Sections 4.1.1 and 4.1.2, the query prototypes are converged to the mean (centroid) query representative of the analyst-defined data subspaces. Meanwhile, the corresponding cardinality prototypes are converged to the median of the cardinality values of those data subspaces.

The competing prototypes $\mathbf{w}_j$ and their associated u_j converge to the centroids and medians corresponding to local maxima of the sampled but unknown probability density $p(\mathbf{q}, y)$. When there are more prototypes than probability maxima, the query prototypes cluster about local probability maxima. We prove this equilibrium theorem of centroid and median convergence of our model.

THEOREM 4.6 (CENTROID CONVERGENCE THEOREM). *If vector $\tilde{\mathbf{q}}_j$ is the centroid of the query subspace of the winner $\mathbf{w}_j = [\mathbf{a}_j, \mathbf{b}_j]$ and $\mathbf{w}_j = [\mathbf{x}_j, \theta_j]$ corresponding to the data subspace $\mathbb{D}(\mathbf{a}_j, \mathbf{b}_j)$ and $\mathbb{D}(\mathbf{x}_j, \theta_j)$, respectively, then $P(\mathbf{w}_j = \tilde{\mathbf{q}}_j) = 1$ at equilibrium.*

PROOF. See Appendix B. □

Theorem 4.6 states that $\mathbf{w}_j$ is the mean vector of those query vectors $\mathbf{q}$ that were *classified* as $\mathbf{w}_j$ while partitioning the query space into M query prototypes. More analytically, for $\mathbf{w}_j = [\mathbf{a}_j, \mathbf{b}_j]$, we obtain that $\mathbf{a}_j = \frac{1}{N}\sum_{n\in[N]} \mathbf{a}_n$ and $\mathbf{b}_j = \frac{1}{N}\sum_{n\in[N]} \mathbf{b}_n$, with $\mathbf{q}_n = [\mathbf{a}_n, \mathbf{b}_n] : \|\mathbf{q}_n - \mathbf{w}_j\|_2 < \|\mathbf{q}_n - \mathbf{w}_i\|_2$ and $i \neq j$. In a similar way, for $\mathbf{w}_j = [\mathbf{x}_j, \theta_j]$, we obtain that: $\mathbf{x}_j = \frac{1}{N}\sum_{n\in[N]} \mathbf{x}_n$ and $\theta_j = \frac{1}{N}\sum_{n\in[N]} \theta_n$, with $\mathbf{q}_n = [\mathbf{x}_n, \theta_n] : \|\mathbf{q}_n - \mathbf{w}_j\|_2 < \|\mathbf{q}_n - \mathbf{w}_i\|_2$ and $i \neq j$. The query space partitioning is also driven by the instantaneous prediction error since we are about to estimate the function f and not just partitioning the query space. Through that prediction-driven quantization, the cardinality prototypes converge to the robust statistic of median.

THEOREM 4.7 (ASSOCIATED MEDIAN CONVERGENCE THEOREM). *If $\tilde{y}_j$ is the median of the cardinality values of the analyst-defined data subspaces $\mathbb{D}(\mathbf{a}_j, \mathbf{b}_j)$ and $\mathbb{D}(\mathbf{x}_j, \theta_j)$ with associated winner query prototypes $\mathbf{w}_j = [\mathbf{a}_j, \mathbf{b}_j]$ and $\mathbf{w}_j = [\mathbf{x}_j, \theta_j]$, respectively, then $P(u_j = \tilde{y}_j) = 1$ at equilibrium.*

PROOF. See Appendix C. □

5. TRAINING AND PREDICTION ALGORITHM

5.1. Training Algorithm

In this section, we provide a training algorithm for estimating the query and cardinality prototypes that gradually minimize the joint optimization function in (11). We are about to train K function estimation models $\hat{f}_k$ for the K distributed data nodes S_k.

The training algorithm for a given $\hat{f}_k$ is provided in Algorithm 1, and the training process is illustrated in Figure 6 (left). This algorithm is the same for both query types. Initially, all prototypes (vectors and scalars) in α_k are randomly initiated. Then, upon arrival of a random query $\mathbf{q}$ to the CN, the CN forwards that query $\mathbf{q}$ to each data node $S_k, k \in [K]$, in parallel as shown in Figure 6 (left). Then, the S_k node either executes

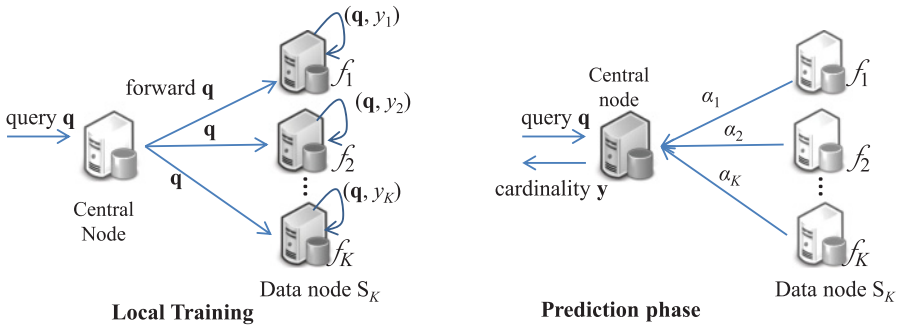


Fig. 6. Decentralized training of the function estimation models and prediction over K distributed data nodes.

the query and generates the cardinality of the answer set y or activates some well-known statistical structure, e.g., histograms, wavelet-based structures, to generate the aggregate answer y. (In the latter case, it is assumed that S_k node *owns* the data $\mathcal{B}_k$.) In any case, the training pair $(\mathbf{q}, y)$ is locally formed on each data node. This training pair is directed to incrementally update the local parameter α_k, through instantaneous prediction feedback adopted by the local update rules provided by (12) and (13).

The training algorithm on each node S_k processes successive random training pairs until a termination criterion Θ_t holds true. This criterion, which is compared to an accuracy threshold $\epsilon > 0$, refers to the L_1 distance between successive estimates of the prototypes:

$$\Theta_t = \sum_{j \in [M]} \big(\|\mathbf{w}_j(t) - \mathbf{w}_j(t-1)\|_1 + |u_j(t) - u_j(t-1)| \big), \tag{19}$$

with $\|\mathbf{w}_j\|_1 = \sum_{i \in [d]} |w_{ji}|$.

Remark 5.1. The termination criterion Θ_t is adopted to represent the discretized step-difference of the gradient of $\mathcal{E}_0$ consisting of the step difference of the scalar cardinality prototypes Δu_j and the query prototypes $\Delta \mathbf{w}_j = [\Delta w_{j1}, \ldots, \Delta w_{jd}]$ derived from Theorem 4.4. The training algorithm terminates when the average step difference, i.e., absolute difference of successive values, for each dimension Δw_{ji} and Δu_j is less than pre-defined scalars, which sum up to Θ_t. We consider equal importance of the convergence of all dimensions in the query and cardinality prototypes; thus, Θ_t is represented through the sum of the L_1 norms of the expanded difference vectors $[\Delta \mathbf{w}_j, \Delta u_j]$, $j = 1 \in [M]$.

Remark 5.2. During the training phase, data node S_k does not send back the pair $(\mathbf{q}, y)$ to the CN. The data node locally updates its parameter α_k. Once the model $\hat{f}_k$ is locally trained, data node S_k sends the model parameter α_k to the CN, as shown in Figure 6 (right) for proceeding with prediction.

The number of prototypes M is not necessarily the same for all estimation models $\hat{f}_k$. The models might have different number of prototypes. This flexibility allows each model to independently determine the quality (resolution) of quantization. Moreover, all models $\hat{f}_k$ are different since they refer to different datasets $\mathcal{B}_k$. The query and cardinality prototypes of each α_k reflect the probability distributions $p_k(\mathbf{q})$ and $p_k(\mathbf{q}, y)$, which are estimated given the executed queries and results over $\mathcal{B}_k$.

Once the training algorithm for a model $\hat{f}_k$ has converged with respect to convergence threshold ϵ, the prototypes of parameter α_k estimate the actual function f in both: query space (input of f) and cardinality domain (output of f). Figure 7 shows the *positions* of

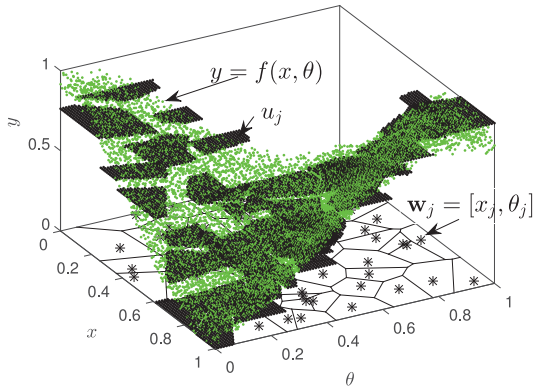


Fig. 7. Function estimation of the actual function $y = f(x, \theta)$ (green dots) through the $\hat{f}(x, \theta)$ (black dots) over the radius query-cardinality space. The 2-dimensional query prototypes $\mathbf{w}_j = [x_j, \theta_j]$ (black stars) have quantized the query space (x, θ). Each query prototype is associated with a cardinality prototype u_j, which is represented by a bounded plane (cell) in the 3-dimensional space over the quantized cell represented by the prototype $\mathbf{w}_j$.

$M = 50$ radius query prototypes along with their corresponding cardinality prototypes in the 3-dimensional query-cardinality space. Specifically, one can observe the median value u_j corresponding to a query prototype $\mathbf{w}_j = [x_j, \theta_j]$ projected onto a 2-dimensional query space of (x, θ), i.e., for radius queries over 1-dimensional data. Evidently, the actual function $f(x, \theta)$ (notated by green dots) is estimated through diverse horizontal planes/surfaces (notated by black dots) over the query subspaces through the trained representatives $\mathbf{w}_j$ (notated by black stars) and their trained corresponding cardinality value u_j on the cardinality y axis.

Now, given this function estimation, the idea of predicting the cardinality of an incoming unknown query is to interpolate among those planes whose corresponding prototypes are close to the query along with the degree of overlapping of the query-defined data subspace with neighboring data subspaces. This will be elaborated in the following section.

5.2. Prediction Algorithm

After the local training of a function estimation model $\hat{f}_k$, no other queries are directed from the CN to the data node S_k to be executed. Hence, no other updates occur to both types of prototypes in the α_k parameter.

Now, we have obtained *all* the available information to proceed with cardinality prediction given an unseen query $\mathbf{q}$ by approximation. This approximation is achieved by two principles: (i) exploitation of the degree of overlapping between the unseen analyst-defined data subspace and a data subspace defined by a query prototype, and (ii) exploitation of the similarity/distance function between the unseen query and a query prototype. The former principle refers to the interpolation of the diverse overlapping identified data subspaces corresponding to query prototypes of past issues analysts' queries with the unseen one. In this context, we inspect the overlapping over the data space. The latter principle deals with the degree of similarity of the unseen query with those query prototypes, whose corresponding data subspaces are overlapped with the unseen analyst-defined data subspace. In this context, we rest on the fundamental assumption in function estimation that similar inputs correspond to similar outputs. Hence, cardinality prediction fuses those principles over both data space and query space in order to approximate $\hat{y}$ given an unseen query $\mathbf{q}$ with actual cardinality y.

ALGORITHM 1: Training Algorithm for Function Estimation $\hat{f}_k$

Input: number of prototypes M, accuracy threshold ϵ, initial parameter β_0
Output: Parameter $\alpha_k = \{\mathbf{w}_j\}_{j=1}^{M} \cup \{u_j\}_{j=1}^{M}$
Random initialization: $\mathbf{w}_j(0), u_j(0), j \in [M]$;
$\eta_j(0) \leftarrow 0.5, j \in [M]$;
$t \leftarrow 1$;
repeat
  Central Node receives and forwards random query $\mathbf{q}(t)$ to node S_k;
  Data node S_k executes $\mathbf{q}(t)$ over $\mathcal{B}_k$;
  Formation of the training pair $(\mathbf{q}(t), y(t))$;
  for $j \in [M]$ **do**
    Calculate instantaneous prediction error $|y - u_j|$;
    Update prototypes $(\mathbf{w}_j(t), u_j(t))$, $j \in [M]$ using (12) and (13);
    Update learning rate $\eta_j(t)$ using (14) and (15);
  end
  Update parameter $\beta(t) \leftarrow \beta_0 \ln(t+1)$;
  Calculate termination criterion Θ_t using (19);
  $t \leftarrow t + 1$;
until $\Theta_t \leq \epsilon$;

5.2.1. Degree of Data Subspace Overlapping. Let us consider the range queries first. By Definition 1.4, two range queries $\mathbf{q}$ and $\mathbf{q}'$ overlap if $\mathcal{A}(\mathbf{q}, \mathbf{q}') = \texttt{TRUE}$. In this case, for each dimension $i \in [d]$, there are intervals defined by $[a_i, b_i]$ and $[a'_i, b'_i]$, respectively, that overlap. If we consider the fraction of the common area of these two intervals out of the maximum covered area of these intervals for a dimension i, then we define as *degree of overlapping* for range queries the normalized ratio $\delta(\mathbf{q}, \mathbf{q}') \in [0, 1]$:

$$\delta(\mathbf{q}, \mathbf{q}') = \begin{cases} \frac{1}{d} \sum_{i=1}^{d} \frac{\min(|b'_i - a_i|, |b_i - a'_i|)}{\max(|b'_i - a_i|, |b_i - a'_i|)}, & \text{if } \mathcal{A}(\mathbf{q}, \mathbf{q}') = \texttt{TRUE}, \\ 0, & \text{if } \mathcal{A}(\mathbf{q}, \mathbf{q}') = \texttt{FALSE}. \end{cases} \tag{20}$$

The two range data subspaces $\mathbb{D}(\mathbf{a}, \mathbf{b})$ and $\mathbb{D}(\mathbf{a}_j, \mathbf{b}_j)$ defined by the range query $\mathbf{q}$ and a range query prototype $\mathbf{w}_j$, respectively, correspond to the highest degree of overlap when $\delta(\mathbf{q}, \mathbf{w}_j) = 1$.

In the case of the radius queries, by Definition of 1.7, two radius queries $\mathbf{q}$ and $\mathbf{q}'$ overlap if $\mathcal{A}(\mathbf{q}, \mathbf{q}') = \texttt{TRUE}$. In order to quantify a degree of overlapping between hyperspheres, we require that the two balls are partially intersected. Let us define the ratio between the L_2 distance of the centers of the corresponding radius data subspaces out of the distance of their corresponding radii, i.e., $\frac{\|\mathbf{x} - \mathbf{x}'\|_2}{\theta + \theta'}$. This ratio takes values in $[0, 1]$ in the case of overlapping, with a value of unity when both spheres just meet each other. In the concentric case, the degree of overlapping should also take into consideration the remaining area from this perfect inclusion. Hence, we define as *degree of overlapping* for radius queries the normalized ratio $\delta(\mathbf{q}, \mathbf{q}') \in [0, 1]$:

$$\delta(\mathbf{q}, \mathbf{q}') = \begin{cases} 1 - \frac{\max(\|\mathbf{x} - \mathbf{x}'\|_2, |\theta - \theta'|)}{\theta + \theta'}, & \text{if } \mathcal{A}(\mathbf{q}, \mathbf{q}') = \texttt{TRUE}, \\ 0, & \text{if } \mathcal{A}(\mathbf{q}, \mathbf{q}') = \texttt{FALSE}. \end{cases} \tag{21}$$

The two radius data subspaces $\mathbb{D}(\mathbf{x}, \theta)$ and $\mathbb{D}(\mathbf{x}_j, \theta_j)$ defined by the radius query $\mathbf{q}$ and a radius query prototype $\mathbf{w}_j$, respectively, correspond to the highest degree of overlap when $\delta(\mathbf{q}, \mathbf{w}_j) = 1$.

5.2.2. Prediction via Approximation. Given an unseen query $\mathbf{q}$, the degree of overlapping with a query prototype is to calculate an affinity measure with respect to the

analyst-defined data subspace. Moreover, since we have partitioned the query space into certain regions to further capture the behavior of the actual function f over those regions, we have to guide our prediction over the quantized values of f within those regions. In our case, these quantized values refer to the medians u_j. Hence, the prediction via approximation is achieved by a weighted-nearest-neighbors regression, where the notion of neighborhood is guided by (W1) the subset of query prototypes whose data subspaces overlap with that of the unseen query (data space) and (W2) the closeness of the query prototypes with the unseen query (query space).

For the weight W1 case, we defined the set $\mathcal{O}(\mathbf{q})$ of overlapping data subspaces defined by the query prototypes in $\mathcal{W}$ with that defined by the query $\mathbf{q}$ to assign an affinity weight with respect to the degree of overlapping δ, i.e.,

$$\mathcal{O}(\mathbf{q}) = \{\mathbf{w} \in \mathcal{W} : \delta(\mathbf{q}, \mathbf{w}) > 0\}. \tag{22}$$

Now, for the weight W2 case, we exploit the probability assignment $p(\mathbf{w}|\mathbf{q})$ in (10) for those query prototypes $\mathbf{w} \in \mathcal{O}$ to assign a distance/similarity weight. Note, here that the β parameter in the probability assignment $p(\mathbf{w}|\mathbf{q})$ is set to the *last* $\beta(\tau)$ value right after the end of the training process of the Algorithm 1, i.e., $\beta(\tau)$ with $\tau = \min\{t \geq 1 : \Theta_t \leq \epsilon\}$.

Our prediction relies on the weighted-nearest-neighbors regression utilizing the cardinalities prototypes and the query prototypes by:

—(W1) taking into account the normalized degree of overlapping $\delta(\mathbf{w}, \mathbf{q})$ of unseen query $\mathbf{q}$ with those prototypes $\mathbf{w} \in \mathcal{O}$ and
—(W2) the probability assignment $p(\mathbf{w}|\mathbf{q})$ in (10) for $\mathbf{w} \in \mathcal{O}$ with $\beta(\tau)$.

We define the *fused* weight $W(\mathbf{w}_j, \mathbf{q}) \in [0, 1]$, with $\mathbf{w}_j \in \mathcal{O}$, which equally balances between weights W1 and W2:

$$W(\mathbf{w}_j, \mathbf{q}) = \frac{1}{2}\Big(\frac{\delta(\mathbf{w}_j, \mathbf{q})}{\sum_{\mathbf{w}_i \in \mathcal{O}} \delta(\mathbf{w}_i, \mathbf{q})} + p(\mathbf{w}_j|\mathbf{q})\Big). \tag{23}$$

In this context, for a function estimation $\hat{f}_k$, given an unseen query $\mathbf{q}$, the prediction $\hat{y}_k$ of the corresponding data subspace cardinality is

$$\hat{y}_k \;=\; \hat{f}_k(\mathbf{q}; \alpha_k) = \sum_{\mathbf{w}_j \in \mathcal{O}} W(\mathbf{w}_j, \mathbf{q}) u_j. \tag{24}$$

Note that, the overlapping set $\mathcal{O}$ is not the same for each prediction. Evidently, it depends on the possible overlapping data subspaces with the queried one. In the case where the overlapping set $\mathcal{O}$ is empty for a given unseen query $\mathbf{q}$, the *best* possible solution that our function estimation model $\hat{f}_k$ can provide is predicting $\hat{y}_k$ with the cardinality u_j of the closest (under L_2) query prototype $\mathbf{w}_j \in \mathcal{W}$. That is in this particular case,

$$\hat{y}_k \;=\; \hat{f}_k(\mathbf{q}; \alpha_k) = u_j : \mathbf{w}_j = \arg\min_{i \in [M]} \|\mathbf{q} - \mathbf{w}_i\|_2. \tag{25}$$

For a data node S_k, the prediction error (absolute difference) is then $|\hat{y}_k - y|$. The cardinality prediction $\hat{y}_k$, which corresponds to data node S_k for a given query $\mathbf{q}$, is achieved only by the CN and no communication with the node is performed. Specifically, the CN, which has stored the estimated parameters α_k, $\forall k$, given a query $\mathbf{q}$, calculates the overlapping sets $\mathcal{O}_k$ and proceeds with the (local) predictions of each $\hat{y}_k$ using either (24) or (25); see Figure 6 (right). Then, the CN provides the predicted cardinality vector:

$$\hat{\mathbf{y}} \;=\; [\hat{y}_1, \ldots, \hat{y}_K], \tag{26}$$

and the global L_1 prediction error is

$$\|\mathbf{y} - \hat{\mathbf{y}}\|_1 = \sum_{k=1}^{K} |y_k - \hat{y}_k|. \tag{27}$$

The cardinality prediction algorithm on the CN is provided in Algorithm 2.

Remark 5.3. Not having a CN for temporarily storing all parameters during the training phase yields higher reliability and less interruption during failures. During the prediction phase, the CN does not communicate with those nodes that have already sent their parameter α_k. Section 7 discusses certain decentralized training and prediction issues along with fault tolerance.

ALGORITHM 2: Cardinality Prediction Algorithm based on Estimation $\hat{f}_k$ on Central Node

Input: Query and cardinality prototypes of parameter α_k, an unseen (range or radius) query $\mathbf{q}$
Output: Prediction of the data subspace cardinality $\hat{y}_k$ defined by query $\mathbf{q}$
Calculation of the overlapping set $\mathcal{O}(\mathbf{q})$ in (22);
if $\mathcal{O}(\mathbf{q}) = \emptyset$ **then**
  Find the closest query prototype $\mathbf{w}_j \in \mathcal{W}$ to query $\mathbf{q}$;
  Predict $\hat{y}_k$ using (25);
end
else
  Calculate the fused weight $W(\mathbf{w}_j, \mathbf{q})$ for each $\mathbf{w}_j \in \mathcal{O}(\mathbf{q})$;
  Predict $\hat{y}_k$ using (24);
end

6. CHANGE DETECTION AND ADAPTATION

We introduce both detection and adaptation mechanisms for each function estimation model $\hat{f}_k$. These mechanisms will give the opportunity for $\hat{f}_k$ to stay up-to-date, i.e., (i) detect *new* popular data subspaces that are of interest to the analysts and (ii) autonomously adapt the trained parameter α_k through following the trend of the analysts to explore unseen data subspaces. A change detection in query patterns is carried out at the CN only. Meanwhile, the adaptation to $\hat{f}_k$ due to detected changes in the query patterns is achieved using adaptation pairs $(\mathbf{q}, y)$ that guide the adaptation of α_k. The adaptation should be driven by the fact that the *updated* function estimation $\hat{f}_k$ should continue to minimize the EPE in (1). This naturally requires communication between the CN and S_k.

6.1. Changes in Query Subspaces

In this section, we discuss how, through our model, we are able to recognize that an unexplored data subspace is gradually considered to be of interest to an analyst by issuing therein range and/or radius queries. Let us illustrate how the CN detects a change in the query patterns corresponding to a data node S_k based only on the available information reflected by the parameter α_k.

Consider an incoming query $\mathbf{q}$ to the CN and focus on a trained $\hat{f}_k$, i.e., operating in prediction mode. The essence of the proposed query subspace detection approach lies in two components:

—First, to decide on whether $\mathbf{q}$ is a candidate *novel* query with respect to the current model $\hat{f}_k$.
—Second, to track over time the number of such *novel* queries and decide that some query subspaces corresponding to data subspaces defined by those novel queries have been posed by the analysts.

Consider the probability assignment of query $\mathbf{q}$ to a winner query prototype $\mathbf{w}^*$ under L_2 after the $\hat{f}_k$ training. This assignment is provided with probability $p(\mathbf{w}^*|\mathbf{q})$ in (10). However, such an assignment is *reconsidered* if $\mathbf{q}$ is *distant* from its winner. Evidently, after training (i.e., convergence of $p(\mathbf{w}^*|\mathbf{q})$ to (16) as discussed in Section 4.2.3), this consideration yields useful information on how close query $\mathbf{q}$ is to prototype $\mathbf{w}^*$. The distance $\|\mathbf{q} - \mathbf{w}^*\|_2$ quantifies the likelihood that query $\mathbf{q}$ is expected to be drawn from $p(\mathbf{q}|\mathbf{w}^*)$ given that $\mathbf{q}$ is assigned to $\mathbf{w}^*$. To decide whether query $\mathbf{q}$ can be properly represented by prototype $\mathbf{w}^*$, we rely on the idea of Adaptive Resonance Theory [Carpenter and Grossberg 1988] for reasoning about *novelty*. We associate the winner $\mathbf{w}^*$ with a dynamic *vigilance* variable $\rho^* > 0$, which depends on the distance of the assigned query $\mathbf{q}$ to $\mathbf{w}^*$. The vigilance variable is a normalized L_2^2 centroid squared distance ratio of $\|\mathbf{q}-\mathbf{w}^*\|_2^2$ out of the average squared distances of all n queries $\mathbf{q}_\ell$, $\ell = 1, \ldots, n$, that were assigned (classified) to prototype $\mathbf{w}^*$ after its centroid convergence, i.e., the variance of those L_2 distances. In this context, we obtain that

$$\rho^* = \frac{\|\mathbf{q} - \mathbf{w}^*\|_2^2}{\frac{1}{n}\sum_{\ell=1}^{n} \|\mathbf{q}_\ell - \mathbf{w}^*\|_2^2}. \tag{28}$$

Based on this variance ratio, if ρ^* is less than a threshold $\rho > 0$, the query $\mathbf{q}$ is properly represented by its winner. Otherwise, the query $\mathbf{q}$ is currently deemed as *novel*. A ρ value normally ranges between 2.5 and 5 [Newton et al. 1992].

Let us now move to keeping track of query novelty candidates over time focusing on a winner prototype $\mathbf{w}^*$. For the t-th incoming query $\mathbf{q}$, which is assigned to winner $\mathbf{w}^*$, we define as novelty indicator of $\mathbf{q}$ with respect to $\mathbf{w}^*$ the random variable:

$$I_t = \begin{cases} 1, & \text{if } \|\mathbf{q} - \mathbf{w}^*\|_2^2 > \rho \frac{1}{n}\sum_{\ell=1}^{n} \|\mathbf{q}_\ell - \mathbf{w}^*\|_2^2 \\ 0, & \text{otherwise} \end{cases} \tag{29}$$

A cumulative sum of I_t's with a high portion of 1's causes the CN to consider that the conditional probability $p(\mathbf{q}|\mathbf{w}^*)$ within the query subspace represented by the winner $\mathbf{w}^*$ for model $\hat{f}_k$ might have changed. More specifically, upon reception of a query $\mathbf{q}$, the CN observes for a winner query prototype $\mathbf{w}^*$ the random variables $\{I_1, \ldots, I_t\}$ in each α_k. In this context, the CN detects a change in the $p(\mathbf{q})$ within the query subspace $\mathbb{Q}^*$ of each model $\hat{f}_k$ based on the cumulative sum $\mathcal{S}_t$ of the variables $I_1, I_2, \ldots, I_t$ up to t-th assigned query, i.e.,

$$\mathcal{S}_t = \sum_{\tau=1}^{t} I_\tau. \tag{30}$$

A change in $p(\mathbf{q})$ for model $\hat{f}_k$ is independent of a change in $p(\mathbf{q})$ for another model $\hat{f}_m$, since both query distributions are reflected by different query prototypes; recall, the query space quantization is driven by the prediction error, which results to different parameters α_k and α_m, respectively.

The I_t indicator for a model $\hat{f}_k$ is a discrete random process with independent and identically distributed (i.i.d.) samples; queries independently arrive at the CN. Each variable I_t follows an unknown probability distribution depending on the distance of the query $\mathbf{q}$ to is winner $\mathbf{w}^*$. The I_t random variable has finite mean $\mathbb{E}[I_t] < \infty$, $t = 1, \ldots$, which depends on the squared L_2 norm $\|\mathbf{q}_t - \mathbf{w}^*\|_2^2$. Specifically, the expectation of the novelty indicator is

$$\mathbb{E}[I] = 0 \cdot P(\{I = 0\}) + 1 \cdot P(\{I = 1\}) = P(\{I = 1\}),$$

where $\{I = 1\}$ denotes that an assigned query to a winner prototype is classified as novelty. Our knowledge on that distribution, which is not trivial to estimate, will provide

us certain insight to judge whether the conditional distribution $p(\mathbf{q}|\mathbf{w}_j)$ has changed in the subspace determined by query prototype $\mathbf{w}_j, \forall j$. In this case, we should "follow" the trend of that change by updating the winner prototype $\mathbf{w}^*$ only of a subspace, in order for that prototype to continue represent its query subspace.

By observing the random variables I_t and deriving the sum S_t up to t defined in (30), the challenge here is to decide how large the sum should get before deciding that $p(\mathbf{q})$ has changed for those $\mathbf{q} \in \mathbb{Q}^*$. Should we decide at an early stage that $p(\mathbf{q})$ has changed, this might correspond to "premature" decision on the fact that $p(\mathbf{q})$ has changed. A relatively small number of novelties might not correspond to change in $p(\mathbf{q})$ over a query subspace. On the other hand, should we "delay" our decision on a possible change of $p(\mathbf{q})$ in $\mathbb{Q}_*$, we might get high prediction errors since we avoid adapting $\mathbf{w}^*$ to "follow" the trend of the query subspace change.

Our idea for dealing with this time-driven decision making problem is as follows: To decide when the $p(\mathbf{q}|\mathbf{w}^*)$ has changed over the monitored query subspace $\mathbb{Q}^*$, we could wait for an *unknown* finite horizon t^* in order to be more confident on a $p(\mathbf{q})$ change. During the t^* period, we only observe the cumulative sums $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{t^*}$. In this context, we study a stochastic optimization model that that postpones a query distribution change decision (reflecting possible novelties) through additional observations of the I_τ variables. At time t^*, a decision on a possible $p(\mathbf{q})$ change has to be taken. The problem is to find that optimal decision time t_*, hereinafter referred to as *optimal stopping time*, in order to ensure that $p(\mathbf{q})$ has changed from those incoming queries $\mathbf{q}_t$ assigned to $\mathbf{w}^*$ at $t > t^*$.

Let us define our confidence Y_t of a decision on a change of the $p(\mathbf{q})$ within $\mathbb{Q}^*$, based on the cumulative sum S_t. The Y_t confidence could be directly connected to the prediction accuracy performance improvements that a timely decision yields. Obviously, Y_t is a random variable generated by the sum of the random variables I_τ up to t, $\mathcal{S}_t = \sum_{\tau=1}^{t} I_\tau$, discounted by a risk factor $\gamma \in (0, 1)$:

$$Y_t = \gamma^t \mathcal{S}_t, t \geq 1. \tag{31}$$

Our detection model has to find an optimal stopping time t^* in order to start adapting the winner prototype $\mathbf{w}^*$ after considering that $p(\mathbf{q})$ has changed within the query subspace $\mathbb{Q}^*$. If we never start/initiate this adaptation, then our confidence that we follow the *new* trend (query patterns) is zero, $Y_\infty = 0$. This indicates that we do not "follow" the trend of a possible change over the $\mathbb{Q}^*$ subspace. Furthermore, we will never start/initiate adapting prototype $\mathbf{w}^*$ at some t with $\mathcal{S}_t = 0$, since, obviously, there is no piece of evidence of any novelty up to t. As the indicator I_t assumes unity values for certain time instances, $\mathcal{S}_t$ increases at a high rate, thus indicating a possible change due to a significant number of novelties. Our problem is, then, to decide how large $\mathcal{S}_t$ should get before we start/initiate adapting $\mathbf{w}^*$. That is, we have to find a time $t > 0$ that maximizes our confidence, i.e., to find the optimal stopping time t^* that maximizes the essential supremum in (32):

$$t^* = \arg ess \sup_{t \in \mathcal{X}_0} \mathbb{E}[Y_t], \tag{32}$$

where $\mathcal{X}_0$ is the set of almost surely finite stopping times that are greater than 0, which itself may be a stopping time. The semantics of the risk factor γ in (31), also involved in (32), are as follows: A high risk factor value indicates a conservative time-optimized decision model. This means that the model requires additional observations for concluding on a $p(\mathbf{q})$ change decision. This, however, comes at the expense of possible prediction inaccuracies during this period since $\mathbf{w}^*$ might not be a representative of its corresponding assigned queries. A low γ value denotes a rather optimistic model, which reaches premature decisions on a $p(\mathbf{q})$ change. This means that, once we

concluded on a change, we have to adapt the winner $\mathbf{w}^*$ by actually executing every incoming query assigned to $\mathbf{w}^*$ and getting the corresponding actual cardinality. This continues until the updated $\mathbf{w}^*$ converges (discussed later). In the following section, we attempt to approach an optimal stopping time that maximizes (32) for a fixed risk factor $\gamma \in (0, 1)$.

6.2. Optimally Scheduled Query Subspace Change Detection

We propose an optimally scheduled query subspace change detection model that solves the problem in (32). Such problem is treated as an infinite horizon Optimal Stopping Time (OST) problem with discounted future reward [Peskir and Shiryaev 2006]. Initially, we briefly discuss OST. Then, we prove the existence of the optimal stopping time t_* in our problem, report on the corresponding optimal stopping rule, elaborate on the optimality of the proposed scheme, and provide the corresponding algorithm for the CN.

6.2.1. Optimal Stopping Theory. The OST deals with the problem of choosing the best time instance to take the decision of performing a certain action. This decision is based on sequentially observed random variables in order to maximize the expected reward [Peskir and Shiryaev 2006]. For given random variables $X_1, X_2, \ldots$ and measurable functions $Y_t = \mu_t(X_1, X_2, \ldots, X_t)$, $t = 1, 2, \ldots$ and $Y_\infty = \mu_\infty(X_1, X_2, \ldots)$, the problem is to find a stopping time τ to maximize $\mathbb{E}[Y_\tau]$. τ is a random variable with values in $\{1, 2, \ldots\}$ such that the event $\{\tau = t\}$ is in the Borel field (filtration) $\mathbb{F}_t$ generated by $X_1, \ldots, X_t$, i.e., the only available information we have obtained up to t:

$$\mathbb{F}_t = \mathbb{B}(X_1, \ldots, X_t). \tag{33}$$

Hence, the decision to stop at t is a function of $X_1, \ldots, X_t$ and does not depend on future observables $X_{t+1}, \ldots$. The theorem in Chow et al. [1971] refers to the existence of the optimal stopping time.

THEOREM 6.1 (EXISTENCE OF OPTIMAL STOPPING TIME). *If* $\mathbb{E}[\sup_t Y_t] < \infty$ *and* $\lim_{t\to\infty} \sup_t Y_t \leq Y_\infty$ *almost surely (abbreviated as a.s.), then the stopping time* $t^* = \inf\{t \geq 1 | Y_t = ess\sup_{\tau \geq t} \mathbb{E}[Y_\tau | \mathbb{F}_t]\}$ *is optimal.*

PROOF. See Chow et al. [1971]. □

The (essential) supremum $ess\sup_{\tau\geq t} \mathbb{E}[Y_\tau|\mathbb{F}_t]$ is taken over all stopping times τ such that $\tau \geq t$ a.s. The optimal stopping time t^* is obtained through the "principle of optimality" [Bertsekas 2005].

6.2.2. An Optimal Stopping Time for Query Subspace Change Detection. The proposed detection model concludes on a change of $p(\mathbf{q}|\mathbf{w}^*)$ based on sequential observations of I_t values. Persisting unity values of I_t make ourselves confident on a change. On the other hand, as we delay our decision on a change, the observation process progresses further, thus yielding possible low prediction accuracy. A decision taken at time t is:

—either to assert that a change on $p(\mathbf{q}|\mathbf{w}^*)$ holds true and, then, start the adaptation of the winner prototype $\mathbf{w}^*$,
—or continue the observation process at time $t + 1$ and, then, proceed with a decision.

We will show that our model based on the cumulative sums $\mathcal{S}_t = \sum_{\tau=1}^{t} I_\tau$ can determine an optimal stopping time that maximizes (32).

THEOREM 6.2. *An optimal stopping time for the query subspace change/novelty detection problem in (32) exists.*

PROOF. See Appendix D. □

In our case, I_t are non-negative; thus, the problem is *monotone* Chow et al. [1971]. In such case, the optimal stopping time t^*, since it exists by Theorem 6.2, is obtained by the *1-stage look-ahead optimal rule* (1-sla) [Li and Zhang 2005]. That is, we should start adapting the winner prototype $\mathbf{w}^*$ at the *first* stopping time t at which $Y_t \geq \mathbb{E}[Y_{t+1}|\mathbb{F}_t]$, i.e.,

$$t^* = \inf\{t \geq 1 | Y_t \geq \mathbb{E}[Y_{t+1}|\mathbb{F}_t]\}. \tag{34}$$

In our context, for a monotone stopping problem with observations $I_1, I_2, \ldots$ and rewards $Y_1, Y_2, \ldots, Y_\infty$, the 1-sla is optimal since $\sup_t Y_t$ has finite expectation, which is $\mathbb{E}[I]\frac{\gamma}{1-\gamma}$, and $\lim_{t\to\infty} \sup_t Y_t = Y_\infty = 0$ a.s.; please refer to proof of Theorem 6.2.

THEOREM 6.3. *The optimal stopping time t^* for the query subspace change/novelty detection problem in (32) is*

$$t^* \;=\; \inf\left\{t \geq 1 | S_t \geq \frac{\gamma}{1-\gamma}\mathbb{E}[I]\right\}. \tag{35}$$

PROOF. See Appendix E. □

Since our optimal stopping time problem exists and is monotone according to Theorems 6.2 and 6.1, the provided optimal stopping time in 6.3 is unique.

LEMMA 6.4. *The solution of the optimal stopping time t^* provided in Theorem 6.3 is unique.*

PROOF. See Darling et al. [1972]. □

In order to derive the optimal stopping time and rule in Theorem 6.3, we have to estimate the expectation of the novelty indicator $\mathbb{E}[I] = P(\{I = 1\})$. Empirically, the probability of the event $P(\{I = 1\})$ can be experimentally calculated by those assigned queries whose ratio of the squared distances from their winner prototypes out of the total variance of the distances is at least ρ; please refer to (29).

We now further provide an estimate for $P(\{I = 1\})$ based on the fundamental characteristics of our query quantization model. The probability of a query $\mathbf{q}$ being assigned to a winner prototype $\mathbf{w}^*$ is $p(\mathbf{w}^*|\mathbf{q})$ provided in (10). The probability of the event $\{I_t = 1\}$ actually refers to the conditional probability of an incoming query $\mathbf{q}_t$ be a novelty given that it is assigned to $\mathbf{w}^*$ with $p(\mathbf{w}^*|\mathbf{q}_t)$. In that case, the probability $P(\{I_t = 1\})$ is, therefore, associated with the probability that the L_2^2 norm distance $\|\mathbf{q}_t - \mathbf{w}^*\|_2^2 > \theta$, with scalar $\theta = \rho\frac{1}{n}\sum_{\ell=1}^{n}\|\mathbf{q}_\ell - \mathbf{w}^*\|_2^2$. If we define the random vector $\mathbf{z}_t = \mathbf{q}_t - \mathbf{w}^*$, then we seek the probability density distribution of its squared L_2^2 norm $\|\mathbf{z}_t\|_2^2$. Therefore, based on the centroid convergence Theorem 4.6, the prototype $\mathbf{w}^*$ refers to the centroid (mean) vector of those $\mathbf{q} \in \mathbb{Q}^*$, i.e., $\mathbf{w}^* = \mathbb{E}[\mathbf{q}; \mathbf{q} \in \mathbb{Q}^*]$. In this context, the squared L_2^2 norm of the multivariate $\mathbf{z} = [z_1, \ldots, z_d] = [q_1 - w_1^*, \ldots, q_d - w_d^*]$ vector under the assumption of normally distributed random components follows a non-central squared chi distribution $\chi^2(d, \zeta)$ with d degrees of freedom and non-centrality parameter $\zeta = \sum_{i=1}^{d}(w_i^*)^2$. We can then approximate the probability of a novelty as

$$P(\{I = 1\}) \;=\; P(\|\mathbf{z}\|_2^2 > \theta) = 1 - P(\|\mathbf{z}\|_2^2 \leq \theta), \tag{36}$$

with $P(\|\mathbf{z}\|_2^2 \leq \theta)$ being the cumulative distribution function $CDF_{\chi^2(d,\zeta)}(\theta) = P(\|\mathbf{z}\|_2^2 \leq \theta)$ of $\chi^2(d, \zeta)$.

Let now $Q_{\kappa_1}(\kappa_2, \kappa_3)$ be the monotonic, log-concave Marcum Q-function, with parameters κ_1, κ_2, and κ_3. Then, we obtain that $CDF_{\chi^2(d,\zeta)}(\theta) = P(\|\mathbf{z}\|_2^2 \leq \theta) = 1 - Q_{\kappa_1}(\kappa_2, \kappa_3)$

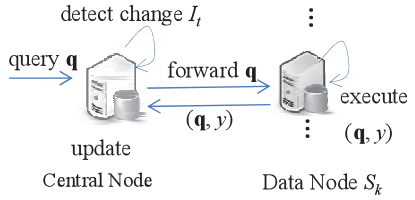


Fig. 8. The query patterns detection running on the CN and adaptation communicating with the data node S_k.

and, therefore, the probability of novelty is

$$P(\{I = 1\}) = 1 - CDF_{\chi^2(d,\zeta)}(\theta) = Q_{\frac{d}{2}}\left(\sqrt{\zeta}, \sqrt{\theta}\right) \tag{37}$$

by substitution in the Q function: $\kappa_1 = \frac{d}{2}$, $\kappa_2 = \sqrt{\zeta}$, and $\kappa_3 = \sqrt{\theta}$. For an analytical expression of (37), please refer to Appendix F.

The novelty probability in (37) depends on the θ quantity that quantifies the average variance of all queries assigned to $\mathbf{w}^*$ and the squared norm of $\mathbf{w}^*$. Once we have estimated that probability $P(\{I = 1\})$, the optimal stopping rule is obtained in Theorem 6.3 by $\mathbb{E}[I] = P(\{I = 1\})$ from (37) and (35).

6.3. Query Subspace Change and Adaptation Algorithm

The CN detects a change in the query subspace related to a function estimation model $\hat{f}_k$ by evaluating the optimal stopping rule provided in (35). This is achieved locally on the CN and no communication with the corresponding data node S_k is required. Once the CN has detected a change in the query subspace, it initiates a process that adapts the model $\hat{f}_k$ by modifying the query prototype $\mathbf{w}^*$ of the parameter α_k as follows.

A change in the query subspace indicates that new query patterns can be formed or existing patterns change reflected by changes in $p(\mathbf{q}|\mathbf{w}^*)$. Moreover, we have to associate the cardinalities of these queries to the cardinality prototypes. This requires modifications to the cardinality prototypes corresponding to updated $\mathbf{w}^*$. For adaptation purposes, the CN communicates with the data node S_k, for which a change has been detected, for executing every incoming query $\mathbf{q}$ appearing at $t > t^*$. After each query execution, the CN obtains the *adaptation* pair $(\mathbf{q}, y)$ to update only the winner prototype $\mathbf{w}^*$ and the associated cardinality prototype u^* of α_k to follow the actual cardinality y of $\mathbf{q}$. This is achieved by applying the update rules (12) and (13) on the winner prototype pair $(\mathbf{w}^*, u^*)$. This adaptation continues until $\hat{f}_k$ converges with respect to the winner prototype only, i.e., when the L_1-norm successive estimates of the winner $\mathbf{w}^*$ and cardinality u^* prototypes, i.e.,

$$\Theta_t^* = \|\mathbf{w}^*(t) - \mathbf{w}^*(t-1)\|_1 + |u^*(t) - u^*(t-1)|, t > t^*, \tag{38}$$

do not exceed the accuracy threshold ϵ.

Figure 8 illustrates the detection and adaptation processes for a specific function estimation model $\hat{f}_k$ for data node $\mathcal{S}_k$. The corresponding time-optimized query change detection and adaptation algorithm is provided in Algorithm 3 for the model $\hat{f}_k$.

7. DISTRIBUTED PLATFORM ISSUES

A key feature of our approach so far is its design to derive independent local models $\hat{f}_k$ at each data node $\mathcal{S}_k$ with parameters α_k, which then inform the CN that uses them to perform cardinality prediction. Such a distributed framework entails issues

ALGORITHM 3: Query Change Detection/Adaptation Algorithm for Model $\hat{f}_k$

Input: Risk factor $\gamma \in (0, 1)$, accuracy threshold ϵ
Output: Updated parameter α_k
$\theta \leftarrow \rho \frac{1}{n} \sum_{\ell=1}^{n} \|\mathbf{q}_\ell - \mathbf{w}^*\|_2^2$ from (29);
Calculate $\mathbb{E}[I]$ using (37) or, empirically;
$t \leftarrow 0$;
$\mathcal{S}_0 \leftarrow 0$;
/*Novelty detection*/;
repeat
  Central Node receives a query $\mathbf{q}(t)$;
  Assign the query $\mathbf{q}(t)$ to its winner prototype $\mathbf{w}^*$ of α_k;
  Calculate the novelty indicator I_t using (29);
  $\mathcal{S}_t \leftarrow \mathcal{S}_t + I_t$;
  $t \leftarrow t + 1$;
until $\mathcal{S}_t \geq \frac{\gamma}{1-\gamma}\mathbb{E}[I]$;
/*Novelty detected; starting adaptation*/;
$t \leftarrow 0$;
repeat
  Central Node receives a query $\mathbf{q}(t)$;
  Central Node forwards query $\mathbf{q}(t)$ to data node S_k;
  Data node S_k executes the query $\mathbf{q}(t)$ over $\mathcal{B}_k$ and generate the adaptation pair $(\mathbf{q}(t), y(t))$;
  Central Node updates the prototypes pair $(\mathbf{w}^*(t), u^*(t))$ using (12) and (13);
  Calculate termination criterion Θ_t^*;
  $t \leftarrow t + 1$;
until $\Theta_t^* \leq \epsilon$;

arising from the independent operation (during the training and prediction phases) of a number of independent nodes. Notably, these involve operation in the face of (i) data node failures and (ii) desynchronized model convergence among nodes.

7.1. Training and Prediction Phases for Desynchronized Convergence

The entire system runs in two phases: In the training phase, data nodes learn their local models $\hat{f}_k$ (i.e., their parameters α_k) as queries arrive from the CN. In the prediction phase, the CN performs cardinality prediction upon a new query $\mathbf{q}$, thus returning the prediction vector $\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_K]^\top$ in (26).

Given that the model operates on a distributed environment, we account for the possibility that not all local models $\hat{f}_k$ will converge at the same time (i.e., after processing the same number of training pairs). This can be due to different number of prototypes for each $\hat{f}_k$, different accuracy thresholds ϵ_k, networking connectivity resources (e.g., bandwidth) to the remote data nodes, computational resources (e.g., query execution engines), and/or different underlying data distributions $p(\mathcal{B}_k)$. This desynchronized per-node model convergence affects the convergence rate and the expected optimization error bounds of the training phase of each function estimation model. Therefore, we distinguish between the *individual* and *combined* operating modes for training and prediction phases.

7.1.1. Individual Operational Mode. In the individual mode, the CN "waits" for all per-node models to converge, i.e., to reach the number of training pairs T_k required for optimization accuracy ϵ_k. Under SGD over the objective minimization function $\mathcal{E}_0$ in (11) with the hyperbolic learning schedule in (15), a model $\hat{f}_k$ requires $O(1/\epsilon_k)$ [Bousquet and Bottou 2008] number of training pairs T_k to reach accuracy ϵ_k. This means that the residual error/difference between $\mathcal{E}_0^{T_k}$ after T_k pairs and the optimal $\mathcal{E}_0^*$ (i.e., with the

optimal parameter α_k^*) asymptotically decreases exponentially (also known as *linear convergence*; Jr. and Schnabel [1983]). Hence, the CN will be in its training phase with a duration of $O(\max_{k\in[K]}(1/\epsilon_k))$. Once all models have converged and sent their parameters α_k to the CN, the latter moves to its prediction phase. Thus, in individual mode, there is a clear separation between the training and prediction phases. In this mode also, the CN is capable of returning the prediction $\hat{\mathbf{y}}$ to an incoming query $\mathbf{q}$. Moreover, the upper bound of the expected excess error $\mathbb{E}[\mathcal{E}_0^T - \mathcal{E}_0^*]$ for each function estimation model, after $T = \max(T_1, \ldots, T_K)$ training pairs, is $O(\sqrt{\frac{\log T}{T}})$ [Vapnik and Chervonenkis 1971], given a hyperbolic learning schedule. Nonetheless, the CN will move into prediction phase only *when* all models have converged, which might not be desirable.

7.1.2. Combined Operational Model. To avoid high delays in the training phase, we also allow the combined mode of operation, whereby the training and prediction phases overlap. The idea is to enable CN to proceed with partial cardinality prediction even when not all local function estimation models have converged. In the combined mode, the CN runs its training algorithm, and when, say $\ell, 1 \leq \ell < K$, local models have converged, it moves to a mode of operation where cardinality prediction is executed by returning only the prediction $\hat{\mathbf{y}}' = [\hat{y}_1, \ldots, \hat{y}_\ell]^\top$ of those ℓ models upon a query $\mathbf{q}$. In addition, the CN still forwards that query to those $K - \ell$ nodes, whose models have not yet converged, allowing thus their continued training.

Since $\hat{\mathbf{y}}'$ does not contain the predictions of all models, the CN uses the parameters of the $K - \ell$ models that have not yet converged as *approximate* prediction $\hat{\mathbf{y}}'' = [\hat{y}_{\ell+1}, \ldots, \hat{y}_K]^\top$. Let $T' = \max(T_1, \ldots, T_\ell)$ be the number of training pairs required for all $\ell < K$ models to converge and $T = \max(T_1, \ldots, T_K)$ be the total number of training pairs for all models to converge as discussed in the individual mode, i.e., $T' < T$. Then, under a hyperbolic learning schedule in (15), the expected excess error upper bound for the $K - \ell$ models that have not yet converged is given by $O(\sqrt{\frac{\log \lambda T}{\lambda T}})$ [Vapnik and Chervonenkis 1971] with $\lambda = \frac{T'}{T}$. This information is useful for judging the quality of the approximate prediction $\hat{\mathbf{y}}'' = [\hat{y}_{\ell'}]_{\ell'=\ell+1}^K$ w.r.t. accuracy threshold $\epsilon_{\ell'}$. The CN continues to run on individual mode and will eventually move to being only in its prediction phase, i.e., $\ell = K$. Note that T is often not known in advance and is determined empirically (e.g., till satisfactory performance is obtained).

7.2. Failures and Fault Tolerance

First, note that the existence of a CN is in line with the architectural paradigms of most popular Big Data platforms: Hadoop [White 2009; Dean and Ghemawat 2008]; Yarn [Vavilapalli et al. 2013]; Spark [Zaharia et al. 2012], Stratosphere [Alexandrov et al. 2014]; and AsterixDB [Alsubaiee et al. 2014], which have a master/coordinator and a set of workers. Second, note that failures, by design, impose little disruption to our prediction task. Specifically, the failure of any data node S_k (or equivalently, a partition failure preventing S_k and CN from communicating), once S_k's model has been produced, leaves the CN's prediction task unaffected. The CN continues predicting S_k's response based on α_k sent to the CN prior to the failure. In most cases, S_k's will be swift. When recovering from a crash, S_k can then simply notify the CN that it is now available. In cases where S_k remains unavailable for extended periods of time, the CN can switch to a combined operation mode, whereby S_k is treated as if it had not yet converged, as discussed above. Note that this is required as CN cannot differentiate between a partition failure and a crash failure and must assume the worst (i.e., that S_k is live, possibly accepting data updates that may have changed its model significantly

since it was communicated to CN). When recovering from a partition failure, the S_k will need to have kept track of any local updates to its model's parameters α_k and send the new α_k to CN.

CN failures introduce minimal service disruptions. Akin to most Big Data platforms, this can be detected and a new CN is elected. Upon election, the new CN needs communicate with data nodes S_k and receive their models' parameters α_k. From this point onward, the CN can start performing cardinality prediction and change detection.

8. PERFORMANCE ANALYSIS

The function estimation model's training phase requires $O(dM)$ space and $O(1/\epsilon)$ [Bousquet and Bottou 2008] iterations to get $\Theta_t \leq \epsilon$ (convergence). The CN prediction per range and radius query is performed in $O(dM)$ time for evaluating the overlapping set $\mathcal{O}$. Detection and adaptation, on the other hand, requires $O(d \log M)$ time to search for the winner prototype using a 1-nearest neighbor search for the winner by adopting a $2d$-dimensional and $(d+1)$-dimensional tree structure over the range and radius query prototypes, respectively.

We will next show that by extracting knowledge from the pairs $(\mathbf{q}, y)$ without accessing the underlying data, our approach achieves similar or even better prediction accuracy while adapting to query patterns changes. We first turn to study the model's performance sensitivity on (i) cardinality prediction accuracy, (ii) adaptation capability to query patterns changes, (iii) required storage, (iv) number of training pairs, and (v) training and prediction time, when training/predicting with real-world datasets.

We subsequently provide a comparative assessment of our approach versus data-centric approaches (despite their aforementioned lack of applicability to reduced/restricted data access environments). We consider range queries for (i) GenHist histogram [Gunopulos et al. 2005], (ii) learning frameworks for STHs [Viswanathan et al. 2011], (iii) ISOMER STH [Srivastava et al. 2006], and radius queries for (iv) sampling using the reservoir sampling [Vitter 1985].

The relative absolute prediction error e_k for a model $\hat{f}_k$ is defined as the ratio of the absolute-deviation loss $|y_k - \hat{y}_k|$ out of the actual cardinality y_k corresponding to the dataset $\mathcal{B}_k$ of data node S_k:

$$e_k(y_k, \hat{y}_k) = \frac{|y_k - \hat{y}_k|}{y_k}.$$

We adopt this metric to evaluate the prediction efficiency of our model as introduced in Gunopulos et al. [2005] and Viswanathan et al. [2011] for cardinality prediction to enable direct comparisons against those works. The total relative absolute prediction error for all K models is $e = \frac{1}{K} \sum_{k=1}^{K} e_k$.

8.1. Datasets and Workloads

The RS1 dataset from the UCI Repository (MLR)[3] contains multivariate data with dimension $d \in \{1, \ldots, 8\}$ and is used for performance analysis of our method and for comparison against the sampling method. The RS2 real dataset from UCI MLR[4] is used for comparison with GenHist ($d \in \{5, 10\}$). All points in RS2 are normalized in $[0, 1]$ as in the GenHist paper [Gunopulos et al. 2005]. The RS3 real dataset [Lichman 2013] from the UCI MLR consists of 3-dimensional points scaled in $[0, 1]$ and adopted for comparison with Viswanathan et al. [2011] and ISOMER.

[3]http://archive.ics.uci.edu/ml/machine-learning-databases/00235/.
[4]kdd.ics.uci.edu/summary.data.type.html.

We generate the training set $\mathcal{T}$ and a different evaluation set $\mathcal{E}$. The size of $\mathcal{T}$, $|\mathcal{T}|$, ranges between $0.25\%|\mathcal{E}|$ and $4\%|\mathcal{E}|$, with $|\mathcal{E}| = 2{\cdot}10^5$ for each workload (default value).[5] In order to demonstrate the capability of the proposed model to learn, adapt, and predict, we evaluate its performance using uniform, skewed, and multi-modal multivariate query distributions. Obviously, when the model has to estimate a cardinality function $f(\mathbf{q})$, where the queries $\mathbf{q}$ derive from a query distribution $p(\mathbf{q})$ with, say, zero degree of skewness (e.g., uniform distribution), the corresponding query prototypes can satisfyingly approximate $p(\mathbf{q})$ by evenly distributing the prototypes. The query training phase becomes demanding in terms of (a) minimizing the objective $\mathcal{E}_0$ and (b) identification of the query prototypes when $p(\mathbf{q})$ involves multi-modal distributions; i.e., consisting of query subspaces that generate queries with higher popularity than other subspaces. In such cases, the model has to learn those subspaces and intelligently allocate prototypes to better capture not only the statistical characteristics of those regions but also to associate query prototypes to cardinality representatives in light of cardinality prediction. We generate queries from a variable mixture of query distributions. In addition, in the comparative assessment against the data-centric methods, for objective comparison, we generate queries from distributions involving uniformity (as dictated by those papers). With this methodology, we extensively evaluate the capability of our model for cardinality prediction by learning the query to cardinality association.

The set $\mathcal{T}$ contains random queries drawn from a number of N *query subspaces* $\mathbb{Q}_n \subset \mathbb{Q}, n \in [N]$. For range queries, each $\mathbb{Q}_n$ is characterized by a generator $(\mathbf{c}_n, \mathbf{v}_n, \ell_n)$. The query center is sampled from a Gaussian $\mathcal{N}(c_{ni}, v_{ni})$ with mean c_{ni}, variance v_{ni}, and radius ℓ_{ni}, $i \in [d]$, i.e., lower bound $a_{ni} = x_{ni} - \ell_{ni}$ and upper bound $b_{ni} = x_{ni} + \ell_{ni}$, where center $x_{ni} \sim \mathcal{N}(c_{ni}, v_{ni})$. The query volume $2\ell_{ni}$ is drawn uniformly at random from 1% to 20% of the range of dimension i. To generate a range query, a $\mathbb{Q}_n$ is selected with probability $\frac{1}{N}$, where we obtain the lower, upper, and volume.

For radius queries, each query subspace $\mathbb{Q}_n$ is characterized by a generator $(\mathbf{c}_n, \mathbf{v}_n, \mu_{\theta_n}, \sigma_{\theta_n})$. For a radius query $\mathbf{q} \in \mathbb{Q}_n$, each dimension $x_i \sim \mathcal{N}_{ni}$ is sampled from Gaussian $\mathcal{N}_{ni} = \mathcal{N}(c_{ni}, v_{ni})$ with mean c_{ni} and variance v_{ni}. The corresponding radius θ is sampled from Gaussian $\mathcal{N}_\theta = \mathcal{N}(\mu_{\theta_n}, \sigma^2_{\theta_n})$. To generate a radius query, a $\mathbb{Q}_n$ is selected with probability $\frac{1}{N}$ and from the n-th generator we obtain $x_i \sim \mathcal{N}_{ni}, \forall i$, and $\theta \sim \mathcal{N}_\theta$.

The (range query) workload WL1 for ISOMER, EquiHist, and SpHist [Viswanathan et al. 2011] is generated with the exact same way as in Viswanathan et al. [2011]. In WL1, the center $c_i = \frac{a_i+b_i}{2}$ is selected uniformly at random from [0,1], $i \in [d]$; a query is a d-dimensional hyper-rectangle centered around c_i with volume $b_i - a_i$ at most 20% of [0,1]. The (range query) workload WL2 for GenHist is generated with the exact same way as in Gunopulos et al. [2005]. WL2 contains queries whose points are chosen uniformly at random in the data domain. The (radius query) workload WL3 for the sampling method contains radius queries from $N = 1{,}000$ subspaces.

The number of data nodes K ranges from 1 to 100. Each real dataset $\mathcal{B}$ is split into K disjoint subsets $\mathcal{B}_k$ (i.e., without replicates) distributed to the K data nodes, such that $\cup_{k=1}^K \mathcal{B}_k \equiv \mathcal{B}$ and $\cap_{k=1}^K \mathcal{B}_k \equiv \emptyset$. We assume that each function estimation model $\hat{f}_k$ has the same number of prototypes M, which ranges from 500 to 2,000. The vigilance threshold for novelty detection $\rho = 3$ as proposed in Newton et al. [1992]. The risk factor $\gamma = 0.9$ denotes a detection model, which prefers to "delay" its optimal decision in light of observing more pieces of evidence for novelty in query subspaces. The $1 - \gamma$ value reflects the probability of an analyst to start exploring *new* data subspaces, thus issuing analytics queries over those regions of data. The interested reader could also refer to Dubins and Teicher [1967] for an interpretation of a probabilistic modeling of

[5]NB: that $|\mathcal{T}|$ is only a very small percentage of $|\mathcal{E}|$.

Table I. Parameters

Parameter	Default values
Data dimension d	{2,3,4,5,8,10}
Dataset size $\lvert\mathcal{B}\rvert$	2.075.259 (RS1), 545.424 (RS2), 210.138 (RS3)
Evaluation set size and training set size	$\lvert\mathcal{T}\rvert = [0.25, 4]\%\lvert\mathcal{E}\rvert$, $\lvert\mathcal{E}\rvert = 2 \cdot 10^5$
Query subspaces N	{100, 500, 1,000, 5,000}
Prototypes M	{500, 1,000, 2,000}
Data nodes K	$\{1, \ldots, 100\}$
Accuracy threshold	$\epsilon = 10^{-3}$
Vigilance threshold	$\rho = 3$
Risk factor γ	0.9
Scalar $\theta \sim \mathcal{N}(\mu_\theta, \sigma_\theta^2)$	$\mu_\theta = 0.3$, $\sigma_\theta^2 = 0.1$

Table II. Statistics for the RS1, RS2, and RS3 Datasets

Statistic per dimension	Values
RS1: Mean ($d = 6$)	[1.3399 0.1114 3.6737 5.6449 7.6751 8.7356]
RS1: Std. deviation ($d = 6$)	[1.7440 0.1188 241.0857 7.3765 2.0798 8.2428]
RS1: Coeff. of variation ($d = 6$)	[0.7683 0.9381 0.0152 0.7652 3.6903 1.0598]
RS2: Mean ($d = 8$)	[2.9594 0.1557 0.0141 0.2694 0.0464 2.3501 0.2121 0.2233]
RS2: Std. deviation ($d = 8$)	[0.2800 0.1119 0.0075 0.2125 0.0583 1.5593 0.0268 0.0198]
RS2: Coeff. of variation ($d = 8$)	[0.0946 0.7190 0.5309 0.7889 1.2559 0.6635 0.1262 0.0885]
RS3: Mean, std., coeff. of variation x_1	(38.5816, 13.64, 0.3535)
RS3: # discrete textual values x_2, x_3	(16, 8)

observing novel pieces of evidence in the theory of optimal stopping. Table I shows the parameters for our experimentation.

8.2. Statistics of the Datasets

This section presents the basic statistics of the datasets RS1, RS2, and RS3 used in our experiments. For each dataset, we report on the dimensionality of the data space, the mean and standard deviation of each dimension and the corresponding *coefficient of variation* in Table II. The coefficient of variation is a standardized measure of dispersion of a probability distribution. It is expressed as the ratio of the standard deviation out of the mean value. A distribution with coefficient of variation less than 1 is considered low-variance, while with coefficient of variation greater than 1 refers to a high-variance distribution. Moreover, for each dimension, we illustrate the probability density and mass function for scalar and discrete dimensions in Figures 9, 10, and 11, for RS1, RS2, and RS3, respectively.

8.3. Accuracy of Data Subspace Cardinality Prediction

Figure 12(a)–(c) shows the impact of the number of prototypes M on error e with different number N of radius query subspaces for $d \in \{2, 3, 4\}$ over RS1, with $K = 50$ nodes and $\lvert\mathcal{T}\rvert = 0.1\%\lvert\mathcal{B}\rvert$. Each model achieves very low error as M increases. We observe an error lower than 5% for $d = 4$ and the model's robustness in terms of N. For $N = 5{,}000$, the error increases; thus, the models have to increase the number of prototypes M to capture all radius query subspaces. For $N < 5{,}000$, an increase in M is not mandatory to achieve lower error. Figure 12(d) shows the impact of the number of training pairs $\lvert\mathcal{T}\rvert$ with $\lvert\mathcal{T}\rvert \in [0.025\%, 0.4\%]\lvert\mathcal{B}\rvert$ on the average error for $K = 50$ nodes, $N = 5{,}000$, and $d \in \{2, 3, 4\}$. The more training pairs used, the more statistical information of the query subspaces can be captured by the prototypes. However, for

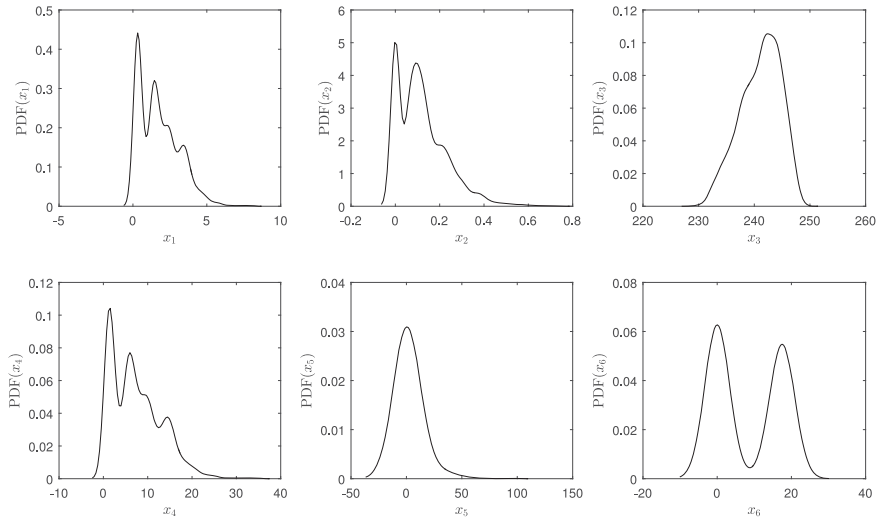


Fig. 9. The probability density function (PDF) for each dimension of the RS1 dataset.

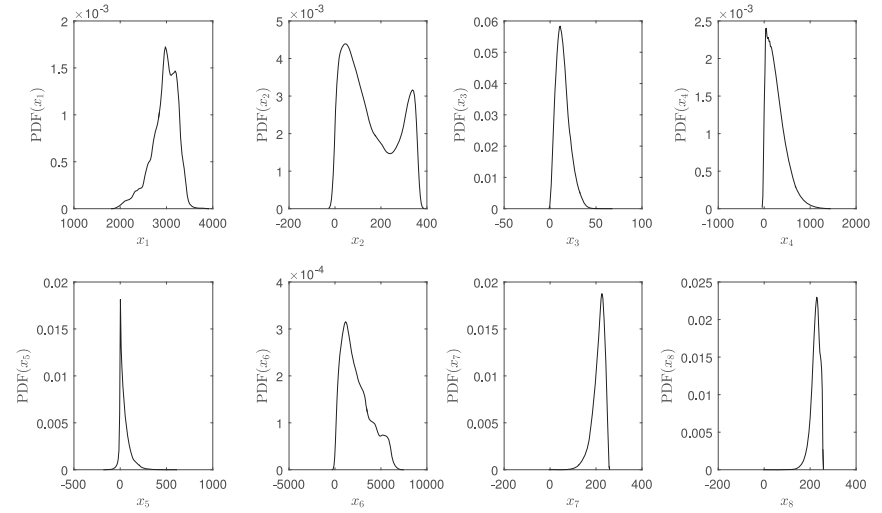


Fig. 10. The probability density function (PDF) for each dimension of the RS2 dataset.

$M = 1{,}000$, there is no need to increase the training set size since the models have captured the statistical properties of the actual query-cardinality function.

The prediction error depends on the resolution of quantization (M) and the training set size $|\mathcal{T}|$. Figure 13 shows the scalability of our method over radius and range queries with a varying number of nodes K for different N (Figure 13(a) and (b)) and different $|\mathcal{T}|$ (Figure 13(c) and (d)). The error e remains constant with the number of data nodes K, which denotes that any scale-out partition of a huge dataset into K disjoint subsets results to robust models. Each data node creates its own *local* function estimation model and all nodes, as a whole, achieve the similar cardinality prediction error as if it were a single node with one *global* function estimation model.

In terms of the curse of dimensionality, Figure 15(a) shows the accuracy of our model as the dimension increases over radius queries (RS1/WL3). With a high M, the

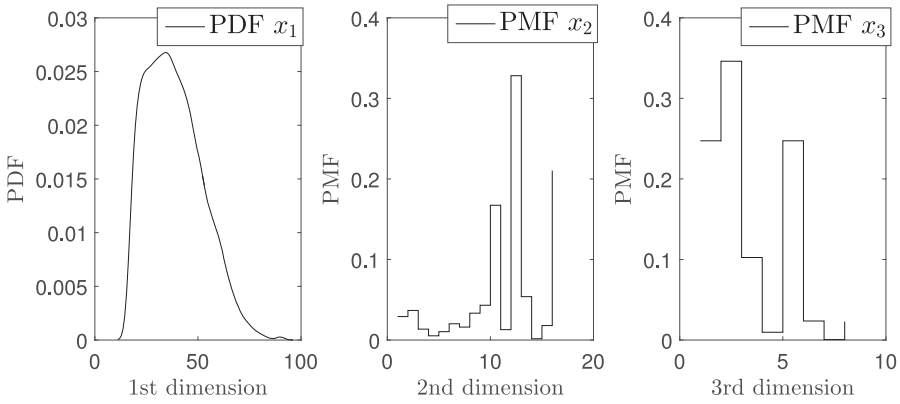
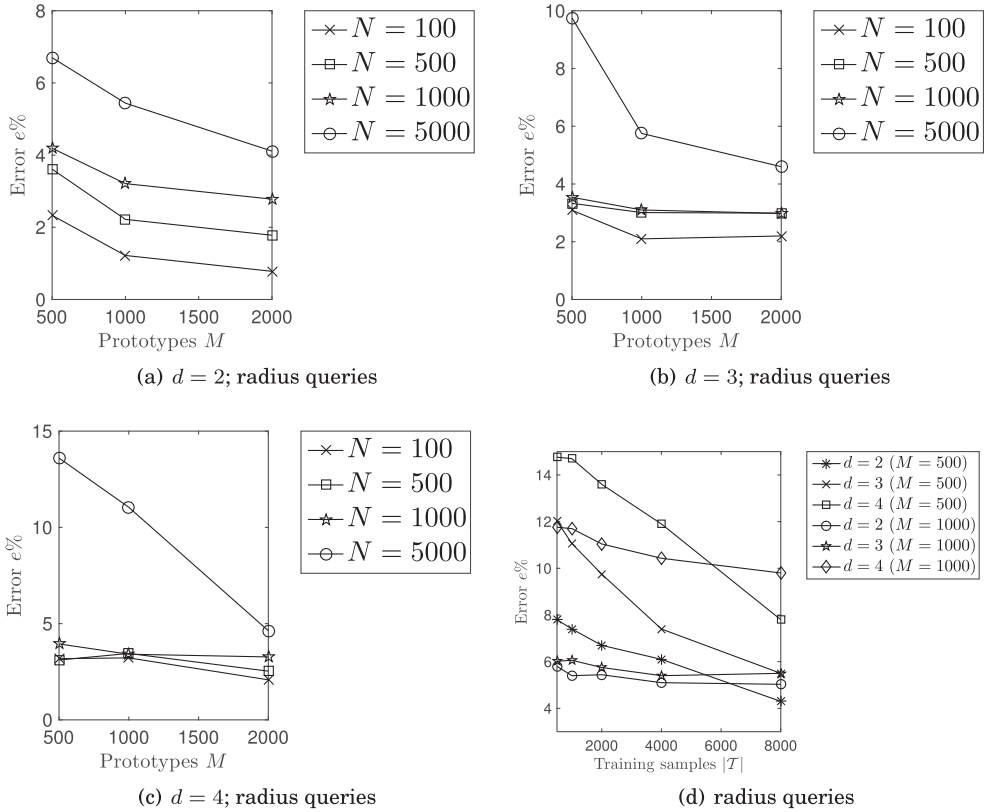


Fig. 11. The probability density function (PDF) for the 1st dimension and the probability mass function (PMF) for the 2nd and the 3rd dimensions of the RS3 dataset.



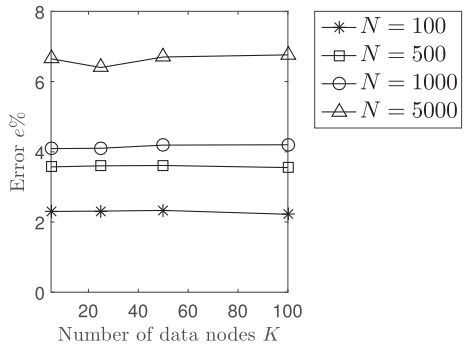
(a) $d = 2$; radius queries

(b) $d = 3$; radius queries

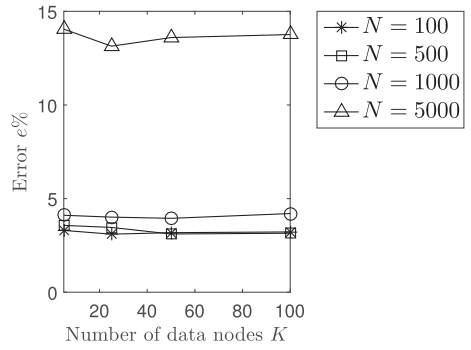
(c) $d = 4$; radius queries

(d) radius queries

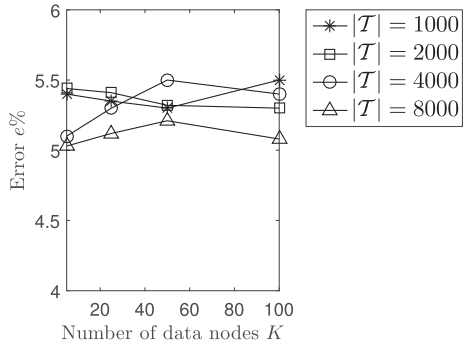
Fig. 12. (a–c) Error e vs. number of prototypes M (RS1, $K = 50$), (d) impact of $|\mathcal{T}|$ on error e ($d \in \{2, 3, 4\}$, $N = 5{,}000$) for radius queries.



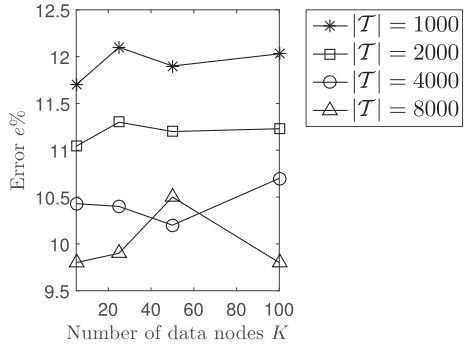
(a) $d = 2$; radius queries



(b) $d = 4$; range queries



(c) $d = 2$; radius queries



(d) $d = 4$; range queries

Fig. 13. (a, b) Number of nodes K vs. error e; $d \in \{2, 4\}$, RS1, $M = 500$, for radius and range queries. (c, d) Number of nodes K vs. error e; $M = 500$, $N = 1{,}000$, $d \in \{2, 4\}$, RS1, for radius and range queries.

model captures the diversity on the underlying mapping $\mathbf{q} \mapsto \mathbf{y}$, thus achieving small cardinality prediction error in high dimensions.

8.4. Training Time

The learning time (in seconds) is measured over RS1 on a PC Intel Core i5 CPU at 3.40GHz, 16GB RAM. For number of prototypes $M = 1{,}000$ and $|\mathcal{T}| = 3{,}000$ training pairs until convergence, the model takes (1.4s, 3.8s) in $d = 2$, and (2.8s, 6.7s) in $d = 4$ to be trained given $N = (100, 1{,}000)$ query subspaces, respectively. The learning time depends on N (variability of patterns): As N increases, the model must learn a "richer" query space, thus more training pairs are needed to converge. The cardinality prediction time for a (radius/range) query on average ranges in [0.16s, 1.14s]; $100 \leq M \leq 1{,}000$.

8.5. Query Subspace Detection Change and Adaptation Capability

In order to experiment with the behavior of the proposed time-optimized change detection and adaptation algorithm we proceed with the following. Consider the scenario shown in Figure 14 repeated 450 times. Focus on a model $\hat{f}$ trained with $M = 1{,}000$ prototypes, $|\mathcal{T}| = 2{,}000$ radius queries and $N = 1{,}000$ over RS1 and dimension $d = 4$. After training of the estimation model $\hat{f}$, the CN receives radius queries one at a time generated by a workload where we alter all parameters of the query subspaces $\mathbb{Q}_n$ $\forall n$; thus, queries are drawn from different distributions. After the first 100 queries, we alter all $\mathbb{Q}_n$. The model detects the changes after (on average) 30 *novel* queries, and

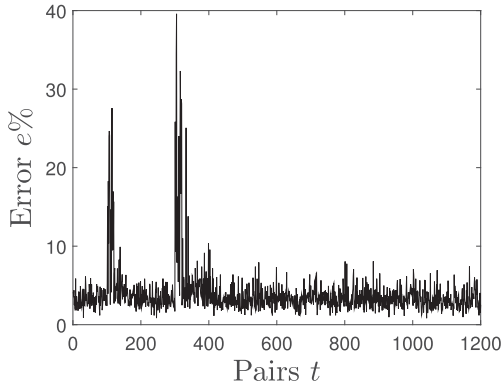


Fig. 14. Change detection and adaptation to radius query subspace changes ($d = 4$, RS1, $N = 1{,}000$, $M = 1{,}000$).

then it starts to adapt to this change using up to 56 queries (on average). At the 300th query, we alter again all $\mathbb{Q}_n$ and observe that the model detects the changes and is adjusted again, by proceeding with prediction errors equivalent with those before any query subspace change.

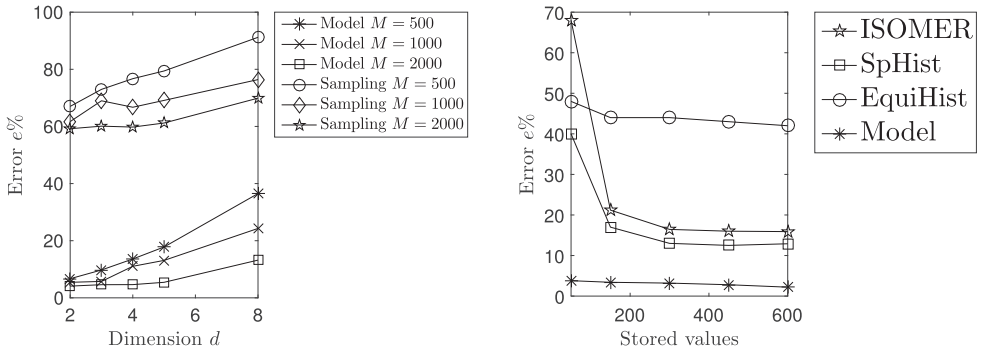
8.6. Models Comparison

We provide a comparative assessment against data-centric approaches. Despite their applicability shortfalls explained earlier, we aim to demonstrate our method's advantages.

8.6.1. Sampling and Histogram-Based Models. A sampling technique draws points from $\mathcal{B}$ randomly and uniformly without replacement and obtains a sample $\mathcal{B}'$. The cardinalities of $\mathcal{B}'$ are used as estimates for cardinality prediction over $\mathcal{B}$. We obtain $\mathcal{B}'$ using *reservoir sampling* [Vitter 1985]. The ISOMER [Srivastava et al. 2006], is a QFR STH, using the principle of maximum entropy to approximate the underlying probability density $p(\mathbf{x})$. The STHs framework in Viswanathan et al. [2011] uses QFRs for (i) the EquiHist algorithm, which learns a fixed size-bucket equi-width histogram and (ii) the SpHist algorithm, which uses Haar wavelets to construct a histogram. The GenHist histogram in Gunopulos et al. [2005] estimates the $p(\mathbf{x})$ by allowing the buckets to overlap. As most of the data-centric approaches have been proposed only for centralized systems, we compare our model with these data-centric centralized models by setting $K = 1$.

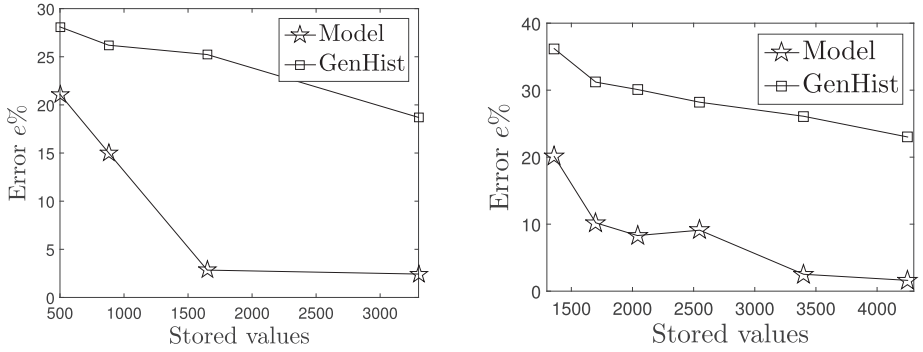
8.6.2. Comparative Assessment. The cardinality prediction error of the sampling method in Figure 15(a) is very high compared to our model for different dimensions of RS1 using WL3 given exactly the same number of stored points for radius queries (we observe similar results using range queries). The training set is $|\mathcal{T}| = 1\%|\mathcal{E}|$, with $|\mathcal{E}| = 2 \cdot 10^5$. The sample size is $|\mathcal{B}'| = M(1 + \frac{1}{d})$ since our model stores M vectors of $d+1$ dimension (radius query prototypes and cardinality prototypes) and, in sampling, we store data points of d dimensions. Sampling-based cardinality prediction apart from being inappropriate in data-restricted scenarios, it is not adaptable to updates in query patterns. The average cardinality prediction time per query for sampling is 38s, compared to 1.07s in our model.

We compare our model with EquiHist, SpHist, and ISOMER using the same RS3 as in Viswanathan et al. [2011] over range queries, with $|\mathcal{T}| = 0.2\%|\mathcal{B}|$ and $|\mathcal{E}| = 3\%|\mathcal{B}|$ over WL1. Figure 15(b) shows the impact of stored values on error for SpHist, EquiHist,



(a) radius queries (b) $d = 3$; range queries

Fig. 15. (a) Error e vs. dimension d over (RS1/WL3) for model ($K = 1$) and sampling ($N = 1{,}000$). (b) Error e vs. stored values for model ($K = 1$), SpHist, EquiHist, and ISOMER (RS3/WL1).



(a) $d = 5$; range queries (b) $d = 8$; range queries

Fig. 16. Error e vs. stored values for our model ($K = 1$) and GenHist; range queries over RS2 for dimension $d \in \{5, 8\}$.

ISOMER, and our model (storing $M(2d + 1)$ range query prototypes and cardinality prototypes). The histogram-based approaches use $B = 64$ buckets, where each bucket, in each dimension, consists of two boundary values and one value for the frequency. Hence, each histogram stores $B(2d + 1)$ values. The more information is stored by SpHist and ISOMER the less error they achieve. Our model achieves significantly lower error than the data-centric approaches for different numbers of stored values. SpHist, EquiHist, and ISOMER attempt to tune a histogram while our model learns from WL1.

Moreover, we compare our model with GenHist over RS2 (as used in Gunopulos et al. [2005]; normalized in [0,1]) using range queries from WL2. Figure 16 shows the error against stored values over $d \in \{5, 8\}$ data with $|\mathcal{T}| = 2{,}000$ corresponding to $0.3\%|\mathcal{B}|$ and $|\mathcal{E}| = 25|\mathcal{T}|$. The GenHist stores $B(2d + 1)$ values referred to B buckets, while our model stores $M(2d + 1)$ values corresponding to M ranging in [46,300] and [86, 262] for $d = 5$ and $d = 8$, respectively. Our model outperforms GenHist by achieving at most $(81, 66)\%$ lower error for $d = (5, 8)$ given the same memory size. An increase in $M > 200$ ($d \in \{5, 8\}$) does not contribute to better accuracy, thus, we could use fewer prototypes to learn WL2.

9. CONCLUSIONS AND FUTURE PLANS

With this work we focused on cardinality prediction over data subspaces defined by analysts' range and radius queries. This predictive learning process is central to predictive analytics, yet computing exact answers of these queries is very expensive and can cause scalability problems.

We have begun a novel investigation route, whereby results from previously executed analytics queries are exploited in order to predict future query results. Our method is query-driven involving training, making it applicable to emerging data environments where data accesses are undesirable or infeasible. The proposed method contributes to a function estimation model based on unsupervised regression integrated with supervised vector quantization. Such model (being decentralized) relies on independently operating data nodes' local models, which are subsequently submitted to a central node, based on which we perform the prediction task. The prediction task is achieved via approximation and weighted interpolation among the cardinality prototypes corresponding to the most popular data subspaces from the analysts' perspective. Our contribution also shows how to efficiently detect and adapt to updates in query-patterns, treating them as novelty under the principles of the theory of optimal stopping.

Comprehensive experiments showcase the model's robustness where it is shown to achieve small error rates with small memory footprints—even outperforming the data-centric state-of-the-art (despite their lack of applicability in the targeted data environments). Therefore, our model represents a solution that is applicable to data environments with undesirable/infeasible data accesses or not.

Our plans for future work focus on: (i) developing a framework that can dynamically and optimally switch between the training and query execution phases as analysts' interests shift between data subspaces, (ii) autonomously determining the resolution of the query space quantization based on the analysts' interest over data subspaces, (iii) evolving and expanding the fundamental representatives of both data and query subspaces for supporting robust query subspace adaptation, and (iv) dealing with data spaces with on-line data mutations (insertions, deletions, and updates).

APPENDIXES

A. PROOF OF THEOREM 4.4

We adopt the Robbins–Monro stochastic approximation for minimizing $\mathcal{E}_0$ using SGD over $\mathcal{E}_1$ and $\mathcal{E}_2$. Given the t-th training pair $(\mathbf{q}(t), y(t))$, the stochastic samples $E_1(t)$ and $E_2(t)$ of the corresponding $\mathcal{E}_1$ and $\mathcal{E}_2$ are

$$E_1(t) = \|\mathbf{q}(t) - \mathbf{w}^*(t)\|_2^2$$

and

$$E_2(t) = \sum_{j \in [M]} p(\mathbf{w}_j(t)|\mathbf{q}(t))|y(t) - u_j(t)|,$$

respectively. $E_1(t)$ and $E_2(t)$ have to decrease at each new pair at t by descending in the direction of their negative gradient with respect to $\mathbf{w}_j(t)$ and $u_j(t)$. Hence, the update rules for prototypes $\mathbf{w}_j$ and u_j are

$$\Delta \mathbf{w}_j(t) = -Y_j(t)\frac{1}{2}\eta_j(t)\frac{\partial E_1(t)}{\partial \mathbf{w}_j(t)} - \eta_j(t)\frac{\partial E_2(t)}{\partial \mathbf{w}_j(t)}$$

and

$$\Delta u_j(t) = -\eta_j \frac{\partial E_2(t)}{\partial u_j(t)}.$$

$Y_j(t) = 1$ if $\mathbf{w}_j(t)$ is the winner of $\mathbf{q}(t)$; 0, otherwise (since $E_1(t)$ involves only the winner prototype). Scalar $\eta_j(t)$ satisfies $\sum_{t=0}^{\infty} \eta_j(t) = \infty$ and $\sum_{t=0}^{\infty} \eta_j^2(t) < \infty$. From the partial derivatives of $E_1(t)$ and $E_2(t)$, we obtain (12) and (13). By starting with arbitrary initial training pair $(\mathbf{q}(0), y(0))$, the sequence $\{(\mathbf{w}_j(t), u_j(t))\}$ converges to optimal $\mathbf{w}_j$ and u_j parameters, $j \in [M]$.

B. PROOF OF THEOREM 4.6

Let the j-th winner prototype $\mathbf{w}_j$ reach equilibrium, i.e., $\Delta\mathbf{w}_j = \mathbf{0}$, which, in this case, holds with probability 1. Then, from (16), we obtain that the assignment probability $p(\mathbf{w}_j|\mathbf{q}) = 1$, while this probability is zero $\forall i \neq j$. Based on the update rule in (12), this yields that $\Delta\mathbf{w}_j \propto (\mathbf{q} - \mathbf{w}_j)$. By taking the expectation of both sides of $\Delta\mathbf{w}_j = \mathbf{0}$ at equilibrium, we have that

$$\mathbb{E}[\Delta\mathbf{w}_j] = \int_{\mathbb{Q}_j} (\mathbf{q} - \mathbf{w}_j) p(\mathbf{q}) \mathrm{d}\mathbf{q} = \int_{\mathbb{Q}_j} \mathbf{q} p(\mathbf{q}) \mathrm{d}\mathbf{q} - \mathbf{w}_j \int_{\mathbb{Q}_j} p(\mathbf{q}) \mathrm{d}\mathbf{q}.$$

By solving $\mathbb{E}[\Delta\mathbf{w}_j] = 0$, the $\mathbf{w}_j$ equals the centroid (mean) vector of all query vectors of the subspace $\mathbb{Q}_j$.

C. PROOF OF THEOREM 4.7

Let $\mathbb{Y}_j$ be the image of query subspace $\mathbb{Q}_j$ via the function $y = f(\mathbf{q})$. The median $\tilde{y}_j$ of the domain $\mathbb{Y}_j$ satisfies $P(y \geq \tilde{y}_j) = P(y \leq \tilde{y}_j) = \frac{1}{2}$. Suppose the j-th winner $\mathbf{w}_j$ and its associated cardinality u_j have reached equilibrium, i.e., $\Delta\mathbf{w}_j = 0$ and $\Delta u_j = 0$ hold with probability 1. In this case, the assignment probability $p(\mathbf{w}_j|\mathbf{q}) = 1$ and $Y_j = 1$ and zero for $i \neq j$. By taking the expectations of both sides and replacing $\Delta\mathbf{w}_j$ and Δu_j with the update rules from Theorem 4.4:

$$\mathbb{E}[\Delta u_j] = \int_{\mathbb{Y}_j} \mathrm{sgn}(y - u_j) p(y) \mathrm{d}y = P(y \geq u_j) \int_{\mathbb{Y}_j} p(y) \mathrm{d}y - P(y < u_j) \int_{\mathbb{Y}_j} p(y) \mathrm{d}y = 2P(y \geq u_j) - 1.$$

Since $\Delta u_j = 0$, thus u_j is constant, then $P(y \geq u_j) = \frac{1}{2}$, which denotes that u_j converges to the median of $\mathbb{Y}_j$.

D. PROOF OF THEOREM 6.2

Based on the theorem in Chow et al. [1971], we have to prove that the optimal stopping time t^* exists and is derived from the principle of optimality. Specifically, we have to prove that (i) $\lim_{t\to\infty} \sup_t Y_t \leq Y_\infty$ a.s. and (ii) $\mathbb{E}[\sup_t Y_t] < \infty$.

Note that I_t are non-negative and from the strong law of numbers $(\frac{1}{t}) \sum_{\tau=1}^{t} I_\tau \to \mathbb{E}[I] = P(\{I = 1\})$ a.s., so that

$$Y_t = t\gamma^t (S_t/t) \leq t\gamma^t (1/t) \sum_{\tau=1}^{t} I_k \simeq t\gamma^t \mathbb{E}[I] \xrightarrow{a.s.} 0$$

with $\lim_{t\to\infty} \sup_t Y_t = Y_\infty = 0$.

In addition, $\sup_t Y_t = \sup_t \gamma^t S_t \leq \sup_t \gamma^t \sum_{\tau=1}^{t} I_\tau \leq \sup_t \sum_{\tau=1}^{t} \gamma^\tau I_\tau \leq \sum_{\tau=1}^{\infty} \gamma^\tau I_\tau$. Hence,

$$\mathbb{E}[\sup_t Y_t] \leq \sum_{\tau=1}^{\infty} \gamma^\tau \mathbb{E}[I] = \mathbb{E}[I] \frac{\gamma}{1-\gamma} < \infty.$$

E. PROOF OF THEOREM 6.3

Let us first study the characteristics of the confidence values Y_t under the filtration $\mathbb{F}_t = \mathbb{B}(I_1, \ldots, I_t)$ referring to Lemmas 2–4 and Theorem 1 in Dubins and Teicher [1967].

We then provide the optimal stopping criterion in our case. Let $S_t = I_1 + \cdots + I_t$ and $\gamma^t = \prod_{\tau=1}^{t} \gamma_\tau$ with the conventions that $S_0 = 0$ and $\gamma^0 = 1$. In our case, I_t are non-negative and $\gamma_\tau = \gamma \in (0, 1)$ constant. We assume that at $t = 0$ our detection model initiates the observation for a specific query subspace $\mathbb{Q}^*$ represented by the query prototype $\mathbf{w}^*$ with some $\omega \in \{0, 1\}$ (i.e., initial state of the observation process). Then, we define the function

$$J(\omega) = \sup_t \mathbb{E}[\gamma^t(\omega + S_t)]$$

and t ranges over all stopping times for $I_1, I_2, \ldots$. The problem is to find an optimal t^* for initial state ω, that is, a t^* such that the $\sup_t \mathbb{E}[\gamma^t(\omega+S_t)]$ is attained. Interestingly, this problem has a solution given in Haggstrom [1967] in which t^* is evaluated by adopting the principle of optimality [Bertsekas 2005] provided that: (i) I_t are non-negative, (ii) $\omega = 0$ at $t = 0$, and (iii) γ is constant. This refers to our case.

Now, from Lemma 2 [Dubins and Teicher 1967], $J(\omega)$ is convex and non-decreasing; i.e., there exists a unique number $s > 0$ such that $\omega < J(\omega) < s$ for $\omega < s$ and, for $\omega \geq s$, $J(\omega) = \omega$. Then, the application of the principle of optimality to such function denotes that $J(\omega) \geq \mathbb{E}[\gamma J(\omega+I)]$ for all ω [Dubins and Teicher 1967, Lemma 3]. Hence, the process $J(\omega)$, $\gamma J(\omega + S_1)$, $\gamma^2 J(\omega + S_2)$, ... is a non-negative, expectation decreasing semi-martingale. Through this, $J(\omega) \geq \mathbb{E}[\gamma^t(\omega + S_t)]$ for all stopping times t [Dubins and Teicher 1967, Lemma 4].

For each $z \geq 0$, let $\tau(z)$ be the first stopping time t such that $S_t > z$. For $\omega \leq s$, let $r(t, \omega) = \min(t, \tau(s - \omega))$. Based on Theorem 1 [Dubins and Teicher 1967], the process $\{\gamma^{r(t,\omega)} J(\omega + S_{r(t,\omega)}), t = 0, 1, \ldots\}$ is a uniformly integrable martingale and converges certainly to $\gamma^{\tau(s-\omega)} J(\omega + S_{\tau(s-\omega)})$ since J is continuous and $r(t, \omega)$ converges to $\tau(s - \omega)$. In addition, for each $\omega \leq s$, $\tau(s-\omega)$ is optimal for ω or, equivalently, $J(\omega) = \mathbb{E}[\gamma^{\tau(s-\omega)}(\omega + S_{\tau(s-\omega)})]$ (see also Dubins and Teicher [1967, Theorem 1]).

For $\omega = 0$, we obtain $J(0) = \mathbb{E}[\gamma^{\tau(s)} S_{\tau(s)}] = \mathbb{E}[Y_{\tau(s)}]$, and s is obtained from $s = \mathbb{E}[\gamma^{\tau(0)}(s + S_{\tau(0)})]$ or $s = \mathbb{E}[\gamma^{\tau(0)} S_{\tau(0)}]/(1 - \mathbb{E}[\gamma^{\tau(0)}]$. The determination of the expectation of $\gamma^{\tau(0)}$ and $S_{\tau(0)}$ is as follows: $\tau(0)$ is the minimum t such that $S_t > 0$. Since I is non-negative, thus, one sees that the problem is monotone, then $\tau(0) \equiv 1$ and $S_{\tau(0)} \equiv I_1$. That is because the confidence for stopping s must be the same as the confidence for continuing using the rule that stops the first time the sum of the future observations is positive. Hence, in our case, we obtain the optimal stopping criterion $s = \frac{\gamma}{1-\gamma}\mathbb{E}[I]$; thus, from the one-stage look-ahead optimal rule [Li and Zhang 2005], our detection model stops at the first time t at which $S_t \geq s = \frac{\gamma}{1-\gamma}\mathbb{E}[I]$ and then starts with updating the $\mathbf{w}^*$ prototype.

F. ANALYTICAL EXPRESSION OF $P\{I = 1\}$ USING THE MARCUM Q-FUNCTION.

We have that

$$P(\{I = 1\}) = Q_{\frac{d}{2}}\left(\sqrt{\zeta}, \sqrt{\theta}\right).$$

According to Sun et al. [2010], the $Q_{\kappa_1}(\kappa_2, \kappa_3)$ function can be expressed in infinite series with respect to the lower incomplete Γ function by substituting the $(\kappa_1, \kappa_2, \kappa_3) = (\frac{d}{2}, \sqrt{\zeta}, \sqrt{\theta})$ in our case (37), we obtain

$$Q_{\frac{d}{2}}(\sqrt{\zeta}, \sqrt{\theta}) = e^{-\frac{\zeta}{2}} \sum_{k=0}^{\infty} \frac{\zeta^k}{2^k k!} \frac{\Gamma(k + \frac{d}{2}, \frac{\theta}{2})}{\Gamma(k + \frac{d}{2})},$$

where the lower incomplete function $\Gamma(z, x) = \int_x^\infty e^{-t} t^{z-1} dt$ is defined in Abramowitz [1974, equation (6.5.3)] and Euler function $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt$. Note, the discussed

generalized Marcum Q-function is approximated using the Matlab function `marcumq`$(\kappa_2, \kappa_3, \kappa_1)$ using the algorithm developed in Shnidman [1989].

G. NOMENCLATURE

Table III. Nomenclature

Notation	Explanation
d	Data dimension
$\mathbf{x}$	Multivariate data point in $\mathbb{R}^d$
$\mathcal{B}$	Dataset of points in $\mathbb{R}^d$
$\lvert\mathcal{B}\rvert$	The cardinality (size) of the dataset $\mathcal{B}$
$\mathbf{q} = [\mathbf{x}, \theta]$	Vectorial representation of radius query
θ	Positive scalar
$\mathbf{q} = [a_i, b_i]_{i=1}^d$	Vectorial representation of range query
L_p	p-norm
S_k	k-th data node
$p(\mathbf{x})$	Probability density function of $\mathbf{x}$
y	Answer set cardinality
$\mathcal{E}_0, \mathcal{E}_1, \mathcal{E}_2$	Optimization functions
$\mathcal{T}$	Training set
$\mathcal{E}$	Evaluation set
$\hat{f}_k$	Function estimation model of node S_k
α_k	Learning parameter for model $\mathcal{M}_k$
n	Counter
η	Learning rate
β	Quantization parameter
Θ	Training termination criterion
ϵ	Accuracy threshold
N	Number of query subspaces
M	Number of (query/cardinality) prototypes
$\mathbb{Q}$	Query space ($\mathbb{R}^{2d}$ for range queries and $\mathbb{R}^{d+1}$ for radius queries)
$\mathbf{w} \in \mathbb{Q}$	Query prototype
$\mathbf{w}^* \in \mathbb{Q}$	Winner query prototype
$u \in \mathbb{R}$	Cardinality prototype
$u^* \in \mathbb{R}$	Winner cardinality prototype
$I \in \{0, 1\}$	Indicator function
ρ	Vigilance
γ	Risk factor
t^*	Optimal stopping time
Y	Confidence payoff
B	Number of histogram buckets

REFERENCES

Milton Abramowitz. 1974. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. Dover Publications, Incorporated.

Alexander Alexandrov, Rico Bergmann, Stephan Ewen, Johann-Christoph Freytag, Fabian Hueske, Arvid Heise, Odej Kao, Marcus Leich, Ulf Leser, Volker Markl, Felix Naumann, Mathias Peters, Astrid Rheinlander, Matthias J. Sax, Sebastian Schelter, Mareike Hoger, Kostas Tzoumas, and Daniel Warneke. 2014. The stratosphere platform for big data analytics. *The VLDB Journal* 23, 6 (December 2014), 939–964. DOI:http://dx.doi.org/10.1007/s00778-014-0357-y

Sattam Alsubaiee, Yasser Altowim, Hotham Altwaijry, Alexander Behm, Vinayak Borkar, Yingyi Bu, Michael Carey, Inci Cetindil, Madhusudan Cheelangi, Khurram Faraaz, Eugenia Gabrielova, Raman Grover,

Zachary Heilbron, Young-Seok Kim, Chen Li, Guangqiang Li, Ji Mahn Ok, Nicola Onose, Pouria Pirzadeh, Vassilis Tsotras, Rares Vernica, Jian Wen, and Till Westmann. 2014. AsterixDB: A scalable, open source BDMS. *Proceedings of the VLDB Endowment* 7, 14 (October 2014), 1905–1916. DOI:http://dx.doi.org/10.14778/2733085.2733096

Christos Anagnostopoulos and Peter Triantafillou. 2015a. Learning set cardinality in distance nearest neighbours. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'15)*. 691–696. DOI:http://dx.doi.org/10.1109/ICDM.2015.17

Christos Anagnostopoulos and Peter Triantafillou. 2015b. Learning to accurately COUNT with query-driven predictive analytics. In *Proceedings of the IEEE International Conference on Big Data (Big Data'15)*. 14–23. DOI:http://dx.doi.org/10.1109/BigData.2015.7363736

Atanas Atanasov, Madhusudhanan Srinivasan, and Tobias Weinzierl. 2012. Query-driven parallel exploration of large datasets. In *Proceedings of the IEEE Symposium on Large Data Analysis and Visualization, (LDAV'12)*. 23–30. DOI:http://dx.doi.org/10.1109/LDAV.2012.6378972

Natasha Balac, Tamara B. Sipes, Nicole Wolter, Kenneth Nunes, Robert S. Sinkovits, and Homa Karimabadi. 2013. Large scale predictive analytics for real-time energy management. In *Proceedings of the 2013 IEEE International Conference on Big Data*. 657–664. DOI:http://dx.doi.org/10.1109/BigData.2013.6691635

Stefan Berchtold, Christian Bohm, Daniel A. Keim, and Hans-Peter Kriegel. 1997. A cost model for nearest neighbor search in high-dimensional data space. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'97)*. ACM, New York, NY, 78–86. DOI:http://dx.doi.org/10.1145/263661.263671

Dimitri P. Bertsekas. 2005. *Dynamic Programming and Optimal Control. Volume I*. Athena Scientific, Belmont, MA.

Leon Bottou and Yoshua Bengio. 1994. Convergence properties of the K-means algorithms. In *Advances in Neural Information Processing Systems 7, NIPS Conference, Denver, Colorado, USA*. 585–592.

Olivier Bousquet and Leon Bottou. 2008. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis (Eds.). Curran Associates, Inc., 161–168.

Gail A. Carpenter and Stephen Grossberg. 1988. The ART of adaptive pattern recognition by a self-organizing neural network. *IEEE Computer* 21, 3 (1988), 77–88. DOI:http://dx.doi.org/10.1109/2.33

Kaushik Chakrabarti, Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim. 2000. Approximate query processing using wavelets. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB'00)*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 111–122.

Abon Chaudhuri, Tzu-Hsuan Wei, Teng-Yok Lee, Han-Wei Shen, and Tom Peterka. 2014. Efficient range distribution query for visualizing scientific data. In *Proceedings of the IEEE Pacific Visualization Symposium, (PacificVis'14)*. 201–208. DOI:http://dx.doi.org/10.1109/PacificVis.2014.60

Yuan Shih Chow, Herbert Ellis Robbins, and David Siegmund. 1971. *Great Expectations: The Theory of Optimal Stopping*. Houghton Mifflin, Boston.

Graham Cormode, Minos Garofalakis, Peter J. Haas, and Chris Jermaine. 2012. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases* 4, 1–3 (January 2012), 1–294. DOI:http://dx.doi.org/10.1561/1900000004

D. A. Darling, T. Liggett, and H. M. Taylor. 1972. Optimal stopping for partial sums. *The Annals of Mathematical Statistics* 43, 4 (1972), 1363–1368.

Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified data processing on large clusters. *Commun. ACM* 51, 1 (January 2008), 107–113.

Luke J. Gosink, Christoph Garth, John C. Anderson, E. Wes Bethel, and Kenneth I. Joy. 2011. An application of multivariate statistical analysis for query-driven visualization. *IEEE Transactions on Visualization and Computer Graphics* 17, 3 (2011), 264–275. DOI:http://dx.doi.org/10.1109/TVCG.2010.80

Mihajlo Grbovic and Slobodan Vucetic. 2009. Regression learning vector quantization. In *Proceedings of the 9th IEEE International Conference on Data Mining (ICDM'09)*. 788–793. DOI:http://dx.doi.org/10.1109/ICDM.2009.145

Dimitrios Gunopulos, George Kollios, Vassilis J. Tsotras, and Carlotta Domeniconi. 2005. Selectivity estimators for multidimensional range queries over real attributes. *The VLDB Journal* 14, 2 (2005), 137–154. DOI:http://dx.doi.org/10.1007/s00778-003-0090-4

Gus W. Haggstrom. 1967. Optimal sequential procedures when more than one stop is required. *The Annals of Mathematical Statistics* 38, 6 (1967), 1618–1626.

Elena Ikonomovska, Joao Gama, and Savso Dvzeroski. 2010. Learning model trees from evolving data streams. *Data Mining and Knowledge Discovery* 23, 1 (2010), 128–168.

John E. Dennis Jr. and Robert B. Schnabel. 1983. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice Hall.

Alan F. Karr. 2010. Secure statistical analysis of distributed databases, emphasizing what we don't know. *Journal of Privacy and Confidentiality* 1, 2 (2010), 197–211.

Tuevo Kohonen. 1989. *Self-Organization and Associative Memory: 3rd Edition*. Springer-Verlag New York, Inc., New York, NY.

Teuvo Kohonen. 2013. Essentials of the self-organizing map. *Neural Networks* 37 (January 2013), 52–65.

Jong-Seok Lee, Cheol Hoon Park, and Touradj Ebrahimi. 2013. *Theory and Applications of Hybrid Simulated Annealing*. Springer Berlin Heidelberg, 395–422. DOI:http://dx.doi.org/10.1007/978-3-642-30504-7_16

Lester E. Dubins and Henry Teicher. 1967. Optimal stopping when the future is discounted. *The Annals of Mathematical Statistics* 38, 2 (1967), 601–605.

Shoumei Li and Jinping Zhang. 2005. A general method for convergence theorems of fuzzy set-valued random variables and its applications to martingales and uniform amarts. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 13, 3 (2005), 243–254. DOI:http://dx.doi.org/10.1142/S0218488505003436

Moshe Lichman. 2013. UCI Machine Learning Repository. Retrieved from http://archive.ics.uci.edu/ml.

Chieh-Yen Lin, Cheng-Hao Tsai, Ching-Pei Lee, and Chih-Jen Lin. 2014. Large-scale logistic regression and linear support vector machines using spark. In *Proceedings of the 2014 IEEE International Conference on Big Data, (Big Data'14), Washington, DC, USA, October 27–30*. 519–528. DOI:http://dx.doi.org/10.1109/BigData.2014.7004269

Junshui Ma, James Theiler, and Simon Perkins. 2003. Accurate on-line support vector regression. *Neural Computing* 15, 11 (November 2003), 2683–2703.

James B. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, L. M. Le Cam and J. Neyman (Eds.), Vol. 1. University of California Press, 281–297.

Scott C. Newton, Surya Pemmaraju, and Sunanda Mitra. 1992. Adaptive fuzzy leader clustering of complex data sets in pattern recognition. *IEEE Transactions on Neural Networks* 3, 5 (1992), 794–800. DOI:http://dx.doi.org/10.1109/72.159068

Frank Olken and Doron Rotem. 1990. Random sampling from database files: A survey. In *Proceedings of the 5th International Conference on Statistical and Scientific Database Management (SSDBM'90)*. Springer-Verlag, London, UK, 92–111.

Goran Peskir and Albert Shiryaev. 2006. *Optimal Stopping and Free-Boundary Problems*. Springer, Dordrecht.

Milos Radovanovic, Alexandros Nanopoulos, and Mirjana Ivanovic. 2010. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research* 11 (December 2010), 2487–2531.

Timothy T. Rogers and James L. McClelland. 2014. Parallel distributed processing at 25: Further explorations in the microstructure of cognition. *Cognitive Science* 38 (2014), 1024–1077. DOI:http://dx.doi.org/10.1111/cogs.12148

Kenneth Rose, Eitan Gurewitz, and Geoffrey C. Fox. 1992. Vector quantization by deterministic annealing. *IEEE Transactions on Information Theory* 38, 4 (1992), 1249–1257. DOI:http://dx.doi.org/10.1109/18.144705

David A. Shnidman. 1989. The calculation of the probability of detection and the generalized Marcum Q-function. *IEEE Transactions on Information Theory* 35, 2 (1989), 389–400. DOI:http://dx.doi.org/10.1109/18.32133

Utkarsh Srivastava, Peter J. Haas, Volker Markl, Marcel Kutsch, and Tam Minh Tran. 2006. ISOMER: Consistent histogram construction using query feedback. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE'06, Atlanta, GA, USA), April 3–8*. 39. DOI:http://dx.doi.org/10.1109/ICDE.2006.84

Yin Sun, Árpád Baricz, and Shidong Zhou. 2010. On the monotonicity, log-concavity, and tight bounds of the generalized Marcum and Nuttall Q-functions. *IEEE Transactions on Information Theory* 56, 3 (2010), 1166–1186. DOI:http://dx.doi.org/10.1109/TIT.2009.2039048

Yufei Tao, Christos Faloutsos, and Dimitris Papadias. 2003. The power-method: A comprehensive estimation technique for multi-dimensional queries. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM'03)*. ACM, New York, NY, 83–90.

Hien To, Kuorong Chiang, and Cyrus Shahabi. 2013. Entropy-based histograms for selectivity estimation. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM'13)*. ACM, New York, NY, 1939–1948.

Vladimir N. Vapnik and Alexey Ya. Chervonenkis. 1971. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications* 16, 2 (1971), 264–280.

Vinod Kumar Vavilapalli, Arun C. Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, Bikas Saha, Carlo Curino, Owen O'Malley, Sanjay Radia, Benjamin Reed, and Eric Baldeschwieler. 2013. Apache hadoop YARN: Yet another resource negotiator. In *Proceedings of the 4th Annual Symposium on Cloud Computing (SOCC'13)*. ACM, New York, NY, Article 5, 16 pages.

Raajay Viswanathan, Prateek Jain, Srivatsan Laxman, and Arvind Arasu. 2011. A learning framework for self-tuning histograms. *CoRR* abs/1111.7295 (2011). http://arxiv.org/abs/1111.7295.

Jeffrey S. Vitter. 1985. Random sampling with a reservoir. *ACM Transactions on Mathematical Software* 11, 1 (March 1985), 37–57.

Tom White. 2009. *Hadoop: The Definitive Guide* (1st ed.). O'Reilly Media, Inc.

Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2012. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation (NSDI'12)*. USENIX Association, Berkeley, CA, 2.

Received May 2016; revised December 2016; accepted March 2017