



Large-scale predictive modeling and analytics through regression queries in data management systems

Christos Anagnostopoulos¹ · Peter Triantafillou²

Received: 2 May 2018 / Accepted: 24 November 2018 / Published online: 27 December 2018
© The Author(s) 2018

Abstract

Regression analytics has been the standard approach to modeling the relationship between input and output variables, while recent trends aim to incorporate advanced regression analytics capabilities within data management systems (DMS). Linear regression queries are fundamental to exploratory analytics and predictive modeling. However, computing their exact answers leaves a lot to be desired in terms of efficiency and scalability. We contribute with a novel predictive analytics model and an associated statistical learning methodology, which are efficient, scalable and accurate in discovering piecewise linear dependencies among variables by observing only regression queries and their answers issued to a DMS. We focus on in-DMS piecewise linear regression and specifically in predicting the answers to mean-value aggregate queries, identifying and delivering the piecewise linear dependencies between variables to regression queries and predicting the data dependent variables within specific data subspaces defined by analysts and data scientists. Our goal is to discover a piecewise linear data function approximation over the underlying data only through query–answer pairs that is competitive with the *best* piecewise linear approximation to the ground truth. Our methodology is analyzed, evaluated and compared with exact solution and near-perfect approximations of the underlying relationships among variables achieving orders of magnitude improvement in analytics processing.

Keywords Predictive analytics · Piecewise linear regression learning · Query-driven analytics · Data subspace exploration · Vector regression quantization

1 Introduction

Predictive Modeling and Analytics (PMA) concerns data exploration, model fitting, and regression model learning tasks used in many real-life applications [5,16,22,40,43]. The major goal of PMA is to *explore* and *analyze* multi-dimensional feature vector data spaces [1]. Recently, we have seen a rapid growth of large-scale advanced regression analytics in areas like deep learning for image recognition [22], genome analysis [43] and aggregation analytics [9].

Predictive models like linear regression for prediction and logistic regression for classification are typically desired for exploring data subspaces of a d -dimensional data space of interest in \mathbb{R}^d real-valued space. In in-DMS exploratory analytics and exploratory computing [24], such data subspaces are identified using selection operators over the values of attributes of interest. Within such data subspaces, PMA can provide *local approximation functions or models* focusing mainly on identifying dependencies among features like co-variance estimations and linear regression coefficients. Selection operators include *radius* (a.k.a. *distance near neighbor* (dNN) [7]) queries, which are of high importance in nowadays applications: contextual data stream analytics [4], aggregate predictive analytics over DMS [6], edge computing analytics over data streams in Internet of Things environments [31], location-based predictive analytics [3], searching for statistical correlations of spatially close objects (within a radius), measuring multivariate skewness [36], spatial analytics [38] focusing on the construction of semi-variograms in a specific geographical region [55,56]

✉ Christos Anagnostopoulos
christos.anagnostopoulos@glasgow.ac.uk
Peter Triantafillou
p.triantafillou@warwick.ac.uk

¹ School of Computing Science, University of Glasgow,
Glasgow G12 8QQ, UK

² Department of Computer Science, University of Warwick,
Coventry CV4 7AL, UK

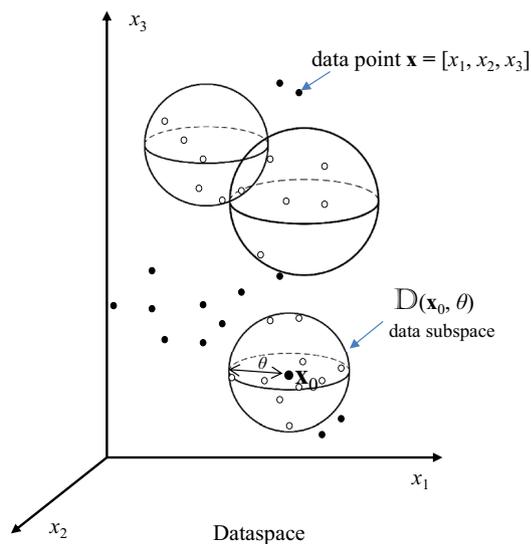


Fig. 1 The distance near neighbors (dNN) queries define data subspaces of interest $\mathbb{D}(\mathbf{x}_0, \theta)$ over the three-dimensional data space $(x_1, x_2, x_3) \in \mathbb{R}^3$

earth analytics monitoring regions of interest from sensors' acoustic signals, and environmental monitoring for chemical compounds correlation analysis given a geographical area.

The interactive predictive analytics process conducted by data science analytics, engineers, and statisticians is as follows [24,41]: Analysts and data scientists interact with in-DMS analytics tools by issuing selection queries (i.e., dNN queries) to define real-valued data subspaces $\mathbb{D} \subset \mathbb{R}^d$ in a d -dimensional data space of interest for exploration and analysis. Then, the local dependencies among the features (dimensions) in those subspaces are extracted and certain regression models are evaluated for their goodness of fit over those data subspaces \mathbb{D} , i.e., by identifying the statistical model that is most likely to have generated those data in \mathbb{D} . For concreteness, we focus on defining data subspaces of interest $\mathbb{D}(\mathbf{x}_0, \theta)$ using a dNN query, notated by Q , as the convex subset of d -dim. data points (row vectors) $\mathbf{x} = [x_1, \dots, x_d] \in \mathbb{R}^d$ lying within a hypersphere (ball) with center \mathbf{x}_0 and scalar radius θ , i.e., $\mathbb{D}(\mathbf{x}_0, \theta)$ contains all $\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x} - \mathbf{x}_0\|_2 \leq \theta$, where $\|\mathbf{x}\|_2$ is the Euclidean norm; for an illustration, see Fig. 1.

A major challenge in PMA is to model and learn the very local statistical information of analysts' interested data subspaces, e.g., local regression coefficients and local data approximation functions, and then extrapolate such knowledge to predict such information for unexplored data subspaces [53]. Based on this abstraction of PMA, which is massively applied on the above-mentioned real-life applications, we focus on two important predictive analytics queries for in-DMS analytics: mean-value queries and linear regression queries.

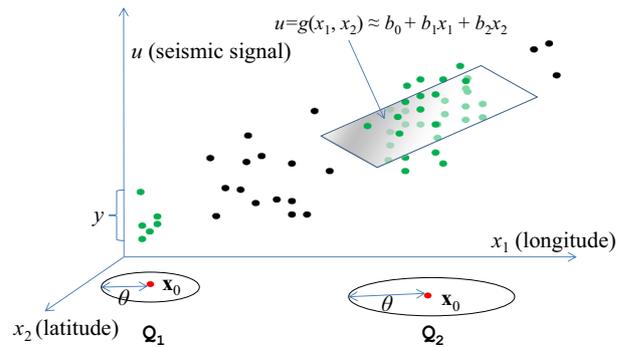


Fig. 2 Mean-value Q_1 and linear regression Q_2 queries over the data space $(u, x_1, x_2) \in \mathbb{R}^3$

Example 1 Consider the running example in Fig. 2. Seismologists issue a mean-value query Q_1 over a 3-dim. space $(u, x_1, x_2) \in \mathbb{R}^3$, which returns the mean value y of the feature u (seismic signal; P-wave speed) of those spatial points $(x_1, x_2) \in \mathbb{D}(\mathbf{x}_0, \theta) \subset \mathbb{R}^2$ projections (referring to surface longitude and latitude) within a disk of center \mathbf{x}_0 and radius θ . The query Q_1 is central to PMA because the average y is always used as a linear sufficient statistic for the data subspace \mathbb{D} , and it is the best linear predictor of the seismic signal output u based on the region identified around the center point $(x_1, x_2) \in \mathbb{D}(\mathbf{x}_0, \theta)$ [32].

A linear regression query Q_2 calculates the coefficients of a linear regression function within a defined data subspace. For example, in Fig. 2, consider geophysicists issuing queries Q_2 over a 3-dim. space $(u, x_1, x_2) \in \mathbb{R}^3$, which returns the seismic primary-wave (P-wave) velocity u -intercept (b_0) and the coefficients b_1 and b_2 for x_1 (longitude) and x_2 (latitude), where the $\mathbf{x} = [x_1, x_2]$ points belong to a subspace $\mathbb{D}(\mathbf{x}_0, \theta) \in \mathbb{R}^2$. By estimating the linear coefficients, e.g., the parameter row vector $\mathbf{b} = [b_0, b_1, b_2]$, we can then interpret the relationships among the features \mathbf{x} and u and assess the statistical significance of each feature of \mathbf{x} within $\mathbb{D}(\mathbf{x}_0, \theta)$. The output of the Q_2 query refers to the dependency of u with \mathbf{x} , which in our example is approximated by a 2-dim. plane $u \approx b_0 + b_1x_1 + b_2x_2$, and quantifies how well the local linear model fits the data.

Query Q_2 is important in PMA because it supports model fitting through, e.g., piecewise linear regression (PLR) [10], and provides confidence whether linear models fit well or not the underlying data. To better illustrate Q_1 and Q_2 queries, consider their corresponding SQL syntax. The mean-value query Q_1 over data subspace $\mathbb{D}(\mathbf{x}_0, \theta)$ for the example data space (u, x_1, x_2) shown in Fig. 2 (relation $R(u, x_1, x_2)$) is represented by a disk of center $\mathbf{x}_0 = [x_{0(1)}, x_{0(2)}]$ and radius θ :

```

Q1: SELECT avg(R.u) as y
    FROM R
    WHERE SQRT((R.x1 - x0(1)) * (R.x1 - x0(1)) +
              (R.x2 - x0(2)) * (R.x2 - x0(2))) <=  $\theta$ ,
  where SQRT( $x$ ) is the square root of real number  $x$ .

```

Consider now the regression query Q2 over subspace $\mathbb{D}(\mathbf{x}_0, \theta)$ for the example data space (u, x_1, x_2) in Fig. 2. Based on the XLERATORDB/statistics LINEST function in SQL Server 2008 syntax, Q2 first defines the subspace $\mathbb{D}(\mathbf{x}_0, \theta)$, then it stores the corresponding tuples (u, x_1, x_2) temporarily to a relation $S(u, x_1, x_2)$, i.e., $(x_1, x_2) \in \mathbb{D}(\mathbf{x}_0, \theta)$, and, finally, invokes the multivariate linear regression function LINEST over relation S :

```

Q2: SELECT u, x1, x2 INTO S(u, x1, x2)
    FROM R
    WHERE SQRT((R.x1 - x0(1)) * (R.x1 - x0(1)) +
              (R.x2 - x0(2)) * (R.x2 - x0(2))) <=  $\theta$ 
  SELECT *
  FROM package.LINEST('S', '**', '',
                     NULL, 1, 'False')

```

The result is the intercept b_0 and regression coefficients $\mathbf{b} = [b_1, b_2]$.

To evaluate queries Q1 and Q2, the system must access the data to establish the data subspace $\mathbb{D}(\mathbf{x}_0, \theta)$, and then take the average value of u in that subspace for query Q1 (e.g., the average seismic signal speed in San Andreas, CA, region) and invoke a multivariate linear regression algorithm [32] for Q2. The Q1 and Q2 type queries are provided by all modern PMA systems like Spark analytics [47], MATLAB¹ and DMS systems, e.g., XLERATORDB² of Microsoft SQL Server³ and Oracle UTL_NLA.⁴

Remark 1 Please refer to Table 2 in “Appendix” for a table of notations and symbols used in this paper.

1.1 Desiderata

We focus on in-DMS analytics with PMA using models and algorithms for the query types Q1 and Q2. The aim is to meet the following desiderata, providing answers to the following questions:

- D1 Are there linear dependencies among dimensions in unexplored data subspaces, and which are such subspaces?
- D2 If there are data subspaces, where linear approximations fit well with high confidence, can the system provide

these yet unknown linear regression models efficiently and scalably to the analysts?

- D3 If in some subspaces linear approximations do not fit well w.r.t. analysts needs, can the system provide fitting models through piecewise local linear approximations?
- D4 A solution must meet *scalability*, *efficiency*, and *accuracy* desiderata as well.

Concerning desideratum D1 We study the regression problem—a fundamental inference task that has received tremendous attentions in data mining, data exploration, predictive modeling, machine and statistical learning during the past fifty years. In a regression problem, we are given a set of n observations of (\mathbf{x}_i, u_i) , where u_i 's are the dependent variables (outputs) and the \mathbf{x}_i 's are the independent variables (inputs); for instance, refer to the 3-dim. input–output points (\mathbf{x}, u) with input $\mathbf{x} = [x_1, x_2]$ and output u shown in Fig. 2 in our Example 1.

We desire to model the relationship between inputs and outputs. The typical assumption is that there exists an unknown data function g that approximately models the underlying relationship and that the dependent observations are corrupted by random noise. Specifically, we assume that there exists a family of functions \mathcal{L} such that for some data function $g \in \mathcal{L}$ it holds true the generative model:

$$u_i = g(\mathbf{x}_i) + \epsilon_i, \quad (1)$$

where ϵ_i 's are independent and identically distributed (i.i.d.) random variables drawn from a distribution, e.g., Gaussian.

Let us now move a step further to provide more information about the modeled relationship function g in (1). The derivation of several *local* linear approximations, as opposed to a single linear approximation over the *whole* data space, can provide more accurate and significant insights. The key issue to note here is that a global (single) linear approximation of g interpolating among all items of the whole data space \mathbb{D} leaves, in general, much to be desired: The analysts presented with a single global linear approximation might have an inaccurate view due to missing ‘local’ statistical dependencies within *unknown local data subspaces that comprise* \mathbb{D} . This will surely lead to prediction errors and approximation inaccuracies when issuing queries Q1 and Q2 to the DMS.

Example 2 Consider the input–output in a (u, x) 2-dim. space in Fig. 3(upper) and the actual data function $u = g(x)$ (in red). A Q2 query issued over the data subspace $\mathbb{D}(x_0, \theta)$ will calculate the intercept b_0 and slope b_1 of the linear approximation $u \approx \hat{g}(x) = b_0 + b_1x$ (the green line l) over those $x \in \mathbb{D}(x_0, \theta)$. Evidently, such a line shows a very coarse and unrepresentative dependency between output u and input x , since u and x do not linearly depend on each other within

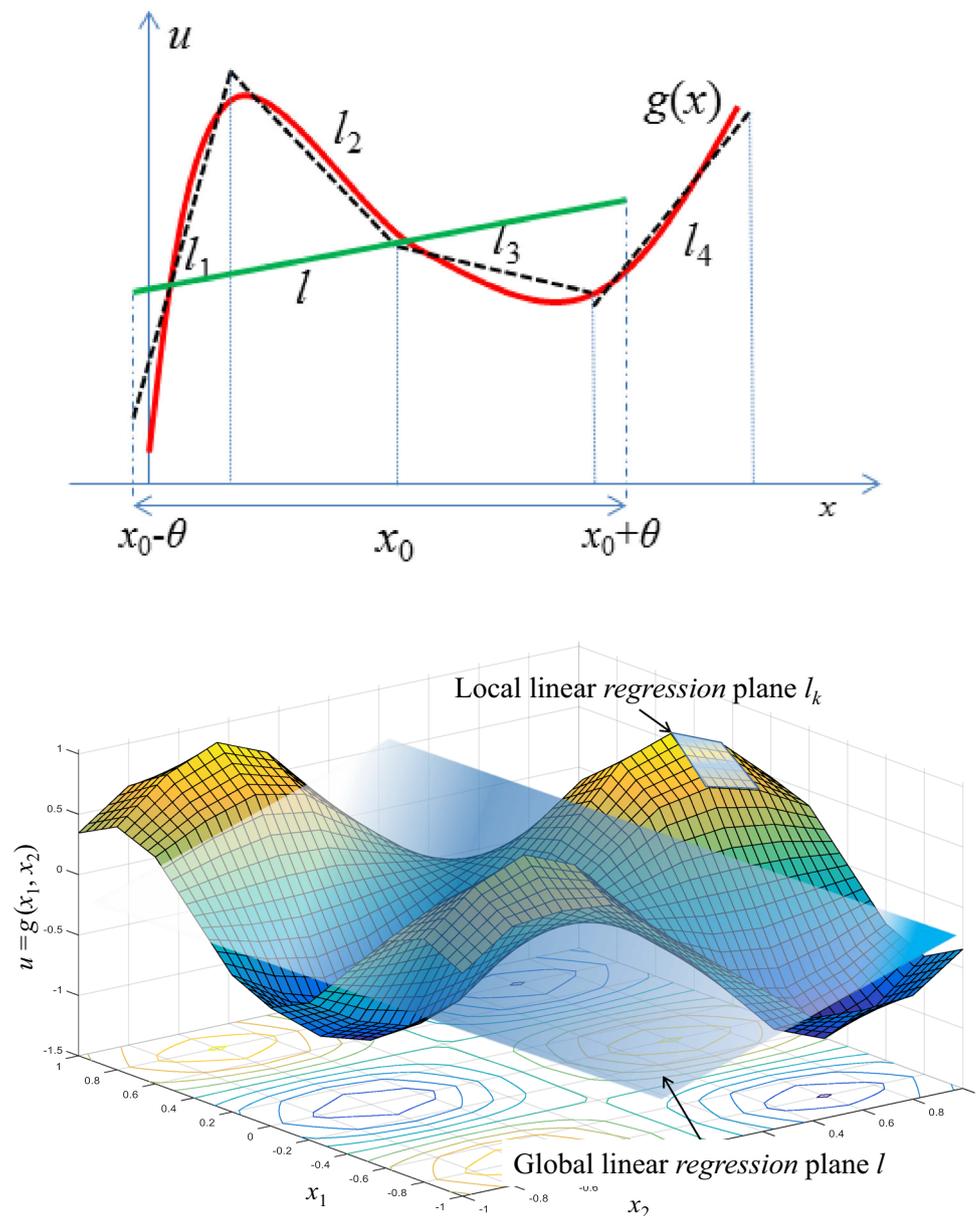
¹ <https://www.mathworks.com>.

² <http://www.westclintech.com>.

³ <https://msdn.microsoft.com/en-us/library/cc280445.aspx>.

⁴ https://docs.oracle.com/cd/B19306_01/appdev.102/b14258/u_nla.htm.

Fig. 3 (Upper) Nonlinearity of the data function $u = g(x)$, global linear l and local linear l_1, \dots, l_4 approximations of $g(x) : |x - x_0| \leq \theta$; (lower) Nonlinearity of the bivariate data function $u = g(x_1, x_2)$, multiple local linear regression planes (PLR segments) l_k , and a global linear regression plane l



the entire data subspace $\mathbb{D}(x_0, \theta)$. The point is that we should obtain a finer grained and more accurate dependency between output u and input x . The *principle of local linearity* [26] states that linear approximations of the underlying data function in *certain* data subspaces fit the global nonlinearity better in the entire data subspace of interest. In Fig. 3(upper), we observe four local linear approximations l_1, \dots, l_4 in the data subspace. Therefore, it would be preferable if, as a result of query Q2, the analysts were provided with a list of the local line segments $\mathcal{S} = \{l_1, \dots, l_4\}$, a.k.a. piecewise linear regression. These ‘local’ segments better approximate the linearity of output u . Moreover, in Fig. 3(lower) the underlying data function $u = g(x_1, x_2)$ in the 3-dim. data space does not exhibit linearity over the entire (x_1, x_2) plane. We

can observe how the global linear relationship $\hat{g}(x_1, x_2)$ cannot capture the very local statistical dependencies between $\mathbf{x} = [x_1, x_2]$ and u , which are better captured in certain data subspaces by certain local line segments $\hat{g}_k(x_1, x_2)$.

Concerning desiderata D2 and D3 Consider the notion of the mean squared error (MSE) [26] to measure the performance of an estimator. Given the n samples (\mathbf{x}_i, u_i) and following the generative model in (1) having mean value $\mathbb{E}[\epsilon_i] = 0$ and variance $\mathbb{E}[\epsilon_i^2] = \sigma^2$, our goal is to estimate a data function \hat{g} that is close to the true, unknown data function g with high probability over the noise terms ϵ_i . We measure the distance between our estimate \hat{g} and the unknown function g with the MSE:

$$\text{MSE}(\hat{g}) = \frac{1}{n} \sum_{i=1}^n (g(\mathbf{x}_i) - \hat{g}(\mathbf{x}_i))^2. \tag{2}$$

In the general case, the data function g is nonlinear and satisfies some well-defined structural constraints. This has been extensively studied in a variety of contexts [11,17,37]. In our desiderata D2 and D3, we focus on the case that the data function g is nonlinear but can be approximated by a piecewise linear function through an *unknown* number K of *unknown* pieces (line segments). We then provide the following definition of this type of data function:

Definition 1 The data function $g : \mathbb{R}^d \rightarrow \mathbb{R}$ is a K -piecewise linear function if there exists a partition of the input data space $\mathbb{D} \subset \mathbb{R}^d$ into K disjoint subspaces $\mathbb{D}_1, \dots, \mathbb{D}_K$ with corresponding linear regression parameters $\mathbf{b}_{X,1}, \dots, \mathbf{b}_{X,K} \in \mathbb{R}^d$ such that for all $\mathbf{x} = [x_1, \dots, x_d] \in \mathbb{R}^d$ we have that $u(\mathbf{x}) = \mathbf{b}_{X,k} \cdot \mathbf{x}^\top$, if $\mathbf{x} \in \mathbb{D}_k$.

The case where K is fixed (given) has received considerable attention in the research community [54]. The special case of piecewise polynomial functions (splines) has been also used in the context of inference including density estimation, histograms, and regression [39].

Let us now denote with \mathcal{L}_K the space of K -piecewise linear functions. While the ground truth may be close to a piecewise linear function, even in certain subspaces, generally we do not assume that it exactly follows a piecewise linear function (yet unknown). In this case, our goal is to recover a piecewise linear function that is competitive with the *best* piecewise linear approximation to the ground truth.

Formally, let us define the following problem, where we assume that the generative model in (1) representing the data function g is any arbitrary function. We define:

$$\text{OPT}_K = \min_{g' \in \mathcal{L}_K} \text{MSE}(g') \tag{3}$$

to be the error of the best fit K -piecewise linear function to g and let g^* be any K -piecewise linear function that achieves this minimum. Then, the central goal of desiderata D2 and D3 is to discover g^* , which achieves a MSE as close to OPT_K as possible, provided that we observe only queries and their answers and not having access to the input–output actual pairs (\mathbf{x}_i, u_i) .

Remark 2 If the segments of the data function g were known a priori, the segmented regression problem could have been immediately reduced to K independent linear regression problems. In the general case, where the location of the segment boundaries and their corresponding coefficients are unknown, one needs to discover them using information provided only by the observations of input–output pairs (\mathbf{x}_i, u_i) . To address this problem, previous works [13,54] while being

statistically efficient are computationally slow and prohibited for large-scale data sets, i.e., the running time for a given data subspace scales at least quadratically with the size of data points n in the queried data subspace, thus, being impractical for large data subspaces or even worse for the entire data space.

In our context, however, the analysts explore the data space *only* by issuing queries over specific data subspaces, thus, observing only the answers of the analytics queries. Specifically, the analysts do not know before issuing a query how the data function behaves within an ad hoc defined data subspace $\mathbb{D}(\mathbf{x}_0, \theta)$. When a query Q2 is issued, it is not known whether the data function g behaves with the same linearity throughout the entire $\mathbb{D}(\mathbf{x}_0, \theta)$ or not, and within which subspaces, if any, g changes its trend and u and \mathbf{x} exhibit linear dependencies. Thus, the desiderata D2 and D3 focus on learning the boundaries of these local subspaces within $\mathbb{D}(\mathbf{x}_0, \theta)$ and within each local subspace, discovering the linear dependency (segment) between output u and input \mathbf{x} . This would arm analysts and data scientists with much more accurate knowledge on how the data function $g(\mathbf{x})$ behaves within a given data subspace $\mathbb{D}(\mathbf{x}_0, \theta)$. Hence, decisions on further data exploration w.r.t. complex model selection and/or validation can be taken by the analysts.

Concerning desideratum D4 Our motivation comes from the availability of the past issued and executed queries over large-scale datasets. In the words of [34]: ‘As data grows, it may be beneficial to consider faster inferential algorithms, because the increasing statistical strength of the data can compensate for the poor algorithmic quality’, it seems to be advantageous to sacrifice statistical prediction accuracy in order to achieve faster running times because we can then achieve the desired error guarantee faster [35].

Our motivation rests on learning and predicting how the data function g behaves differently in certain data subspaces, within a greater data subspace defined by past issued and executed queries. The state-of-the-art methods leave a lot to be desired in terms of efficiency and scalability. Our key insight and contribution in this direction lies in the development of statistical learning models which can deliver the above functionality for queries Q1 and Q2 in a way that is highly accurate and insensitive to the sizes of the underlying data spaces in number of data points, and thus scalable. The essence of the novel idea we put forward rests on *exploiting previously executed queries and their answers* obtained from the DMS/PMA system to train a model and then use that model to:

- *approximate* the underlying data function g over the analysts’ queried data subspaces by estimating the unknown K segments and their unknown local model coefficients/PLR segments. This has to be achieved based *only*

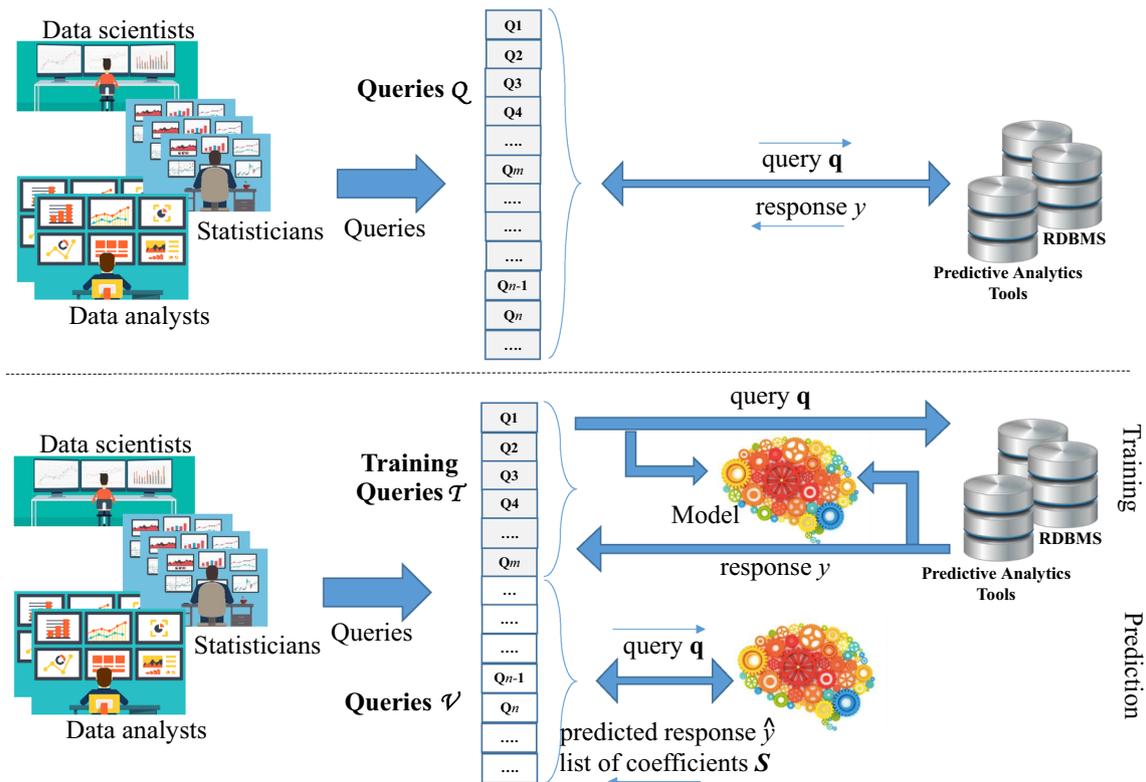


Fig. 4 The System Context: (Upper) User community queries \mathcal{Q} issued and executed without our model; (Lower) our model learns from past queries \mathcal{T} and predicts future query results \mathcal{V}

on the issued queries and their answers, where no data access is provided to analysts by the DMS;

- *predict* the list \mathcal{S} of the linear models (segments) that model the PLR estimator data function \hat{g} minimizing the MSE in (3). Such models *best* explain (fit) the underlying data function g over a given data subspace $\mathbb{D}(\mathbf{x}_0, \theta)$;
- *predict* the answer y of any unseen mean-value query over a data subspace $\mathbb{D}(\mathbf{x}_0, \theta)$;
- *predict* the output data value \hat{u} given an unseen input \mathbf{x} based on the approximate data function \hat{g} .

Remark 3 In the prediction phase, that is, after training, no access to the underlying data systems is required, thus, ensuring desideratum D4.

1.2 Challenges and organization

In Fig. 4, we show the system context within which our rationale and contributions unfold. A DMS serves analytics queries from a large user community. Over the time, all users (data scientists, statisticians, analysts, applications) will have issued a large number of queries ($\mathcal{Q} = \{Q_1, Q_2, \dots, Q_n\}$), and the system will have produced responses (e.g., y_1, y_2, \dots, y_n for Q_1 queries). Our key idea is to inject a novel statistical learning model and novel query processing algorithms

in between users and the DMS that monitors queries and responses and learns to associate a query with its response. After training, say after the first $m < n$ queries $\mathcal{T} = \{Q_1, \dots, Q_m\}$ then, for any *new/unseen* query Q_t with $m < t \leq n$, i.e., $Q_t \in \mathcal{V} = \mathcal{Q} \setminus \mathcal{T} = \{Q_{m+1}, \dots, Q_n\}$, our model approximates the data function g with an estimator function \hat{g} through a list \mathcal{S} of local linear regression coefficients (line segments) that best fits the actual and unknown function g given the query Q_t 's data subspace and predicts its response \hat{y}_t *without accessing the DMS*. The efficiency and scalability benefits of our approach are evident. Computing the exact answers to queries Q_1 and Q_2 can be very time-consuming, especially for large data subspaces. So if this model and algorithms can deliver highly accurate answers, query processing times will be dramatically reduced. Scalability is also ensured for two reasons. Firstly, in the *data dimension*, as query Q_1 and Q_2 executions (after training) do not involve data accesses, even dramatic increases in DB size do not impact query execution. Secondly, in the *query-throughput dimension*, avoiding DMS internal resource utilization (that would be required if all Q_1 and Q_2 queries were executed over the DMS data) saves resources that can be devoted to support larger numbers of queries at any given point in time. Viewed from another angle, our contributions aim to exploit all the work performed by the DMS engine when answering previous queries, in order

to facilitate accurate answers to future queries efficiently and scalably.

The research challenge of this rationale is the problem of non-fixed designed segmented regression exploiting only queries and answers by not accessing the underlying data anymore. Specifically, the challenges are:

- Identify the number and boundaries of the data subspaces with local linearities and deliver the local linear approximations for each subspace identified, i.e., predict the list S for an unseen query $Q2$, thus, no need to execute the query $Q2$. Clearly, this challenge copes further with the following problems: the boundaries of these data subspaces are unknown and cannot be determined even if we could scan all of the data, which in any case would be inefficient and less scalable.
- Predict the average value of answer y for an unseen query $Q1$, thus, no need to execute the query $Q1$.

It is worth noting that these cannot be achieved solely by accessing the data, as we need information on which are the users' ad hoc defined subspaces of interest. It is possible to provide this information a priori for *all* possible data subspaces of interest to analysts, i.e., consider all possible center points \mathbf{x}_0 and all possible radii θ values. However, this is clearly impractical—this knowledge is obtained *after* the analysts have issued queries over the data, thus, reflecting their subspaces of interest and exploration.

The paper is organized as follows: Sect. 2 reports on the related work and provides our major contribution of this work. In Sect. 3, we formulate our problems and provide preliminaries, while Sect. 4 provides our novel statistical learning algorithms for large-scale predictive modeling. Section 5 introduces our proposed query-driven methodologies, corresponding algorithms and analyses, while in Sect. 6, we report on the piecewise linear approximation and query-answer prediction methods. The convergence analysis and the inherent computational complexity are elaborated in Sect. 7, and we provide a comprehensive performance evaluation and comparative assessment of our methodology in Sect. 8. Finally, Sect. 9 summarizes our work and discusses our future research agenda in the direction of query-driven predictive modeling.

2 Related work and contribution

2.1 Related work

Out-with DMS environments, statistical packages like MATLAB and R⁵ support fitting regression functions. However,

⁵ <https://www.r-project.org/>.

their algorithms for doing so are inefficient and hardly scalable. Moreover, they lack support for relational and declarative Q1 and Q2 queries. So, if data are already in a DMS, they would need to be moved back and forth between external analytics environments and the DMS, resulting in considerable inconveniences and performance overheads, (if at all possible for big datasets). At any rate, modern DMSs should provide analysts with rich support for PMA.

An increasing number of major database vendors include in their products data mining and machine learning analytic tools. PostgreSQL, MySQL, MADLib (over PostgreSQL) [21] and commercial tools like Oracle Data Miner, IBM Intelligent Miner and Microsoft SQL Server Data Mining provide SQL-like interfaces for analysts to specify regression tasks. Academic efforts include MauveDB [23], which integrates regression models into a DMS, while similarly FunctionDB [50] allows analysts to directly pose regression queries against a DMS. Also, BISMARCK [27] integrates and supports in-DMS analytics, while [48] integrates and supports least squares regression models over training datasets defined by arbitrary join queries on database tables. All such works that also support Q1 and Q2 queries can serve as the DMS within Fig. 4. However, in the big data era exact Q1, Q2 computations leave much to be desired in efficiency and scalability, as the system must first execute the selection, establishing the data subspaces per query, and then access all tuples in Q1, Q2.

Apart from the *standard* multivariate linear regression algorithm, i.e., adopting ordinary least squares (OLS) for function approximation [29], related literature contains more elaborate piecewise regression algorithms, e.g., [10,12,18,28], which can actually detect the nonlinearity of a data function g and provide multiple local linear approximations. Given an ad hoc exploration query over n points in a d -dim. space, the standard OLS regression algorithm has asymptotic computational complexity of $O(n^2d^5)$ and $O(nd^2 + d^3)$, respectively [32]. Therefore, OLS algorithms suffer from poor scalability and efficiency—especially as n is getting larger, and/or in high-dimensional spaces, as will be quantified in Sect. 8. Such methodologies are suffering such overheads for *every* query issued, which is highly undesirable. To address this, one may think to perform data-space analyses *only once*, seeking to derive *all* local linear regression models for the *whole of the data space*, and use the derived models for all queries. Indeed, a literature survey reveals several methods like [12,42], which identify the nonlinearity of data function g and provide multiple local linear approximations. Unfortunately, these methods are very computationally expensive and thus do not scale with the size n of the data points. All these methods execute queries like query Q2, going through a series of stages: partitioning the entire data space into clusters, assigning each data point to one of these clusters, and fitting a linear regression function

to each of the clusters. However, data clustering cannot automatically guarantee that the within-cluster nonlinearity of the data function g is captured by a local linear fit. Hence, all these methods are iterative, repeating the above stages until convergence to minimize the residuals estimation error of *all* approximated local linear regression functions. For instance, the method in [10] clusters and regresses the entire data space against K clusters with a complexity of $O(K(n^2d + nd^2))$. Similarly, the incremental adaptive controller method [20] using self-organizing structures requires $O(n^2dT)$ for training purposes. The same holds for the methods [12,19,20,28] that combine iterative clustering and classification for piecewise regression requiring also $O(n^2dT)$. Linear regression methods indicate their high costs when computing exact answers. As all these methods derive regression models over the whole data space, e.g., over trillions of points, the scalability and efficiency desiderata are missed, as Sect. 8 will showcase.

This paper significantly extends our previous work [8] for scalable regression queries in the dimensions of mathematical analyses, fundamental theorems and proofs for vector quantization and piecewise multivariate linear regression (Sects. 5 and 6), theoretical analyses and proofs of the PLR data approximation and prediction error bounds (Sect. 5), analysis of the model convergence, variants of partial and global convergence of PLR data approximation, query answer prediction (Sects. 7 and 7.2), comprehensive sensitivity analysis, and comparative assessment of the proposed methodology (Sect. 8).

Our approach accurately supports *predicting* the result of mean-value Q1 queries, *approximating* the underlying data function g based on (multiple) local linear models of regression Q2 queries, and *predicting* the output data values given unseen inputs by estimating the underlying data function. It does so while achieving high prediction accuracy and goodness of fit, after training without executing Q1 and Q2, thus, without accessing data. This ensures a highly efficient and scalable solution, which is *independent* of the data sizes, as Sect. 8 will show.

2.2 Contribution

The contribution of this work lies in efficient and scalable models and algorithms to obtain highly accurate results for mean-value and linear regression queries and PLR-based data function approximation. This rests on learning the principal local linear approximations of data function g . Our approach is query-driven, where past issued queries are exploited to partition the queried data space into subspaces in such a way of minimizing the induced regression error and the model fitting/approximation error. In each data subspace, we incrementally approximate the data function g based on a novel PLR approximation methodology only via query–answer

pairs. Given a query over a data subspace $\mathbb{D}(\mathbf{x}_0, \theta)$, we contribute how to:

- deliver a PLR-based data approximation of the data function g over different unseen subspaces that best explain the underlying function g within $\mathbb{D}(\mathbf{x}_0, \theta)$,
- predict the data output value $\hat{u} \approx g(\mathbf{x})$ for each unseen data input $\mathbf{x} \in \mathbb{D}(\mathbf{x}_0, \theta)$,
- predict the average value y of the data output $u = g(\mathbf{x})$ with $\mathbf{x} \in \mathbb{D}(\mathbf{x}_0, \theta)$.

The research outcome of this work is:

- A statistical learning methodology for query-driven PLR approximations of data functions over multi-dimensional data spaces. This methodology *indirectly* extracts information about the unknown data function g only by observing and learning the mapping between aggregation queries and their answers.
- A joint optimization algorithm for minimizing the PLR data approximation error and answer prediction error in light of quantizing the query space.
- Convergence analyses of the methodology including variants for supporting *partial* and *global* convergence.
- Mathematical analyses of the query-driven PLR approximation and prediction error bounds;
- Mean-value and data-value prediction algorithms for unseen Q1 and Q2 queries.
- A PLR data approximation algorithm over data subspaces defined by unseen Q2 queries.
- Sensitivity analysis and comparative performance assessment with PLR and multivariate linear regression algorithms found in the literature in terms of scalability, prediction accuracy, data value prediction error and goodness of fit of PLR data approximation.

3 Problem analysis

3.1 Definitions

Let $\mathbf{x} = [x_1, \dots, x_d] \in \mathbb{R}^d$ denote a multivariate random data *input* row vector, and $u \in \mathbb{R}$ a univariate random *output* variable, with (unknown) joint probability distribution $P(u, \mathbf{x})$. We notate $g : \mathbb{R}^d \rightarrow \mathbb{R}$ with $\mathbf{x} \mapsto u$ the unknown underlying data function from input \mathbf{x} to output $u = g(\mathbf{x})$.

Definition 2 The linear regression function of input $\mathbf{x} \in \mathbb{R}^d$ onto output $u \in \mathbb{R}$ is: $u = b_0 + \sum_{i=1}^d b_i x_i + \epsilon = b_0 + \mathbf{b}\mathbf{x}^\top + \epsilon$, where: ϵ is a random error with mean $\mathbb{E}[\epsilon] = 0$ and variance $\text{Var}(\epsilon) = \sigma^2 > 0$, $\mathbf{b} = [b_1, \dots, b_d]$ is the slope row vector of real coefficients and b_0 is the intercept.

Definition 3 The p -norm (L_p) distance between two input vectors \mathbf{x} and \mathbf{x}' from \mathbb{R}^d for $1 \leq p < \infty$, is $\|\mathbf{x} - \mathbf{x}'\|_p = (\sum_{i=1}^d |x_i - x'_i|^p)^{\frac{1}{p}}$ and for $p = \infty$, is $\|\mathbf{x} - \mathbf{x}'\|_\infty = \max_{i=1, \dots, d} \{|x_i - x'_i|\}$.

Consider a scalar $\theta > 0$, hereinafter referred to as *radius*, and a dataset \mathcal{B} consisting of n input–output pairs $(\mathbf{x}_i, u_i) \in \mathcal{B}$.

Definition 4 Given input $\mathbf{x} \in \mathbb{R}^d$ and radius θ , a data subspace $\mathbb{D}(\mathbf{x}, \theta)$ is the convex data subspace of \mathbb{R}^d , which includes input vectors $\mathbf{x}_i : \|\mathbf{x}_i - \mathbf{x}\|_p \leq \theta$ with $(\mathbf{x}_i, u_i) \in \mathcal{B}$.

Definition 5 Given an input vector $\mathbf{x} \in \mathbb{R}^d$ and radius θ , the mean-value $\mathcal{Q}1$ query over a dataset \mathcal{B} returns the average of the outputs $u_i = g(\mathbf{x}_i)$, whose corresponding input vectors $\mathbf{x}_i \in \mathbb{D}(\mathbf{x}, \theta)$, i.e.,

$$y = \frac{1}{n_\theta(\mathbf{x})} \sum_{i \in [n_\theta(\mathbf{x})]} u_i : \|\mathbf{x}_i - \mathbf{x}\|_p \leq \theta, \tag{4}$$

where $n_\theta(\mathbf{x})$ is the cardinality of the set $\{\mathbf{x}_i : \|\mathbf{x}_i - \mathbf{x}\|_p \leq \theta\}$ and $(\mathbf{x}_i, u_i) \in \mathcal{B}$. We represent a query as the $(d + 1)$ -dim. row vector $\mathbf{q} = [\mathbf{x}, \theta] \in \mathbb{Q} \subset \mathbb{R}^{d+1}$. The $(d + 1)$ -dim. space \mathbb{Q} is referred to as query vectorial space. We adopt the compact notation $i \in [n]$ as for $i = 1, \dots, n$.

Definition 6 The L_2^2 distance or similarity measure between queries $\mathbf{q}, \mathbf{q}' \in \mathbb{Q}$ is $\|\mathbf{q} - \mathbf{q}'\|_2^2 = \|\mathbf{x} - \mathbf{x}'\|_2^2 + (\theta - \theta')^2$.

Definition 7 The queries \mathbf{q}, \mathbf{q}' , which define the subspaces $\mathbb{D}(\mathbf{x}, \theta)$ and $\mathbb{D}(\mathbf{x}', \theta')$, respectively, overlap if for the boolean indicator $\mathcal{A}(\mathbf{q}, \mathbf{q}') \in \{\text{TRUE}, \text{FALSE}\}$ holds true that: $\mathcal{A}(\mathbf{q}, \mathbf{q}') = (\|\mathbf{x} - \mathbf{x}'\|_2 \leq \theta + \theta') = \text{TRUE}$.

A query $\mathbf{q} = [\mathbf{x}, \theta]$ defines a data subspace $\mathbb{D}(\mathbf{x}, \theta)$ w.r.t. dataset \mathcal{B} .

3.2 Problem formulation

Formally, our challenges are:

- *CH1*: predict the aggregate output outcome \hat{y} of a random query $\mathbf{q} = [\mathbf{x}, \theta]$. Given an unknown query function $f : \mathbb{Q} \subset \mathbb{R}^{d+1} \rightarrow \mathbb{R}$, which maps a query $\mathbf{q} = [\mathbf{x}, \theta] \mapsto y$, we seek a **query-PLR** estimate $\hat{f} \in \mathcal{L}_K$ to predict the actual answer $y = f(\mathbf{q}) = f(\mathbf{x}, \theta)$ ⁶ for an unseen query \mathbf{q} , i.e., $\hat{y} = \hat{f}(\mathbf{q}) = \hat{f}(\mathbf{x}, \theta)$. The challenge is:

$$\hat{f} = \arg \min_{f \in \mathcal{L}_K} \text{MSE}(f).$$

⁶ We deliberately proceed with abuse of notation $f(\mathbf{q}) = f(\mathbf{x}, \theta)$ to provide detailed information of the input arguments of the query-PLR function f .

- *CH2*: identify the local linear approximations of the unknown data function $u = g(\mathbf{x})$ over the data subspaces $\mathbb{D}(\mathbf{x}, \theta)$ defined by unseen queries $\mathbf{q} = [\mathbf{x}, \theta]$. Based on the query-PLR estimate \hat{f} we seek a statistical learning methodology \mathcal{F} to extract a **data-PLR** estimate $\hat{g} \in \mathcal{L}_K$ from the query-PLR estimate \hat{f} , notated by $\hat{g} = \mathcal{F}(\hat{f})$ to fit the data function g . The challenge is:

$$\hat{g} = \arg \min_{g' \in \mathcal{L}_K} \{\text{MSE}(g') | g' = \mathcal{F}(\hat{f})\}.$$

- *CH3*: predict the data output \hat{u} of a random input data vector \mathbf{x} based on the data-PLR estimate \hat{g} , i.e., $\hat{u} = \hat{g}(\mathbf{x})$.

Consider the challenge CH1 and let us adopt the squared prediction error function $(y - f(\mathbf{x}, \theta))^2$ for penalizing errors in prediction of aggregate output y given a mean-value query $\mathbf{q} = [\mathbf{x}, \theta]$. This leads to a criterion for choosing a query-PLR function f , which minimizes the *Expected Prediction Error* (EPE):

$$\mathbb{E}[(y - f(\mathbf{x}, \theta))^2] = \mathbb{E}_{\mathbf{x}, \theta}[\mathbb{E}_y[(y - f(\mathbf{x}, \theta))^2 | \mathbf{x}, \theta]], \tag{5}$$

for all possible query points $\mathbf{x} \in \mathbb{R}^d$ and query radii $\theta \in \mathbb{R}$. To calculate the expectation in (5), we approximate EPE by the MSE in (2) over a finite number of query–answer pairs $(y, [\mathbf{x}, \theta])$. Before finding the family of this function that minimizes the EPE in (5), we rest on the *law of iterated expectations* for the dependent variable y from query point \mathbf{x} and radius θ , i.e., $\mathbb{E}[y] = \mathbb{E}[\mathbb{E}[y | \mathbf{x}, \theta]]$, where y breaks into two pieces, as follows:

Theorem 1 (Decomposition). $y = \mathbb{E}[y | \mathbf{x}, \theta] + \epsilon$, where ϵ is mean-independent of \mathbf{x} and θ , i.e., $\mathbb{E}[\epsilon | \mathbf{x}, \theta] = 0$ and therefore ϵ is uncorrelated with any function of \mathbf{x} and θ .

For proof of Theorem 1, refer to [32]. According to Theorem 1, the aggregate output y can be decomposed into a conditional expectation function $\mathbb{E}[y | \mathbf{x}, \theta]$, hereinafter referred to as *query regression function*, which is explained by \mathbf{x} and θ , and a left over (noisy component) which is orthogonal to (i.e., uncorrelated with) any function of \mathbf{x} and θ .

In our context, the query regression function is a good candidate for minimizing the EPE in (5) envisaged as a *local* representative value for answer y over the data subspace $\mathbb{D}(\mathbf{x}, \theta)$. Therefore, the conditional expectation function is the best predictor of answer y given $\mathbb{D}(\mathbf{x}, \theta)$:

Theorem 2 (Conditional Expectation Function). Let $f(\mathbf{x}, \theta)$ be any function of \mathbf{x} and θ . The conditional expectation function $\mathbb{E}[y | \mathbf{x}, \theta]$ solves the optimization problem: $\mathbb{E}[y | \mathbf{x}, \theta] = \arg \min_{f(\mathbf{x}, \theta)} \mathbb{E}[(y - f(\mathbf{x}, \theta))^2]$, i.e., it is the minimum mean squared error predictor of y given \mathbf{x}, θ .

For proof of Theorem 2, refer to [32].

Remark 4 We rely on Theorems 1 and 2 to build our statistical learning methodology \mathcal{F} for estimating a query-PLR \hat{f} and then, based on Theorem 1, we will be estimating the data-PLR \hat{g} only through \hat{f} and the answer–query pairs $(\mathbf{q}, y) = ([\mathbf{x}, \theta], y)$, without accessing the actual data pairs (\mathbf{x}, u) .

The solution to (5) is $f(\mathbf{x}, \theta) = \mathbb{E}[y|\mathbf{x}, \theta]$, i.e., the conditional expectation of answer y over $\mathbb{D}(\mathbf{x}, \theta)$. However, the number of data points $n_\theta(\mathbf{x})$ in $\mathbb{D}(\mathbf{x}, \theta)$ is finite; thus, such conditional expectation is approximated by averaging all data outputs u_i 's conditioning at $\mathbf{x}_i \in \mathbb{D}(\mathbf{x}, \theta)$. Moreover, the answer y of a query \mathbf{q} refers to the best regression estimator over $\mathbb{D}(\mathbf{x}, \theta)$. Each query center $\mathbf{x} \in \mathbb{D}(\mathbf{x}, \theta)$ and corresponding answer y provides information to locally learn the dependency between output u and input \mathbf{x} , i.e., the data function g . In this context, similar queries w.r.t. L_2 distance provide insight for data function g over overlapped data subspaces.

The query-PLR estimate function $\hat{f}(\mathbf{x}, \theta)$ from challenge CHI's outcome is used for estimating the multiple local line segments (i.e., local linear regression coefficients intercept and slope) of the data-PLR estimate function \hat{g} . This is achieved by a novel statistical learning methodology \mathcal{F} , which learns from a continuous query–answer stream $\{(\mathbf{q}_1, y_1), \dots, (\mathbf{q}_t, y_t)\}$ through the interactions between the users and the system. We can then formulate our problems are:

Problem 1 Given a finite number of query–answer pairs, approximate the query-PLR function $\hat{f}(\mathbf{x}, \theta)$ and predict the aggregate answer \hat{y} of an unseen query $\mathbf{q} = [\mathbf{x}, \theta]$.

Problem 2 Given only the query-PLR function $\hat{f}(\mathbf{x}, \theta)$ from Problem 1, approximate the data-PLR function $\hat{g}(\mathbf{x})$ and predict the data output \hat{u} of an unseen data input \mathbf{x} .

3.3 Preliminaries

3.3.1 Incremental learning and stochastic gradient descent

The stochastic gradient descent (SGD) [14] is an optimization method for minimizing an objective function $\mathcal{E}(\alpha)$, where α is a parameter and optimal parameter α^* minimizes the objective \mathcal{E} . SGD leads to fast convergence to the optimal parameter α^* by adjusting the estimated parameter α so far in the direction (negative gradient $-\nabla\mathcal{E}$) that improves the minimization of \mathcal{E} . SGD gradually changes the parameter α upon reception of a new training sample. The standard gradient descent algorithm updates the parameter α in $\mathcal{E}(\alpha)$ as: $\Delta\alpha = -\eta\nabla_\alpha\mathbb{E}[\mathcal{E}(\alpha)]$, where the expectation is approximated by evaluating the objective function \mathcal{E} and its gradient over all training pairs and $\eta \in (0, 1)$ is a learning rate. On the other hand, SGD computes the gradient of \mathcal{E} using only a single training pair at step t , that is we incrementally optimize

the objective \mathcal{E} . The update of parameter α_t at step t is given by: $\Delta\alpha_t = -\eta_t\nabla_{\alpha_t}\mathcal{E}(\alpha_t)$. The learning rate $\{\eta_t\} \in (0, 1)$ is a step-size schedule, which defines a slowly decreasing sequence of scalars that satisfy:

$$\sum_{t=1}^{\infty} \eta_t = \infty, \quad \sum_{t=1}^{\infty} \eta_t^2 < \infty. \quad (6)$$

Usually, we adopt a hyperbolic schedule from [14]:

$$\eta_t = \frac{1}{t+1}. \quad (7)$$

3.3.2 Adaptive vector quantization

Vector quantization refers to a data partitioning processes, which partitions a d -dim. real-valued data space \mathbb{R}^d into a fixed number of K subspaces. A vector quantizer (VQ) $v(\mathbf{x}) : \mathbf{x} \rightarrow \{1, \dots, K\}$ maps a vector $\mathbf{x} \in \mathbb{R}^d$ into a finite collection (a.k.a. codebook in signal processing) of K vector prototypes (codewords) $\{\mathbf{w}_1, \dots, \mathbf{w}_K\}$, which are spread in \mathbb{R}^d . A prototype \mathbf{w}_k represents a subspace of \mathbb{R}^d and behaves as a quantization vector. Given a distortion measure, a common measure for the performance of a VQ v is the expected distortion:

$$\mathbb{E}[\|\mathbf{x} - \mathbf{w}(\mathbf{x})\|^2] = \int_{\mathbb{R}^d} \|\mathbf{x} - \mathbf{w}(\mathbf{x})\|^2 dF(\mathbf{x}), \quad (8)$$

where $F(\mathbf{x})$ is the cumulative distribution of the vectors in \mathbb{R}^d and $\mathbf{w}(\mathbf{x})$ refers to the prototype selected by the VQ $v(\mathbf{x})$. For each random vector \mathbf{x} , the optimal VQ that minimizes (8) determines the best matched prototype from the codebook w.r.t. the Euclidean distance:

$$v(\mathbf{x}) = j : \|\mathbf{w}_j - \mathbf{x}\| < \|\mathbf{w}_k - \mathbf{x}\|, \quad \forall k \in [K], k \neq j. \quad (9)$$

An AVQ algorithm [2,49,57] is a VQ algorithm that incrementally learns as only the closest prototype \mathbf{w}_j to input vector \mathbf{x} , i.e., $v(\mathbf{x}) = j$, changes in response to \mathbf{x} observed once. During incremental partition of \mathbb{R}^d , a stream of input vectors \mathbf{x} are projected onto their closest prototypes (a.k.a. winners), which the latter adaptively move around the space to form optimal partitions (subspaces of \mathbb{R}^d) that minimize the Expected Quantization Error (EQE):

$$\mathbb{E} \left[\min_{k \in [K]} \|\mathbf{x} - \mathbf{w}_k\|^2 \right], \quad (10)$$

with winner prototype \mathbf{w}_j such that

$$\|\mathbf{w}_j - \mathbf{x}\| = \min_{k \in [K]} \|\mathbf{w}_k - \mathbf{x}\|.$$

4 Solution fundamentals

4.1 Methodology overview

We first proceed with a solution of Problem 1 to approximate the query function f through a query-PLR function \hat{f} . Then, we use the approximate \hat{f} to address Problem 2 to approximate the data function g by a data-PLR function \hat{g} .

Concerning Problem 1 and Theorem 2, we approximate $f(\mathbf{x}, \theta) = \mathbb{E}[y|\mathbf{x}, \theta]$ that minimizes (5). However, the answer y in (4) involves the average of the outputs $g(\mathbf{x}_i) = u_i, i \in [n_\theta(\mathbf{x})]$. Hence, $f(\mathbf{x}, \theta)$ is a non-trivial compound function of $g(\mathbf{x})$ for an arbitrary radius θ and L_p norm expressed by definition as:

$$f(\mathbf{x}, \theta) = \frac{1}{n_\theta(\mathbf{x})} \sum_{i \in [n_\theta(\mathbf{x})]} g(\mathbf{x}_i) : \mathbf{x}_i \in \mathbb{D}(\mathbf{x}, \theta), \tag{11}$$

where $n_\theta(\mathbf{x})$ varies depending on the *location* of the query point \mathbf{x} in the input data space \mathbb{R}^d and the query radius θ . Moreover, the nonlinearity of function g over certain subspaces is further propagated to f by definition of the aggregate answer y in (4). Hence, we must identify those data subspaces where data function g behaves *almost* linearly, which should be reflected in the function f approximation by \hat{f} . This will provide the key insight on approximating both functions f and g through a PLR family functions by learning the *unknown* finite set of local linear functions. We call those local linear functions as local linear mappings (LLMs) and derive the corresponding: query-space LLMs and data-space LLMs for the query function f and data function g , respectively.

In Problem 1, we approximate the query function $f(\mathbf{x}, \theta)$ with a set of query-space LLMs (or query-LLMs), each of which is constrained to a local region of the query space \mathbb{Q} , defined by *similar* queries w.r.t. L_2 distance. Similar queries are those queries with similar centers \mathbf{x} and similar radii θ . Our general idea for those query-space LLMs is the quantization of the query space \mathbb{Q} into a finite number of query subspaces \mathbb{Q}_k such that the query function f can be linearly approximated by a query-LLM $f_k, k = 1 \dots, K$, that is the k -th PLR segment. Those query subspaces may be rather large in areas of the query vectorial space \mathbb{Q} where the query function f indeed behaves approximately linear and must be smaller where this is not the case. The total number K of such query subspaces depends on the desired approximation (goodness of fit) and the query–answer prediction accuracy, and may be limited by the available issued queries since overfitting might occur.

Fundamentally, we incrementally quantize the query space \mathbb{Q} over a series of issued queries through quantization vectors, hereinafter referred to as query prototypes, in \mathbb{Q} . Then, we associate each query subspace \mathbb{Q}_k with a query-

LLM f_k in the query–answer space, where the query function f behaves approximately linear.

In Problem 2, principally each query subspace \mathbb{Q}_k is associated with a data subspace \mathbb{D}_k , i.e., for a query $\mathbf{q} \in \mathbb{Q}_k \subset \mathbb{R}^{d+1}$, its corresponding query point $\mathbf{x} \in \mathbb{D}_k \subset \mathbb{R}^d$. This implies that the input vector \mathbf{x} (of the query \mathbf{q}) is constrained to be drawn *only* from the k -th data subspace \mathbb{D}_k . Based on that association, we use the query-LLM f_k to estimate the data-LLM g_k , i.e., estimate the local intercept and slope of the data function g over the k -th data subspace \mathbb{D}_k .

4.2 Query local linear mapping

A query-LLM $f_k : \mathbb{Q}_k \rightarrow \mathbb{R}, k \in [K]$, approximates the dependency between aggregate answer y and query \mathbf{q} over the query subspace \mathbb{Q}_k defined by similar queries under L_2 distance. For modeling a query-LLM, we adopt the multivariate first-order Taylor expansion of the scalar-valued function $f(\mathbf{q}) = f(\mathbf{x}, \theta) = f(x_1, \dots, x_d, \theta)$ for a query \mathbf{q} near a query vector $\mathbf{q}_0 = [\mathbf{x}_0, \theta_0]$, that is:

$$f(\mathbf{q}) \approx f(\mathbf{q}_0) + \nabla f(\mathbf{q}_0)(\mathbf{q} - \mathbf{q}_0)^\top, \tag{12}$$

where $\nabla f(\mathbf{q}_0)$ is the gradient of query function f at query vector \mathbf{q}_0 , i.e., the $1 \times (d + 1)$ matrix of partial derivatives $\frac{\partial f}{\partial x_i}, i \in [d]$ and $\frac{\partial f}{\partial \theta_k}$.

As it will be analyzed and elaborated later, the query vector $\mathbf{q}_0 = [\mathbf{x}_0, \theta_0]$ and the gradient of query function f at \mathbf{q}_0 are not randomly selected. Instead, the proposed methodology \mathcal{F} attempts to find that query vector \mathbf{q}_0 and that gradient vector of query function f at \mathbf{q}_0 , which satisfies the following optimization properties:

- (OP1) minimization of the EPE in (5) as stated in challenge CH1;
- (OP2) minimization of the EQE in (10);
- (OP3) extraction of the data-LLM estimator from the query-LLM estimator such that the query-driven PLR \hat{g} fits well the underlying data function g , as stated in challenges CH2 and CH3.

As it will be proved later by Theorems 7, 8, and 9, we require a query-LLM f_k to derive from a specific Taylor’s approximation around the *local expectation query* $\mathbb{E}[\mathbf{q}] = [\mathbb{E}[\mathbf{x}], \mathbb{E}[\theta]]$ of queries $\mathbf{q} \in \mathbb{Q}_k$:

$$f_k(\mathbf{x}, \theta) \approx f_k(\mathbb{E}[\mathbf{x}], \mathbb{E}[\theta]) + \nabla f_k(\mathbb{E}[\mathbf{x}], \mathbb{E}[\theta])([\mathbf{x}, \theta] - [\mathbb{E}[\mathbf{x}], \mathbb{E}[\theta]])^\top \tag{13}$$

Specifically, the coefficients of the query-LLM f_k which satisfies the optimization properties OP1, OP2, and OP3, are:

- The **local intercept**, with two components: the local expectation of answer y , i.e., $\mathbb{E}[y] = f_k(\mathbb{E}[\mathbf{x}], \mathbb{E}[\theta])$, notated by the scalar coefficient y_k ; and the local expectation query $\mathbb{E}[\mathbf{q}] = [\mathbb{E}[\mathbf{x}], \mathbb{E}[\theta]]$ notated by the vectorial coefficient $\mathbf{w}_k = [\mathbf{x}_k, \theta_k] \in \mathbb{Q}_k$, with $\mathbf{x}_k = \mathbb{E}[\mathbf{x}]$ and $\theta_k = \mathbb{E}[\theta]$ such that $[\mathbf{x}, \theta] \in \mathbb{Q}_k$. Hereinafter, \mathbf{w}_k is referred to as the **prototype** of the query subspace \mathbb{Q}_k .
- The **local slope** $\mathbf{b}_k = [\mathbf{b}_{X,k}, b_{\theta,k}]$ of f_k over \mathbb{Q}_k , which denotes the gradient $\nabla f_k(\mathbb{E}[\mathbf{x}], \mathbb{E}[\theta])$ of f_k at the local expectation query \mathbf{w}_k .

Based on these constructs that satisfy OP1, OP2, and OP3, the query-LLM f_k is rewritten as:

$$f_k(\mathbf{x}, \theta) \approx y_k + \mathbf{b}_{X,k}(\mathbf{x} - \mathbf{x}_k)^\top + b_{\theta,k}(\theta - \theta_k). \quad (14)$$

Up to now, our challenge is for each query-LLM f_k to estimate the parameter $\alpha_k = (y_k, \mathbf{b}_k, \mathbf{w}_k)$ in light of minimizing the EPE as stated in OP1 by the following constrained optimization problem:

$$\begin{aligned} \alpha_k^* &= \arg \min_{y_k, \mathbf{b}_k, \mathbf{w}_k} \mathbb{E}[(y - y_k - \mathbf{b}_k([\mathbf{x}, \theta] - \mathbf{w}_k)^\top)^2] \\ &\text{subject to } y_k = f_k(\mathbf{x}_k, \theta_k), \forall k \in [K], [\mathbf{x}, \theta] \in \mathbb{Q}_k. \end{aligned} \quad (15)$$

Remark 5 It is worth mentioning that the constraint $y_k = f_k(\mathbf{x}_k, \theta_k), \forall k \in [K]$ in the optimization problem (15) requires that in each query subspace \mathbb{Q}_k , the corresponding query-LLM f_k refers to a (hyper)plane that minimizes the EPE and, also, given a query \mathbf{q} with a query point $\mathbf{x} = \mathbb{E}[\mathbf{x}|\mathbf{x} \in \mathbb{D}(\mathbf{x}_k, \theta_k)]$ being the *centroid* of the corresponding data subspace \mathbb{Q}_k and radius $\theta = \mathbb{E}[\theta|\mathbf{x} \in \mathbb{D}(\mathbf{x}_k, \theta_k), \mathbf{q} \in \mathbb{Q}_k]$ being the *mean radius* of all the queries from \mathbb{Q}_k , it secures that f_k supports the OP2 and OP3.

However, we need to further optimize α_k to satisfy *also* the optimization properties OP2 and OP3.

4.3 Our statistical learning methodology

Our statistical learning methodology \mathcal{F} departs from the optimization problem in (15) to additionally support the optimization properties OP2 and OP3. Our methodology is formally based on a joint optimization problem of *optimal quantization and regression*. This is achieved by incrementally identifying within-subspaces linearities in the query space and then estimating therein the query-LLM coefficients such that we preserve the optimization properties OP1, OP2, and OP3.

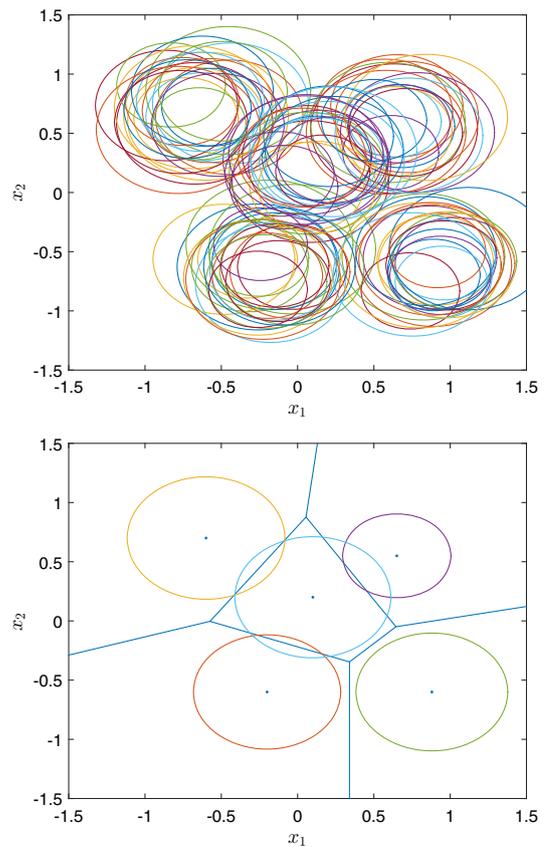


Fig. 5 Example 3. (Upper) 2D representation of queries and (lower) their query prototypes onto the input space $[-1.5, +1.5]^2$

4.3.1 Joint quantization–regression optimization for query-LLMs

Firstly, we should identify the subspaces \mathbb{Q}_k , i.e., determine their prototypes \mathbf{w}_k , their number K , and their coefficients y_k and \mathbf{b}_k , in which the query function f can be well approximated by LLMs. We identify the prototypes \mathbf{w}_k (associated with $\mathbb{Q}_k, k \in [K]$) by incrementally partitioning the query space $\mathbb{Q} = \cup_{k=1}^K \mathbb{Q}_k$. Before elaborating on our methodology, we provide an illustrative example on query space quantization.

Example 3 Figure 5(upper) shows 1,000 issued queries $\mathbf{q}_t = [\mathbf{x}_t, \theta_t]$ over the 2D input space $\mathbf{x} = (x_1, x_2) \in [-1.5, 1.5]^2$. Each query is represented by a disk with center \mathbf{x}_t and radius θ_t . Figure 5(lower) shows the five query prototypes $\mathbf{w}_k = [\mathbf{x}_k, \theta_k], k \in [5]$ projected onto the 2D input space. Note, centers \mathbf{x}_k of the prototypes \mathbf{w}_k correspond to Voronoi sites under L_2 onto the data space.

The introduction of the query space quantization before predicting the query’s answer, i.e., regression of aggregate answer y on the query vector \mathbf{q} , raises a natural fundamental question:

Question: *Since by query quantization will lose information and, thus, likely damage the prediction performance of query function approximate \hat{f} , would it not be better to always proceed with regression based on the original, un-quantized, query vectors?*

Answer: There is one response on that question: one can consider a VQ as part of the regression estimate function \hat{f} . The overall goal is not purely regression, i.e., query–answer prediction using query function f , but also PLR fitting of the underlying data function g . The VQ yields several benefits starting from constructing the query prototypes $\{\mathbf{w}_k = [\mathbf{x}_k, \theta_k]\}_{k=1}^K$ of the query-LLMs f_k , that is minimizing the EQE (OP2), to constructing the intercepts and slopes $\{(y_k, \mathbf{b}_k)\}_{k=1}^K$, which are needed to minimize the EPE (OP1) and also to derive the data-LLMs g_k (OP3). And, based on Theorem 7, the query prototypes \mathbf{w}_k converge to the optimal vector prototypes only when adopted by the VQ; specifically by an incrementally growing AVQ, as it will be elaborated later. The inclusion of estimating the query prototypes \mathbf{w}_k provides a methodology not suggested by the regression/prediction goal alone, which nonetheless allows one to weight the prediction performance as being the more important criterion and which may eventually yield better regression algorithms. However, in this case our goal has with one model to satisfy the optimization properties OP1, OP2, and OP3 simultaneously, and this can be viewed as finding an algorithm for *jointly* designing a VQ and PLR-based predictor to yield performance close to that achievable by an optimal PLR-based predictor operating on the original answer–query pairs and input–output data pairs, as it will be shown at our performance Sect. 8.

Given a finite and unknown number of query prototypes K and a VQ $v(\mathbf{q})$ over the query space, the query quantization performance measured by mean squared distortion error \mathcal{J} is given by:

$$\mathcal{J}(\{\mathbf{w}_k\}) = \sum_{k=1}^K \mathbb{E}[\|\mathbf{q} - \mathbf{w}_k\|^2 | v(\mathbf{q}) = k] P(v(\mathbf{q}) = k) \quad (16)$$

where $P(v(\mathbf{q}) = k)$ is the probability the VQ maps query \mathbf{q} to the query prototype \mathbf{w}_k . We obtain the minimum value of $\mathcal{J}(\{\mathbf{w}_k\})$, i.e.,

$$\mathcal{J}(\{\mathbf{w}_k\}) \geq \sum_{k=1}^K \mathbb{E}[\min_i \|\mathbf{q} - \mathbf{w}_i\|^2 | v(\mathbf{q}) = k] \cdot P(v(\mathbf{q}) = k) \quad (17)$$

which is the lower bound achievable if each query prototype \mathbf{w}_k is chosen by the VQ to be the centroid of the conditional expectation:

$$\mathbf{w}_k = \arg \min_{i \in [K]} \mathbb{E}[\|\mathbf{q} - \mathbf{w}_i\| | v(\mathbf{q}) = k]. \quad (18)$$

In parallel, within each \mathbb{Q}_k , we incrementally estimate the PLR coefficients (y_k, \mathbf{b}_k) of each query-LLM f_k . These coefficients are learned *only* from similar query–answer pairs whose queries belong to the query subspace \mathbb{Q}_k .

We propose a hybrid model by partitioning \mathbb{Q} into K (unknown) subspaces \mathbb{Q}_k , i.e., unsupervised learning of \mathbf{w}_k to minimize the EQE, and supervised learning of the coefficients y_k and \mathbf{b}_k to minimize the EPE. The idea is that each query subspace \mathbb{Q}_k associates the LLM f_k with the query prototype \mathbf{w}_k , as shown in Fig. 6 (see Example 4), conditioned on the result of the VQ. In other words, the regression performance is provided by the conditional EPE \mathcal{H} :

$$\begin{aligned} \mathcal{H}(\{y_k, \mathbf{b}_k\}) &= \sum_{k=1}^K \mathbb{E} \left[(y - y_k - \mathbf{b}_k(\mathbf{q} - \mathbf{w}_k)^\top)^2 \right] P(v(\mathbf{q}) = k) \quad (19) \end{aligned}$$

We obtain the minimum value of \mathcal{H} :

$$\begin{aligned} \mathcal{H}(\{y_k, \mathbf{b}_k\}) &\geq \sum_{k=1}^K \min_i \mathbb{E}[(y - y_i - \mathbf{b}_i(\mathbf{q} - \mathbf{w}_i)^\top)^2] P(v(\mathbf{q}) = k), \quad (20) \end{aligned}$$

which is the lower bound achievable if the regression is chosen to minimize the prediction error derived by the k -th query-LLM f_k , which corresponds to the closest query prototype \mathbf{w}_k , i.e., the VQ chooses $k = v(\mathbf{q})$ such that the query prototype \mathbf{w}_k is the winner prototype.

The joint quantization–regression optimization incrementally minimizes the two objective functions: EQE \mathcal{J} and conditional EPE \mathcal{H} upon receiving a new query–answer pair (\mathbf{q}, y) , that is, our constraint joint optimization problem is:

$$\mathcal{J}(\{\mathbf{w}_k\}) = \mathbb{E} \left[\min_k \|\mathbf{q} - \mathbf{w}_k\|^2 \right], \quad (21)$$

$$\begin{aligned} \mathcal{H}(\{y_k, \mathbf{b}_k\}) &= \mathbb{E} \left[(y - y_k - \mathbf{b}_k(\mathbf{q} - \mathbf{w}_k)^\top)^2 \mid v(\mathbf{q}) = k \right] \\ &\text{subject to } k = \arg \min_{i \in [K]} \|\mathbf{q} - \mathbf{w}_i\| \\ y_k &= f_k(\mathbf{x}_k, \theta_k), \forall k \in [K]. \quad (22) \end{aligned}$$

The objective function in (21) corresponds to optimal partitioning of the query space into K partitions (OP1), each with a prototype. The objective function in (22) corresponds to a conditional EPE **conditioned** on the k -th query prototype \mathbf{w}_k , which is the closest to the query \mathbf{q} (OP2). The constraints will ensure the derivation of the data-LLM \hat{g}_k from the query-LLM f_k at it will be shown later (OP3).

The quantization of the query space \mathbb{Q} operates as a mechanism to project an unseen query \mathbf{q} to the closest

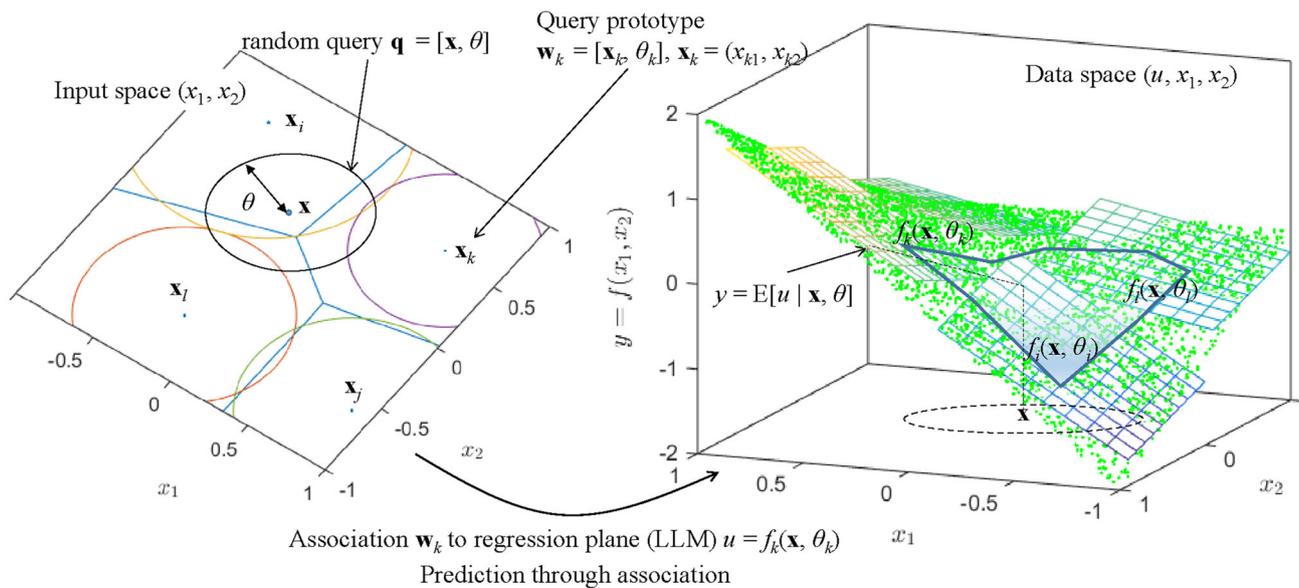


Fig. 6 Example 4. Association of a query prototype \mathbf{w}_j with the query-LLM f_j in the 3D data space (u, x_1, x_2) with underlying data function $u = g(x_1, x_2) = x_1(x_2 + 1)$

query subspace \mathbb{Q}_k w.r.t. L_2 distance from the prototype \mathbf{w}_k , wherein we learn the dependency between the aggregate answer y with the query point \mathbf{x} and radius θ .

Example 4 Figure 6 depicts the *association* from the query space to the 3D data space. A query prototype \mathbf{w}_j , a disk on the input space (x_1, x_2) , is now associated with the query-LLM $f_j(\mathbf{x}, \theta)$ and its corresponding regression plane $u_j = f_j(\mathbf{x}, \theta_j)$ on the data space (u, x_1, x_2) , which approximates the actual data function $u = g(x_1, x_2) = x_1(x_2 + 1)$. Note, in each *local* plane, we learn the local intercept y_j and slope \mathbf{b}_j where \mathbf{x}_j is the representative of the data subspace \mathbb{D}_j (see Theorems 7, 8 and 9).

4.3.2 Data-LLM function derivation from query-LLM function

Concerning Problem 1, the prediction of the aggregate output \hat{y} of an unseen query \mathbf{q} is provided by neighboring query-LLM functions f_k , as will be elaborated later. Concerning Problem 2, we derive the linear data-LLM function g_k (intercept and slope) between output u and input \mathbf{x} over the data subspace \mathbb{D} given the query-LLM function f_k . Then, we approximate the PLR estimate of data function g by interpolating many data-LLMs.

Based on Theorems 1 and 2, we obtain that the data output $u = g(\mathbf{x}) = \mathbb{E}[u | \mathbf{x}] + \epsilon$. In that context, we can approximate the data function $g(\mathbf{x})$ over the data subspace \mathbb{D}_k , i.e., the PLR segment g_k from the corresponding query-LLM function f_k conditioned on the mean radius θ_k .

Theorem 3 The data function $g(\mathbf{x})$ in the data subspace \mathbb{D}_k is approximated by the linear regression function:

$$u = g(\mathbf{x}) \approx y_k + \mathbf{b}_{X,k}(\mathbf{x} - \mathbf{x}_k)^\top = f_k(\mathbf{x}, \theta_k),$$

with slope: $\mathbf{b}_{X,k}$ and intercept $y_k - \mathbf{b}_{X,k}\mathbf{x}_k^\top$.

Proof For any random variable u, \mathbf{x}, θ , and y we can easily prove that $\mathbb{E}[\mathbb{E}[y | \mathbf{x}, \theta] | \mathbf{x}] = \mathbb{E}[y | \mathbf{x}]$. Since $\mathbb{E}[y | \mathbf{x}, \theta] = y_k + \mathbf{b}_{X,k}(\mathbf{x} - \mathbf{x}_k)^\top + b_{\Theta,k}(\theta - \theta_k)$, we obtain that

$$\mathbb{E}[y | \mathbf{x}] = \mathbb{E}[\mathbb{E}[y | \mathbf{x}, \theta] | \mathbf{x}] = y_k + \mathbf{b}_{X,k}(\mathbf{x} - \mathbf{x}_k)^\top,$$

with $\mathbb{E}[\mathbf{x} | \mathbf{x}] = \mathbf{x}$ and $\mathbb{E}[\theta | \mathbf{x}] = \mathbb{E}[\theta] = \theta_k$, by definition of the y function and the independence assumption of \mathbf{x} and θ . Through decomposition in Theorem 1, we approximate the dependency of u with \mathbf{x} through the conditional expectation function:

$$u(\mathbf{x}) = \mathbb{E}[u | \mathbf{x}] + \epsilon = \mathbb{E}[\mathbb{E}[u | \mathbf{x}, \theta] | \mathbf{x}] + \epsilon = \mathbb{E}[y | \mathbf{x}] + \epsilon,$$

since by Definition 5, $y = \mathbb{E}[u | \mathbf{x}, \theta]$. Thus, $u = g(\mathbf{x})$ is approximated by the linear regression function $\mathbb{E}[u | \mathbf{x}] = y_k + \mathbf{b}_{X,k}(\mathbf{x} - \mathbf{x}_k)^\top$ having $\mathbb{E}[\epsilon | \mathbf{x}] = 0$. \square

Example 5 We provide the following visualization in Fig. 7 to better explain and provide insights of the data-LLMs derivation from query-LLMs. Specifically, Fig. 7 interprets the mapping methodology \mathcal{F} from the query-LLMs to the data-LLMs after obtaining the optimal values for the parameters that satisfy the optimization properties OP1, OP2, and

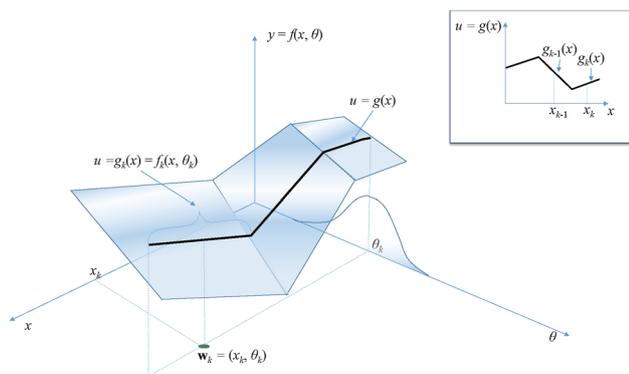


Fig. 7 Example 5: The three query-LLMs f_1, f_2, f_3 as three-dimensional planes in the query space and their corresponding/derived data-LLMs g_1, g_2, g_3 as line segments over the data subspace (inner plot)

OP3. We observe three regression planes in the query–answer space (x, θ, y) , which are approximated by the three query-LLMs f_1, f_2 and f_3 . This indicates the PLR approximate of the query function f . Now, focus on the regression plane $f_k(x, \theta)$ along with the query prototype $\mathbf{w}_k = [x_k, \theta_k]$. The corresponding data-LLM function $g_k(x)$ for those data inputs $x \in \mathbb{D}(x_k, \theta_k)$ derives from the query-LLM $f_k(x, \theta_k)$ since, as proved in Theorem 7, the radius θ_k is the expected radius of all the queries \mathbf{q} with $\mathbf{w}_k = \mathbb{E}[\mathbf{q}|v(\mathbf{q}) = k]$, i.e., $\theta_k = \mathbb{E}[\theta|v(\mathbf{q}) = k]$. The data-LLM is represented by the linear regression approximation $g_k(x)$ laying on the regression plane defined by the query-LLM f_k . We obtain the PLR data approximate g over all data input x space by following the data-LLMs g_k over the planes defined by the query-LLMs f_k , as illustrated in the inner plot in Fig. 7. As we are moving from one query-LLM f_{k-1} to the next one f_k , we derive the corresponding data-LLMs g_{k-1} to g_k by setting $\theta = \theta_{k-1}$ and $\theta = \theta_k$ to the query-LLM definitions (linear models) such that: $\theta_{k-1} = \mathbb{E}[\theta|v(\mathbf{q}) = k - 1]$ and $\theta_k = \mathbb{E}[\theta|v(\mathbf{q}) = k]$, respectively. Hence, based on this trajectory we derive the PLR estimate data function \hat{g} of the underlying data function g .

Remark 6 It is worth noting that the data function g based on Theorem 3 is achieved based only by the knowledge extracted from answer–query pairs and *not* by accessing the data points.

5 Query-driven statistical learning methodology

In this section we propose our query-driven statistical learning algorithm for our methodology through which all the query-LLM parameters α_k minimize both (21) and (22). Then, we provide the PLR approximation error bound of

the PLR estimate functions f_k of query function f and the impact of our VQ algorithm in this error.

Let us focus on the EQE \mathcal{J} in (21) and liaise with Example 3 (Fig. 5). We seek the best possible approximation of a random query \mathbf{q} out of the set $\{\mathbf{w}_k\}_{k=1}^K$ of finite K query prototypes. We consider the closest neighbor projection of query \mathbf{q} to a query prototype \mathbf{w}_j , which represents the j -th query subspace $\mathbb{Q}_j \subset \{\mathbf{q} \in \mathbb{Q} : \|\mathbf{q} - \mathbf{w}_j\|_2 = \min_k \|\mathbf{q} - \mathbf{w}_k\|_2\}$. We incrementally minimize the objective function \mathcal{J} with the presence of a random query \mathbf{q} and update the winning prototype \mathbf{w}_j accordingly. However, the number of the query subspaces and, thus, query prototypes $K > 0$, is completely unknown and not necessarily constant. The key problem is to decide on an appropriate K value. In the literature a variety of AVQ methods exists, however, not suitable for incremental implementation, because K must be supplied in advance.

We propose a *conditionally growing* AVQ algorithm under L_2 distance in which the prototypes are sequentially updated with the incoming queries and their number is adaptively growing, i.e., the number K increases if a criterion holds true. Given that K is not available a-priori, our VQ minimizes the objective \mathcal{J} with respect to a threshold value ρ . This threshold determines the current number of prototypes K . Initially, the query space has a unique (random) prototype, i.e., $K = 1$. Upon the presence of a query \mathbf{q} , our algorithm first finds the winning query prototype \mathbf{w}_j and then updates the prototype \mathbf{w}_j only if the condition $\|\mathbf{q} - \mathbf{w}_j\|_2 \leq \rho$ holds true. Otherwise, the query \mathbf{q} is currently considered as a *new* prototype, thus, increasing the value of K by one. Through this conditional quantization, our VQ algorithm leaves the random queries to self-determine the resolution of quantization. Evidently, a high ρ value would result in coarse query space quantization (i.e., low resolution partition) while low ρ values yield a fine-grained quantization of the query space. The parameter ρ is associated with the stability–plasticity dilemma a.k.a. *vigilance* in Adaptive Resonance Theory [30]. In our case, the vigilance ρ represents a threshold of similarity between queries and prototypes, thus, guiding our VQ algorithm in determining whether a new query prototype should be formed.

Remark 7 To give a physical meaning to the vigilance parameter ρ , we express it through a set of coefficient percentages $a_i \in (0, 1)$ and $a_\theta \in \theta$ of the value ranges of each dimension x_i of query center $\mathbf{x} \in \mathbb{R}^d$ and radius θ , respectively. Then, we obtain that $\rho = \|[a_1, \dots, a_d]\|_2 + a_\theta$ and if we let $a_i = a_\theta = a \in (0, 1), \forall i$, then the vigilance parameter is rewritten as:

$$\rho = a(d^{1/2} + 1) \tag{23}$$

A high quantization coefficient a value over high-dimensional data results in a low number of prototypes and vice versa.

Let us now focus on the EPE \mathcal{H} in (22) and liaise with Examples 3 and 4 (Figs. 5 and 6). The objective function \mathcal{H} is conditioned on the winning query-prototype index $j = \arg \min_k \|\mathbf{q} - \mathbf{w}_k\|_2$, i.e., it is guided by the VQ $v(\mathbf{q}) = j$. Our target is to incrementally learn the query-LLM coefficients offset y_j and slope \mathbf{b}_j of the LLM function f_j , which are associated with the winning query prototype $\mathbf{w}_j \in \mathbb{Q}_j$ for a random query \mathbf{q} .

We incrementally minimize both objective functions \mathcal{J} and \mathcal{H} given a series of issued query–answer pairs (\mathbf{q}_t, y_t) to estimate the unknown parameters set $\alpha = \cup_{k=1}^K \alpha_k$, with LLM parameter $\alpha_k = (y_k, \mathbf{b}_k, \mathbf{w}_k)$ through SGD. Our algorithm processes successive query–answer pairs (\mathbf{q}_t, y_t) until a termination criterion $\max(\Gamma_t^{\mathcal{J}}, \Gamma_t^{\mathcal{H}}) \leq \gamma$. Specifically, $\Gamma_t^{\mathcal{J}}$ and $\Gamma_t^{\mathcal{H}}$ refer to the distance between successive estimates at steps $t-1$ and t of the query prototypes w.r.t. objective \mathcal{J} and query-LLM coefficients w.r.t. objective \mathcal{H} , respectively. The algorithm stops at the first step/observation t^* where:

$$t^* = \arg \min\{\tau > 0 : \max(\Gamma_\tau^{\mathcal{J}}, \Gamma_\tau^{\mathcal{H}}) \leq \gamma\}, \quad (24)$$

where

$$\begin{aligned} \Gamma_t^{\mathcal{J}} &= \frac{1}{K} \sum_{k=1}^K \|\mathbf{w}_{k,t} - \mathbf{w}_{k,t-1}\|_2, \\ \Gamma_t^{\mathcal{H}} &= \frac{1}{K} \left(\sum_{k=1}^K \|\mathbf{b}_{k,t} - \mathbf{b}_{k,t-1}\|_2 + |y_{k,t} - y_{k,t-1}| \right). \end{aligned} \quad (25)$$

The update rules for the optimization parameter α in our SGD-based dual EQE/EPE optimization of the objective functions \mathcal{J} and \mathcal{H} are provided in Theorem 4.

Theorem 4 *Given a pair of query–answer (\mathbf{q}, y) and its winning query prototype \mathbf{w}_j , the optimization parameter α converges to the optimal parameter α^* , if it is updated as:*

If $\|\mathbf{q} - \mathbf{w}_j\| \leq \rho$, then

$$\begin{aligned} \Delta \mathbf{w}_j &= \eta(\mathbf{q} - \mathbf{w}_j) \\ \Delta \mathbf{b}_j &= \eta(y - y_j - \mathbf{b}_j(\mathbf{q} - \mathbf{w}_j)^\top)(\mathbf{q} - \mathbf{w}_j) \\ \Delta y_j &= \eta(y - y_j - \mathbf{b}_j(\mathbf{q} - \mathbf{w}_j)^\top). \end{aligned}$$

If $\|\mathbf{q} - \mathbf{w}_j\| > \rho$, then $\Delta \mathbf{w}_j = \mathbf{0}$, $\Delta \mathbf{b}_j = \mathbf{0}$, $\Delta y_j = 0$. For any prototype \mathbf{w}_k , which is not winner ($k \neq j$):

$$\Delta \mathbf{w}_k = \mathbf{0}, \Delta \mathbf{b}_k = \mathbf{0}, \Delta y_k = 0,$$

where the learning rate $\eta \in (0, 1)$ is defined in Sect. 3.3.

Proof We adopt SGD to minimize both (21) and (22). \mathcal{J} and \mathcal{H} are minimized by updating $\alpha = \{y_k, \mathbf{w}_k, \mathbf{b}_k\}$ in the

negative direction of their sum of gradients. We obtain the set of update rules:

$$\begin{aligned} \Delta \mathbf{w}_{k,t} &= -\eta_t \nabla \mathcal{J}(\{\mathbf{w}_{k,t}\}), \\ \Delta \mathbf{b}_{k,t} &= -\eta_t \nabla_{\mathbf{b}_k} \mathcal{H}(\{y_{k,t}, \mathbf{b}_{k,t}\}), \\ \Delta y_{k,t} &= -\eta_t \nabla_{y_k} \mathcal{H}(\{y_{k,t}, \mathbf{b}_{k,t}\}). \end{aligned}$$

The objective function \mathcal{J} requires competitive learning, thus, at each step we update the winner \mathbf{w}_j , while \mathcal{H} is conditional updated with respect to $j = v(\mathbf{q})$. The \mathbf{w}_j converges when $\mathbb{E}[\Delta \mathbf{w}_j] = \mathbf{0}$ given that $\|\mathbf{q} - \mathbf{w}_j\| \leq \rho$. We require at the convergence that each query \mathbf{q} is assigned to its winner \mathbf{w}_j with probability 1, that is, $P(\|\mathbf{q} - \mathbf{w}_j\| \leq \rho) = 1$, which means that no other prototypes are generated. Therefore, based on Markov's inequality we obtain that:

$$P(\|\mathbf{q} - \mathbf{w}_j\| \geq \rho) \leq \frac{\mathbb{E}[\|\mathbf{q} - \mathbf{w}_j\|]}{\rho}$$

or $P(\|\mathbf{q} - \mathbf{w}_j\| \leq \rho) \geq 1 - \frac{\mathbb{E}[\|\mathbf{q} - \mathbf{w}_j\|]}{\rho}$. To obtain $P(\|\mathbf{q} - \mathbf{w}_j\| \leq \rho) \rightarrow 1$ we have either $\rho \rightarrow \infty$ or $\mathbb{E}[\|\mathbf{q} - \mathbf{w}_j\|] \rightarrow 0$. However, ρ is a real finite number and relatively small, since it interprets the concept of neighborhood. Hence, we require that $\mathbb{E}[\|\mathbf{q} - \mathbf{w}_j\|] \rightarrow 0$, i.e., $\mathbb{E}[(\mathbf{q} - \mathbf{w}_j)] = \mathbf{0}$, or $\mathbb{E}[\Delta \mathbf{w}_j] = \mathbf{0}$, while completes the proof. \square

The provided training Algorithm 1 processes a random pair of query–answer one at a time from a training set $\mathcal{T} = \{(\mathbf{q}, y)\}$; see also Fig. 4. In the initialization phase of the training algorithm, there is only one query prototype \mathbf{w}_1 , i.e., $K = 1$, which corresponds to the first query, while the associated query-LLM coefficients \mathbf{b}_1 and y_1 are initialized to $\mathbf{0}$ and 0, respectively. For the t -th random pair (\mathbf{q}_t, y_t) and onwards with $t \geq 2$, the algorithm either updates the closest prototype to \mathbf{q}_t (out of the so far K prototypes) if their L_2 distance is less than ρ , or adds a new prototype increasing K by one and then the new LLM coefficients are initialized. The algorithm stops updating the query prototypes and query-LLM coefficients at the first step t where $\max(\Gamma_t^{\mathcal{J}}, \Gamma_t^{\mathcal{H}}) \leq \gamma$. At that time and onwards, the algorithm returns the parameters set α and no further modification is performed, i.e., the algorithms has converged.

Through the incremental training of the parameters set $\alpha = \{(y_k, \mathbf{b}_k, \mathbf{w}_k)\}_{k=1}^K$, each query-LLM function f_k has estimated its parameters. The PLR approximation error bound for the LLM function f_k around the query prototype \mathbf{w}_k depends on the dimension d and curvature (second derivative) of the function f_k in the query subspace \mathbb{Q}_k as provided in Theorem 5. The approximation depends on the resolution of quantization K . Notably, the more prototypes K , the better the approximation of the query function f is achieved by query-LLMs, as proved in Theorem 6.

ALGORITHM 1: Query-LLM and VQ Training Algorithm.

Input: vigilance ρ , convergence threshold γ
Result: query-LLM parameters and query prototypes of set α

```

begin
  Get first query–answer pair  $(\mathbf{q}, y)$ ;
  Init.:  $\alpha = \{(y_1 = 0, \mathbf{b}_1 = \mathbf{0}, \mathbf{w}_1 = \mathbf{q})\}, K \leftarrow 1$ ;
  repeat
    Get next query–answer pair  $(\mathbf{q}, y)$ ;
    Find closest query prototype  $j = v(\mathbf{q})$ , i.e.,
     $j = \arg \min_k \|\mathbf{w}_k - \mathbf{q}\|_2$ ;
    if  $\|\mathbf{w}_j - \mathbf{q}\|_2 \leq \rho$  then
      Update  $y_j, \mathbf{b}_j, \mathbf{w}_j$  using Theorem 4.
    else
       $K \leftarrow K + 1$ ;
      Initialize  $(y_K, \mathbf{b}_K) = (0, \mathbf{0}), \mathbf{w}_K \leftarrow \mathbf{q}$ ;
       $\alpha \leftarrow \alpha \cup \{(y_K, \mathbf{b}_K, \mathbf{w}_K)\}$ ;
    end
    Calculate  $\Gamma^{\mathcal{J}}, \Gamma^{\mathcal{H}}$ ;
  until  $\max(\Gamma^{\mathcal{J}}, \Gamma^{\mathcal{H}}) \leq \gamma$ ;
end
  
```

Theorem 5 For a random query \mathbf{q} with closest query prototype \mathbf{w}_k , the conditional expected approximation error bound for the LLM function f_k in query subspace \mathbb{Q}_k around \mathbf{w}_k is:

$$\mathbb{E}[|f(\mathbf{x}, \theta) - f_k(\mathbf{x}, \theta)| | \mathbf{w}_k] \leq C_k O(d)$$

with

$$C_k \geq \frac{1}{2} \max_{i \in [d+1]} \left| \frac{\partial^2 f(\mathbf{q})}{\partial q_i^2} \right|_{\mathbf{q}=\mathbf{w}_k}.$$

Proof The query-LLM $f_k(\mathbf{x}, \theta)$ in the query subspace \mathbb{Q}_k refers to the 1st Taylor series approximation of $f(\mathbf{x}, \theta)$ around the prototype $\mathbf{w}_k = [\mathbf{x}_k, \theta_k]$. The approximation error is then:

$$\lambda = |f(\mathbf{x}, \theta) - f_k(\mathbf{x}, \theta)|.$$

Assume that $f(\mathbf{x}, \theta)$ is differential at most two times on \mathbb{Q}_k . For simplicity of notation, let $\mathbf{q} = [x_1, \dots, x_d, \theta] = [q_1, \dots, q_{d+1}]$ and $f_{(i)}(\mathbf{q})$ and $f_{k,(i)}(\mathbf{q})$ be the actual and approximation function on dimension q_i , where all the other dimensions are fixed. Then by Taylor’s inequality theorem [33] (based on the Mean Value Theorem), we obtain that the approximation error bound $\lambda_{(i)}$ is $\lambda_{(i)} \leq \frac{1}{2} C_{(i)} (q_i - w_{ki})^2$, with prototype $\mathbf{w}_k = [w_{k1}, \dots, w_{k(d+1)}]$ and constant $C_{(i)} \geq \left| \frac{\partial^2 f(\mathbf{q})}{\partial q_i^2} \right|_{\mathbf{q}=\mathbf{w}_k}$. By accumulating the approximation error bounds $\lambda_{(i)}, \forall i$, we obtain that:

$$\lambda = \sum_{i=1}^{d+1} \lambda_{(i)} \leq \frac{1}{2} \max_{i \in [d+1]} C_{(i)} \sum_{i=1}^{d+1} (q_i - w_{ki})^2 = C_k \|\mathbf{q} - \mathbf{w}_k\|_2^2,$$

with $C_k = \frac{1}{2} \max_{i \in [d+1]} C_{(i)}$. Now, from the convergence Theorem 7, the query prototype \mathbf{w}_k is the centroid of all queries $\mathbf{q} \in \mathbb{Q}_k$. If we define the random vector $\mathbf{z} = \mathbf{q} - \mathbf{w}_k$, then the L_2^2 norm $\|\mathbf{z}\|_2^2 = \|\mathbf{q} - \mathbf{w}_k\|_2^2$ is distributed according to the χ^2 (Chi-squared) distribution with $d + 1$ degrees of freedom given that $\mathbb{E}[\mathbf{z}] = \mathbb{E}[\mathbf{q}] - \mathbf{w}_k = \mathbf{0}$ from Theorem 7 and $\mathbf{q} \in \mathbb{Q}_k$. Hence, we obtain that $\mathbb{E}[\|\mathbf{z}\|_2^2] = d + 1$ and the expected approximation error bound is

$$\mathbb{E}[\lambda | \mathbf{w}_k] \leq C_k (d + 1).$$

□

Theorem 6 For a random query \mathbf{q} , the expected approximation error given K query-LLM functions $f_k, k \in [K]$ is bounded by $\sum_{k \in [K]} C_k O(\frac{d}{K})$, where C_k is defined in Theorem 5.

Proof Upon a random query and the quantization of the query space \mathbb{Q} into K LLMs, each with a query prototype \mathbf{w}_k , the derived approximation error of f through all $f_k, k \in [K]$, is

$$\mathbb{E}[\lambda] = \sum_{k=1}^K \mathbb{E}[\lambda_k | \mathbf{w}_k] P(\mathbf{w}_k),$$

where λ_k is the conditional approximation error bound given that \mathbf{q} is assigned to prototype \mathbf{w}_k and $P(\mathbf{w}_k)$ is the prior probability of \mathbf{w}_k . Provided that all \mathbf{w}_k are equiprobable for being assigned to queries, i.e., $P(\mathbf{w}_k) = \frac{1}{K}, \forall k$, then:

$$\mathbb{E}[\lambda] = \frac{1}{K} \sum_{k=1}^K \lambda_k \leq \frac{d+1}{K} \sum_{k=1}^K C_k,$$

where C_k is defined in Theorem 5.

□

6 Data and query functions approximation and prediction

In this section we propose an algorithm that uses the query-LLM functions to approximate the PLR data function g over a data subspace given the corresponding data-LLM functions and an algorithm to predict the aggregate answer y of an unseen query based on the query-LLM functions.

Our algorithms entail the use of the previously trained query-LLM functions from the training query–answer pairs in the training set \mathcal{T} to predict aggregate answers to unseen queries Q1 and Q2 from the test set \mathcal{V} ; see also Fig. 4. We adopt the principle of the nearest neighbors regression for prediction [32]. The notion of neighborhood here is materialized by the overlapping of an unseen query with the query prototypes in the quantized space \mathbb{Q} (see Example 4, Fig. 6). By Definition 7, the queries $\mathbf{q} = [\mathbf{x}, \theta]$ and $\mathbf{q}' = [\mathbf{x}', \theta']$

overlap if the condition $\mathcal{A}(\mathbf{q}, \mathbf{q}') = \text{TRUE}$. To quantify a degree of overlapping between those queries represented as hyper-spheres in the $(d + 1)$ -dim. space, we require that the two spheres are partially intersected. Let us define the ratio between the L_2 distance of the centers of data subspaces $\mathbb{D}(\mathbf{x}, \theta)$ and $\mathbb{D}(\mathbf{x}', \theta')$ over the distance of their 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. We define the *degree of overlapping* for two queries as the normalized ratio $\delta(\mathbf{q}, \mathbf{q}') \in [0, 1]$:

$$\delta(\mathbf{q}, \mathbf{q}') = \begin{cases} 1 - \frac{\max(\|\mathbf{x} - \mathbf{x}'\|_2, |\theta - \theta'|)}{\theta + \theta'}, & \text{if } \mathcal{A}(\mathbf{q}, \mathbf{q}') = \text{TRUE}, \\ 0, & \text{if } \mathcal{A}(\mathbf{q}, \mathbf{q}') = \text{FALSE}. \end{cases} \quad (26)$$

The data subspaces $\mathbb{D}(\mathbf{x}, \theta)$ and $\mathbb{D}(\mathbf{x}_k, \theta_k)$ defined by query \mathbf{q} and query prototype $\mathbf{w}_k = [\mathbf{x}_k, \theta_k]$, respectively, correspond to the highest overlap when $\delta(\mathbf{q}, \mathbf{w}_k) = 1$. We define the *overlapping* query prototypes set $\mathcal{W}(\mathbf{q})$ of query subspaces \mathbb{Q}_k corresponding to data subspaces \mathbb{D}_k given a query $\mathbf{q} = [\mathbf{x}, \theta]$ as:

$$\mathcal{W}(\mathbf{q}) = \{\mathbf{w}_k = [\mathbf{x}_k, \theta_k] : \delta(\mathbf{q}, \mathbf{w}_k) > 0\}. \quad (27)$$

The mean-value query Q1 and the linear regression query Q2 are based on the neighborhood set $\mathcal{W}(\mathbf{q})$ for an unseen query \mathbf{q} .

Example 6 Figure 6 shows the average value and regression query prediction: An unseen query $\mathbf{q} = [\mathbf{x}, \theta]$ is projected onto input space $\mathbf{x} = (x_1, x_2)$ to derive the neighborhood set of prototypes $\mathcal{W}(\mathbf{q}) = \{\mathbf{w}_i, \mathbf{w}_k, \mathbf{w}_l\}$. Then, we access the query-LLM functions f_i, f_k, f_l to predict the aggregate output \hat{y} for query Q1 (see Algorithm 2) and retrieve the data regression planes coefficients \mathcal{S} of the data-LLM functions g_i, g_k, g_l from query-LLM functions f_i, f_k, f_l , respectively, for query Q2 (see Algorithm 3).

6.1 Query Q1: mean-value aggregate prediction

Our algorithm predicts the aggregate output value y given an unseen query $\mathbf{q} = [\mathbf{x}, \theta]$ over a data subspace $\mathbb{D}(\mathbf{x}, \theta)$. The query function f between query \mathbf{q} and answer y over the query space \mathbb{Q} is approximated by K query-LLM functions (hyperplanes) over each query subspace \mathbb{Q}_k ; see Fig. 8(lower). Given a query \mathbf{q} , we derive the overlapping prototypes set $\mathcal{W}(\mathbf{q})$. For those query prototypes $\mathbf{w}_k \in \mathcal{W}(\mathbf{q})$, we access the local coefficients $(y_k, \mathbf{b}_k, \mathbf{w}_k)$ of query-LLM f_k . Then, we pass $\mathbf{q} = [\mathbf{x}, \theta]$ as input to each function f_k to predict the aggregate output \hat{y} through a weighted average based on the normalized degrees of overlapping $\tilde{\delta}(\mathbf{q}, \mathbf{w}_k)$:

$$\tilde{\delta}(\mathbf{q}, \mathbf{w}_k) = \frac{\delta(\mathbf{q}, \mathbf{w}_k)}{\sum_{\mathbf{w}_k \in \mathcal{W}(\mathbf{q})} \delta(\mathbf{q}, \mathbf{w}_k)}. \quad (28)$$

The aggregate output prediction \hat{y} derives from the weighted $\mathcal{W}(\mathbf{q})$ -nearest neighbors regression:

$$\hat{y} = \sum_{\mathbf{w}_k \in \mathcal{W}(\mathbf{q})} \tilde{\delta}(\mathbf{q}, \mathbf{w}_k) f_k(\mathbf{x}, \theta), \quad (29)$$

with

$$f_k(\mathbf{x}, \theta) = y_k + \mathbf{b}_{X,k}(\mathbf{x} - \mathbf{x}_k)^\top + b_{\theta,k}(\theta - \theta_k). \quad (30)$$

In the case where $\mathcal{W}(\mathbf{q}) \equiv \emptyset$, we extrapolate the similarity of the query \mathbf{q} with the closest query prototype to associate the answer with the estimation \hat{y} derived only from the query-LLM function $f_j(\mathbf{q}, \theta)$ with the query prototype \mathbf{w}_j being closest to the query \mathbf{q} . Through this projection, i.e., $j = \arg \min_{k \in [K]} \|\mathbf{q} - \mathbf{w}_k\|_2$, we get the local slope and intercept of the local mapping of query \mathbf{q} onto the aggregate answer y .

The prediction of the query answer depends entirely on the query similarity and the \mathcal{W} neighborhood. The mean-value prediction algorithm is shown in Algorithm 2. Figure 8(lower) shows how accurately the $K = 7$ query-LLM functions (as green covering surfaces/planes over query function f) approximate the linear parts of the query function $f(\mathbf{x}, \theta)$ over a 2D query space \mathbb{Q} defined by the queries (x, θ) .

ALGORITHM 2: Mean-Value Prediction Algorithm (Q1).

Input: unseen query $\mathbf{q} = [\mathbf{x}, \theta]$

Result: average prediction \hat{y} (answer)

begin

 Calculate overlapping set $\mathcal{W}(\mathbf{q})$ using (27);

if $\mathcal{W}(\mathbf{q}) \equiv \emptyset$ **then**

 Find closest prototype from VQ $j = \arg \min_k \|\mathbf{w}_k - \mathbf{q}\|_2$;

 Predict answer $\hat{y} = f_j(\mathbf{q}, \theta)$ using (30);

else

 Calculate normalized overlapping degree $\tilde{\delta}(\mathbf{q}, \mathbf{w}_k)$,

$\mathbf{w}_k \in \mathcal{W}(\mathbf{q})$ using (26) and the engaged query-LLMs;

 Predict answer \hat{y} using (29) and (30);

end

end

6.2 Query Q2: PLR-based data function approximation

The algorithm returns a list of the local data-LLM functions g_k of the underlying data function g over data subspace $\mathbb{D}(\mathbf{x}, \theta)$, given an unseen query $\mathbf{q} = [\mathbf{x}, \theta]$ (see Example 4). An unexplored data subspace \mathbb{D} defined by an unseen query might:

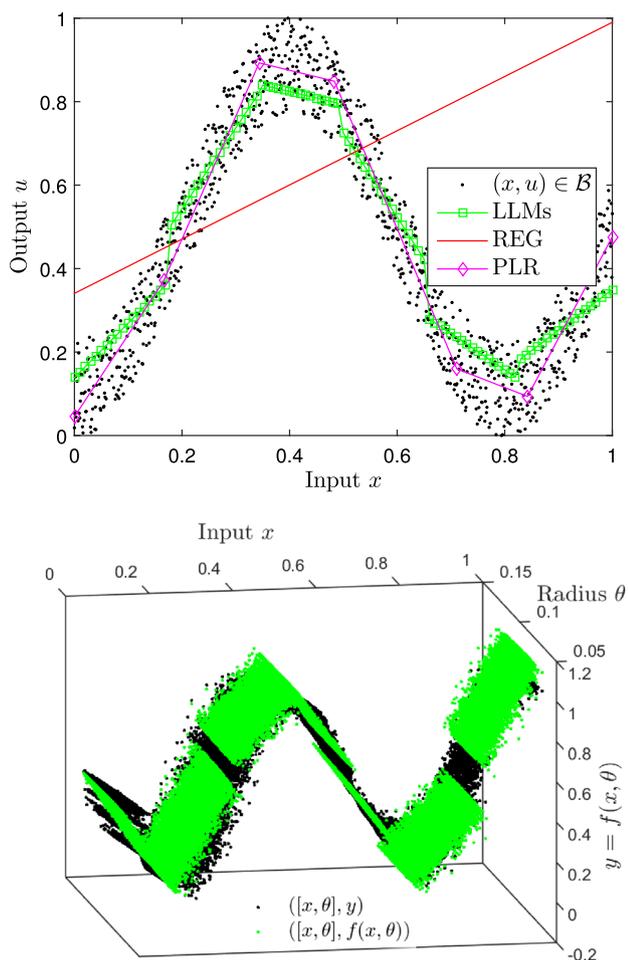


Fig. 8 (Upper) The $K = 6$ data-LLMs $g_k(x) \approx g(x)$, a PLR approximation with $K = 6$ linear models, and a global linear approximation (REG) of $u = g(x)$ over a 2D data subspace \mathbb{D} ; (lower) the $y = f(x, \theta)$ approximated by $K = 7$ query-LLMs $f_k(x, \theta)$ over 3D query space \mathbb{Q}

- (Case 1) either partially overlap with several identified convex data subspaces \mathbb{D}_k (corresponding to query subspaces \mathbb{Q}_k), or
- (Case 2) be contained or contain a data subspace \mathbb{D}_k , or
- (Case 3) be outside of any data subspace \mathbb{D}_k .

In Cases 1 and 2, the algorithm returns the derived data-LLMs of the data function g interpolating over the overlapping data subspaces, using the corresponding query-LLMs, as proved in Theorem 3. In Case 3, the best possible linear approximation of the data function g is returned through the extrapolation of the data subspace \mathbb{D}_k whose query prototype \mathbf{w}_k is closest to the query \mathbf{q} . For Cases 1 and 2, we exploit the neighborhood $\mathcal{W}(\mathbf{q})$ of the query $\mathbf{q} = [\mathbf{x}, \theta]$. For Case 3, we select the data-LLM function, which corresponds to the query-LLM function with the closest query prototype \mathbf{w}_j to query \mathbf{q} since, in this case, $\mathcal{W}(\mathbf{q}) \equiv \emptyset$.

The PLR approximation of the data function $g(\mathbf{x})$ over the data subspace $\mathbb{D}(\mathbf{x}, \theta)$ involves both the radius θ and the query center \mathbf{x} using their similarity with radius θ_k and the point \mathbf{x}_k , respectively, from the $\mathcal{W}(\mathbf{q})$. For the Cases 1 and 2, the set of the data-LLMs for a PLR approximation of data function $g(\mathbf{x})$ is provided directly from those query-LLMs f_k , whose query prototype $\mathbf{w}_k \in \mathcal{W}(\mathbf{q})$. That is, for $\mathbf{x} \in \mathbb{D}_k(\mathbf{x}_k, \theta_k)$, we obtain:

$$u = g(\mathbf{x}) = f_k(\mathbf{x}, \theta_k) \approx y_k + \mathbf{b}_{X,k}(\mathbf{x} - \mathbf{x}_k)^\top, \tag{31}$$

$\forall \mathbf{w}_k \in \mathcal{W}(\mathbf{q})$, where the u intercept in \mathbb{D}_k is: $y_k - \mathbf{b}_{X,k}\mathbf{x}_k^\top$ and the u slope in \mathbb{D}_k is: $\mathbf{b}_{X,k}$.

For the Case 3, the PLR approximation of the data function $g(\mathbf{x})$ derives by extrapolating the linearity trend of $u = g(\mathbf{x}) = f_j(\mathbf{x}, \theta_j) : j = \arg \min_k \|\mathbf{q} - \mathbf{w}_k\|_2$ over the data subspace, with u intercept: $y_j - \mathbf{b}_{X,j}\mathbf{x}_j^\top$ and the u slope: $\mathbf{b}_{X,j}$.

The PLR approximation of the data function is shown in Algorithm 3, which returns the set of the data-LLM functions \mathcal{S} defined over the data subspace $\mathbb{D}(\mathbf{x}, \theta)$ for a given unseen query $\mathbf{q} = [\mathbf{x}, \theta]$. Note that depending on the query radius θ and the overlapping neighborhood set $\mathcal{W}(\mathbf{q})$, we obtain: $1 \leq |\mathcal{S}| \leq K$, where $|\mathcal{S}|$ is the cardinality of the set \mathcal{S} .

Remark 8 Figure 8(upper) shows how the data function $u = g(x)$ is accurately approximated by $K = 6$ data-LLMs (green interpolating local lines) compared with the global linear regression function (REG in red) over the data subspace $\mathbb{D}(0.5, 0.5)$. We also illustrate the K linear models derived by the actual PLR data approximation algorithm [44], i.e., the best possible PLR data approximation should we have access to that data subspace, which corresponds to OPT_K in (3). Unlike our model, PLR needs access to the data and is thus very expensive; specifically, it involves a forward/backward iterative approach to produce the multiple linear models [44]. Our model, instead, incrementally derives the data-LLMs based on the optimization problems in (21) and (22). Note that the derived data-LLMs are highly accurate.

7 Convergence analysis and complexity

7.1 Global convergence analysis

In this section we show that our stochastic joint optimization algorithm is asymptotically stable. Concerning the objective function \mathcal{J} in (21), the query prototypes $\mathbf{w}_k = [\mathbf{x}_k, \theta_k]$ converge to the centroids (mean vectors) of the query subspaces \mathbb{Q}_k . This convergence reflects the partition capability of our proposed AVQ algorithm into the prototypes of the query subspaces. The query subspaces naturally represent the (hyper)spheres of the data subspaces that the analysts are

ALGORITHM 3: PLR Data Approximation (Q2).

```

Input: unseen query  $\mathbf{q} = [\mathbf{x}, \theta]$ 
Result: set  $\mathcal{S}$  of data-LLMs for  $g$  approximation in  $\mathbb{D}(\mathbf{x}, \theta)$ 
begin
   $\mathcal{S} \leftarrow \{\}$ ;
  Calculate overlapping set  $\mathcal{W}(\mathbf{q})$  using (27);
  if  $\mathcal{W}(\mathbf{q}) \equiv \emptyset$  then
    Find closest prototype with  $VQ$   $j = \arg \min_k \|\mathbf{w}_k - \mathbf{q}\|_2$ ;
    Derive data-LLM  $g_j$  from query-LLM  $f_j$ :
     $u = g(\mathbf{x}) = f_j(\mathbf{x}, \theta_j)$ ;
     $\mathcal{S} = \{(y_j - \mathbf{b}_{X,j} \mathbf{x}_j^\top, \mathbf{b}_{X,j})\}$ ;
  else
    foreach  $\mathbf{w}_k \in \mathcal{W}(\mathbf{q})$  do
      Derive data-LLM  $g_k$  from query-LLM  $f_k$ :
       $u = g(\mathbf{x}) = f_k(\mathbf{x}, \theta_k)$ ;
       $\mathcal{S} \leftarrow \mathcal{S} \cup \{(y_k - \mathbf{b}_{X,k} \mathbf{x}_k^\top, \mathbf{b}_{X,k})\}$ ;
    end
  end
end

```

interested in accessed by their query centers \mathbf{x}_k and radii θ_k , $\forall k$.

Concerning the objective function \mathcal{H} in (22), the approximation coefficients slope and intercept in Theorem 3 converge, too. This convergence refers to the linear regression coefficients that would have been derived should we were able to a fit linear regression function over each data subspace \mathbb{D}_k , given that we had access to the data.

Theorem 7 refers to the convergence of a query prototype \mathbf{w}_k to the local expectation query $\mathbb{E}[\mathbf{q}|\mathbb{Q}_k] = \mathbb{E}[\mathbf{q}|v(\mathbf{q}) = k]$ given our AVQ algorithm.

Theorem 7 *If $\mathbb{E}[\mathbf{q}|\mathbb{Q}_k] = \mathbb{E}[\mathbf{q}|v(\mathbf{q}) = k]$ is the local expectation query of the query subspace \mathbb{Q}_k and the query prototype \mathbf{w}_k is the subspace representative from our AVQ algorithm, then $P(\mathbf{w}_k = \mathbb{E}[\mathbf{q}|\mathbb{Q}_k]) = 1$ at equilibrium.*

Proof The update rule for a prototype \mathbf{w}_k based on Theorem 4 is $\Delta \mathbf{w}_k = \eta(\mathbf{q} - \mathbf{w}_k)$, given that $P(\|\mathbf{q} - \mathbf{w}_k\|_2 \leq \rho) = 1$. Let the k -th prototype \mathbf{w}_k reach equilibrium: $\Delta \mathbf{w}_k = \mathbf{0}$, which holds with probability 1. By taking the expectation of both sides we obtain:

$$\begin{aligned} \mathbf{0} &= \mathbb{E}[\Delta \mathbf{w}_k] = \mathbb{E}[(\mathbf{q} - \mathbf{w}_k)] \\ &= \int_{\mathbb{Q}_k} (\mathbf{q} - \mathbf{w}_k) p(\mathbf{q}) d\mathbf{q} \\ &= \int_{\mathbb{Q}_k} \mathbf{q} p(\mathbf{q}) d\mathbf{q} - \mathbf{w}_k \int_{\mathbb{Q}_k} p(\mathbf{q}) d\mathbf{q}. \end{aligned}$$

This indicates that \mathbf{w}_k is constant with probability 1, and then by solving $\mathbb{E}[\Delta \mathbf{w}_k] = \mathbf{0}$, the \mathbf{w}_k equals to the centroid $\mathbb{E}[\mathbf{q}|\mathbb{Q}_k]$. \square

We provide two convergence theorems for the coefficients y_k and \mathbf{b}_k of the query-LLM f_k . Firstly, we focus

on the aggregate answer prediction $y = y_k + \mathbf{b}_k(\mathbf{q} - \mathbf{w}_k)^\top$. Given that the query prototype \mathbf{w}_k has converged, i.e., $\mathbf{w}_k = \mathbb{E}[\mathbf{q}|\mathbb{Q}_k]$ from Theorem 7, then the expected aggregate value $\mathbb{E}[y|\mathbb{Q}_k]$ converges to the y_k coefficient of the query-LLM f_k . This also reflects our assignments of the statistical mapping \mathcal{F} of the local expectation query \mathbf{w}_k to the mean of the query-LLM f_k , i.e., $f_k(\mathbb{E}[\mathbf{x}_k|\mathbb{Q}_k], \mathbb{E}[\theta_k|\mathbb{Q}_k]) = \mathbb{E}[y|\mathbb{Q}_k]$. This refers to the local associative convergence of coefficient y_k given a query $\mathbf{q} \in \mathbb{Q}_k$. In other words, the convergence of the query subspace enforces also convergence in the output domain.

Theorem 8 (Associative Convergence) *If the query prototype \mathbf{w}_k has converged, i.e., $\mathbf{w}_k = \mathbb{E}[\mathbf{q}|\mathbb{Q}_k]$, then the coefficient y_k of the query-LLM f_k converges to the expectation $\mathbb{E}[y|\mathbb{Q}_k]$.*

Proof Based on the law of total expectations, we write the expectation of Δy_k given the output variable y :

$$\mathbb{E}[\Delta y_k] = \int_{\mathbb{R}} \mathbb{E}[\Delta y_k | y] p(y) dy.$$

By using the update rule in Theorem 4, we write for the conditional expectation term $\mathbb{E}[\Delta y_k | y]$ as:

$$\begin{aligned} \mathbb{E}[\Delta y_k | y] &= \mathbb{E}[y - y_k - \mathbf{b}_k(\mathbf{q} - \mathbf{w}_k)^\top | y] \\ &= \mathbb{E}[y | y] - y_k + \mathbf{b}_k \mathbf{w}_k^\top - \mathbf{b}_k \mathbb{E}[\mathbf{q}^\top | y]. \end{aligned}$$

By replacing $\mathbb{E}[\Delta y_k | y]$ into $\mathbb{E}[\Delta y_k]$, we obtain

$$\begin{aligned} \mathbb{E}[\Delta y_k] &= \int_{\mathbb{R}} \mathbb{E}[y | y] p(y) dy - y_k + \mathbf{b}_k \mathbf{w}_k^\top \\ &\quad - \mathbf{b}_k \int_{\mathbb{R}} \mathbb{E}[\mathbf{q}^\top | y] p(y) dy = \mathbb{E}[y|\mathbb{Q}_k] - y_k \end{aligned}$$

given that $\mathbb{E}[\mathbf{q}|\mathbb{Q}_k] = \mathbf{w}_k$ from Theorem 7. By solving $\mathbb{E}[\Delta y_k] = 0$, which implies that y_k is constant with probability 1, we obtain that $y_k = \mathbb{E}[y|\mathbb{Q}_k]$. \square

Finally, we provide a convergence theorem for \mathbf{b}_k as the slope of the linear regression of $\mathbf{q} - \mathbf{w}_k$ onto $y - y_k$.

Theorem 9 *Let*

$$\boldsymbol{\beta}_k = [\mathbb{E}[(\mathbf{q} - \mathbf{w}_k)^\top (\mathbf{q} - \mathbf{w}_k)]]^{-1} \mathbb{E}[(y - y_k)(\mathbf{q} - \mathbf{w}_k)]$$

be the linear regression population coefficient of all pairs $(\mathbf{q} - \mathbf{w}_k, y - y_k)$ for a LLM function $y = y_k + \mathbf{b}_k(\mathbf{q} - \mathbf{w}_k)^\top$. Then $P(\mathbf{b}_k = \boldsymbol{\beta}_k) = 1$ at equilibrium.

Proof Based on law of total expectations, for the LLM coefficient \mathbf{b}_k we obtain $\mathbb{E}[\Delta \mathbf{b}_k] = \int_{\mathbb{R}} \mathbb{E}[\Delta \mathbf{b}_k | y] p(y) dy$. By using the update rule in Theorem 4, we write for the conditional expectation term $\mathbb{E}[\Delta \mathbf{b}_k | y]$:

$$\mathbb{E}[\Delta \mathbf{b}_k | y] = \int_{\mathbb{Q}_k} [(y - y_k) - \mathbf{b}_k(\mathbf{q} - \mathbf{w}_k)^\top](\mathbf{q} - \mathbf{w}_k) p(\mathbf{q} | y) d\mathbf{q}$$

$$\begin{aligned}
 &= \int_{\mathbb{Q}_k} (y - y_k)(\mathbf{q} - \mathbf{w}_k) p(\mathbf{q}|y) d\mathbf{q} \\
 &\quad - \mathbf{b}_k \int_{\mathbb{Q}_k} (\mathbf{q} - \mathbf{w}_k)^\top (\mathbf{q} - \mathbf{w}_k) p(\mathbf{q}|y) d\mathbf{q} \\
 &= \mathbb{E}[(y - y_k)(\mathbf{q} - \mathbf{w}_k)|y] \\
 &\quad - \mathbb{E}[(\mathbf{q} - \mathbf{w}_k)^\top (\mathbf{q} - \mathbf{w}_k)|y].
 \end{aligned}$$

Hence, by replacing $\mathbb{E}[\Delta \mathbf{b}_k|y]$ into $\mathbb{E}[\Delta \mathbf{b}_k]$, we obtain

$$\mathbb{E}[\Delta \mathbf{b}_k] = \mathbb{E}[(y - y_k)(\mathbf{q} - \mathbf{w}_k)] - \mathbb{E}[(\mathbf{q} - \mathbf{w}_k)^\top (\mathbf{q} - \mathbf{w}_k)]$$

By solving $\mathbb{E}[\Delta \mathbf{b}_k] = \mathbf{0}$, which implies that \mathbf{b}_k is constant with probability 1, we obtain that $\mathbf{b}_k = \beta_k$. This refers to the population normal equations for the multivariate linear regression model within the subspace $\mathbb{Q}_k \times \mathbb{R}$. \square

7.2 Partial convergence analysis

The entire statistical learning model runs in two phases: the training phase and the prediction phase. In the *training phase*, the query-LLM prototypes $(\mathbf{w}_k, \mathbf{b}_k, y_k), k \in [K]$, are updated upon the observation of a query–answer pair (\mathbf{q}, y) until their convergence w.r.t. the *global* stopping criterion in (24). In the *prediction phase*, the model proceeds with the mean-value prediction of the aggregate answer \hat{y} , the PLR data approximation of the data function g and the output data value prediction \hat{u} , without execution of any incoming query after convergence at t^* . The major requirement for the model to transit from the training to the prediction phase is the triggering of the global stopping criterion at t^* w.r.t. a fixed $\gamma > 0$ convergence threshold.

Let us now provide an insight of this global criterion. The model convergence means that on average for *all* the trained query-LLM prototypes their improvement w.r.t. a new incoming query–answer pair is not as much significant as it was at the early stage of the training phase. The rate of updating such prototypes, which is reflected by the difference vector norms of $(\mathbf{w}_{k,t}, \mathbf{b}_{k,t}, y_{k,t})$ and $(\mathbf{w}_{k,t-1}, \mathbf{b}_{k,t-1}, y_{k,t-1})$ at observation t and $t - 1$, respectively, is decreasing as the number of query–answer pairs increases, i.e., $t \rightarrow \infty$; refer also to convergence analysis in Sect. 7.

In a real world setting, however, we cannot obtain an infinite number of training pairs to ensure convergence. Instead, we are sequentially provided a finite number of training pairs (\mathbf{q}_t, y_t) from a finite training set \mathcal{T} . We obtain model convergence given that there are enough pairs in the set \mathcal{T} such that the criterion in (24) is satisfied. More interestingly, we have observed that *some* of the query-LLM prototypes, say $L < K$ converge with less training query–answer pairs than all provides pairs $|\mathcal{T}|$. Specifically, for those L prototypes, which represent certain data subspaces \mathbb{D}_ℓ and query subspaces $\mathbb{Q}_\ell, \ell = 1, \dots, L$, it holds true that the convergence criterion $\max\{\Gamma_\ell^{\mathcal{J}}, \Gamma_\ell^{\mathcal{H}}\}_t \leq \gamma$ for $t < t^*$, where t^* cor-

responds to the last observed training pair where the entire model has *globally* converged, given a fixed γ convergence threshold. In this case, we introduce the concept of *partial* convergence if there is at least a subset of query-LLM prototypes, which have already converged w.r.t. γ at an earlier stage than the entire model (entire set of parameters). Interestingly, those ℓ query-LLM prototypes transit from their training phase to the prediction phase. The partial convergence on those data subspaces is due to the fact that there were relatively more queries issued to those data subspaces compared to some other data subspaces up to the t -th observation with $t < t^*$. Moreover, by construction of our model, only a relatively small subset of query-LLM prototypes are required for mean-value prediction and PLR data approximations (refer to the overlapping set \mathcal{W} in Sect. 6). Hence, based on the flexibility of the partial convergence, we can proceed with prediction and data approximation to certain incoming queries issued onto those data subspaces, where their corresponding query-LLM prototypes have partially converged, while the entire model is still on a training phase, i.e., it has not yet globally converged.

The advantage of this methodology is that we deliver predicted answers to the analysts’ queries without imposing the execution delay for those queries. Evidently, we obtain the flexibility to either proceed with the query execution *after the prediction* for refining more the converged data subspace or not. In both options, the analysts ‘do not need to wait’ for the system to execute firstly the query and then being delivered the answers. This motivated us to introduce a *progressive predictive analytics or intermediate* phase, where some parts of the model can, after their local convergence, provide *predicted* answers to the analysts without waiting for the entire model to converge.

The research challenge in supporting the progressive analytics phase is when some of the involved query-LLM parameters are not yet converged with some other query-LLM parameter, which have locally converged. Specifically, assume that at the t -th observation (with $t < t^*$) there are L query-LLM prototypes that have converged and the query $\mathbf{q}_t = (\mathbf{x}_t, \theta_t)$ is arriving to the system (note: $L < K$ at observation t). The overlapping set $\mathcal{W}(\mathbf{q}_t)$ consists of $\ell \leq L$ query prototypes $\mathbf{w}_i, i = 1, \dots, \ell$, which have converged and $\kappa < K - L$ query prototypes $\mathbf{w}_j, j = 1, \dots, \kappa$, which have not yet converged, i.e., $\mathcal{W}(\mathbf{q}_t) = \{\mathbf{w}_i\} \cup \{\mathbf{w}_j\}$. In this case, the mean-value prediction and the PLR data approximation over the data subspace $\mathbb{D}(\mathbf{x}_t, \theta_t)$ involves $\ell + \kappa$ prototypes such that:

$$\begin{aligned}
 \mathcal{C}(\mathbf{q}_t) &= \{\mathbf{w}_i \in \mathcal{W}(\mathbf{q}_t) : \max\{\Gamma_i^{\mathcal{J}}, \Gamma_i^{\mathcal{H}}\}_t \leq \gamma\} \\
 \mathcal{U}(\mathbf{q}_t) &= \{\mathbf{w}_j \in \mathcal{W}(\mathbf{q}_t) : \max\{\Gamma_j^{\mathcal{J}}, \Gamma_j^{\mathcal{H}}\}_t > \gamma\}
 \end{aligned} \tag{32}$$

with $\ell = |\mathcal{C}(\mathbf{q}_t)|$ and $\kappa = |\mathcal{U}(\mathbf{q}_t)|$. We adopt a convergence voting/consensus scheme for supporting this intermediate

phase between training and prediction phase in light of delivering either predicted answers or actual answers to the analysts.

- *Case A* If the consensual ratio $\frac{\ell}{\ell+\kappa} \geq r$, i.e., more than $r\%$ of the query prototypes in $\mathcal{W}(\mathbf{q}_t)$ have locally converged, with $r \in (0.5, 1)$ then two options are available:
 - *Case A.I* The model *predicts* and *delivers* the answer based *only* on those ℓ query prototypes which have converged to the analysts and, then, *executes* the query for updating the κ not yet converged query prototypes to align with the model convergence mode. In this case, the analysts are delivered a *predicted answer* where the degree of confidence for this answer is regulated through the consensual ratio r . The mean-value prediction and PLR data approximation is achieved as described in Algorithms 2 and 3 by replacing $\mathcal{W}(\mathbf{q})$ with the locally converged query prototypes $\mathcal{C}(\mathbf{q})$ in (32). After the query execution, the query-LLM prototypes from the un-converged set $\mathcal{U}(\mathbf{q})$ in (32) are updated as described in Algorithm 1. Obviously, if the consensual ratio $\frac{\ell}{\ell+\kappa} = 1$, then there is no such an intermediate phase.
 - *Case A.II* The model *predicts* and *delivers* the answer based *only* on those ℓ query prototypes which have converged, to the analysts, and *does not execute* the query, thus, no update is performed for those κ query prototypes. The mean-value prediction and PLR data approximation is achieved as described in Algorithm 2 and Algorithm 3 by replacing $\mathcal{W}(\mathbf{q})$ with the locally converged query prototypes $\mathcal{C}(\mathbf{q})$ in (32). This obviously delays the global convergence and reduces the number of queries executed for convergence. This option is only preferable when most of the incoming queries *focus on specific data subspaces* and *not* on the entire data space. In other words, there is no meaning for the *entire* model to globally converge to transit from the training phase to the prediction phase, if most of the queries are issued on very specific data subspaces. At the extreme case, the model could delay a lot its convergence if more than 50% of the query prototypes are involved in the overlapping sets for *all* the incoming queries. To alleviate this case, our model creates new prototypes (incrementally) only when there is at least some interest on a specific data subspace, as discussed in Sect. 5 adopting the principles of adaptive resonance theory [30].
- *Case B* Otherwise, i.e., the consensual ratio $\frac{\ell}{\ell+\kappa} < r$, the model acts as usual in the training phase, i.e., it first *executes* the query and *delivers* the actual answer to the

analyst, and then based on this actual answer it updates the prototypes as discussed in Sect. 5.

Algorithm 4 shows the partial convergence methodology of the model transition from the training phase to the intermediate phase, and then to the prediction phase.

ALGORITHM 4: Partial Convergence Algorithm.

Input: convergence threshold γ ; consensual threshold r

Result: query-LLM parameters and query prototypes of set α

begin

Get first query–answer pair (\mathbf{q}, y) ;

Init.: $\alpha = \{(y_1 = 0, \mathbf{b}_1 = \mathbf{0}, \mathbf{w}_1 = \mathbf{q})\}$, $K \leftarrow 1$;

repeat

Observe only the query \mathbf{q} ;

Calculate local criteria $\Gamma_i^{\mathcal{J}}, \Gamma_i^{\mathcal{H}}, i = 1, \dots, L$;

Calculate overlapping set $\mathcal{W}(\mathbf{q}) \equiv \mathcal{C}(\mathbf{q}) \cup \mathcal{U}(\mathbf{q})$;

Derive ℓ converged and κ un-converged query-LLM prototypes, respectively, from $\mathcal{W}(\mathbf{q})$;

if $\frac{\ell}{\ell+\kappa} \geq r$ **then**

Call prediction Algorithm (3) or Algorithm (4) replacing $\mathcal{W}(\mathbf{q})$ with $\mathcal{C}(\mathbf{q})$;

else

Execute query and obtain query–answer pair (\mathbf{q}, y) ;

Call training Algorithm (1);

Calculate $\Gamma^{\mathcal{J}}, \Gamma^{\mathcal{H}}$;

end

until $\max(\Gamma^{\mathcal{J}}, \Gamma^{\mathcal{H}}) \leq \gamma$;

end

Our progressive predictive analytics methodology allows the combined mode of operation, whereby the training and prediction phases overlap. In this combined mode, the model runs its training and prediction algorithms based on the consensual threshold r . Let us define t_{\top} the first observation at which the consensual ratio $\frac{\ell}{\ell+\kappa}$ exceeds threshold r , i.e.,

$$t_{\top} = \arg \min \{t > 0 : \frac{\ell_t}{\kappa_t + \ell_t} \geq r, \mathcal{C}_t \cup \mathcal{U}_t \equiv \mathcal{W}(\mathbf{q}_t)\}. \quad (33)$$

For any observation $t < t_{\top}$ the model is in a single training phase, while for any observation $t_{\top} \leq t < t^*$ the model is in the intermediate phase, i.e., prediction and/or training phase depending on the consensual ratio at the t -th observation (Cases A and/or B). At $t > t^*$ the model transits to the single prediction phase. Figure 9 illustrates the activation of the training, intermediate and prediction phases over the observation time axis and the landmarks t_{\top} and t^* . The landmark t_{\top} denotes the minimum number of training pairs the model requires to deliver to the analysts predicted and/or actual answers w.r.t. Cases A and B, while only predicted answers are delivered after t^* training pairs.

Remark 9 The prediction performance of the model in the intermediate phase is up to the performance of the model in

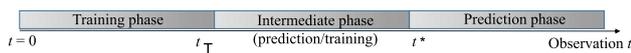


Fig. 9 The landmarks t_T and t^* for model transition from the training to the intermediate phase, and from the intermediate to the prediction phase, respectively

the single prediction phase. This is attributed to the predicted answers based on the partial convergence w.r.t. consensual threshold r , where only $r\%$ of the query-LLM prototypes from the overlapping set $\mathcal{W}(\mathbf{q})$ are used for prediction given an unseen query \mathbf{q} . The prediction performance is a non-decreasing function with the number of observations t with $t_T \leq t \leq t^*$ as will be shown in our performance evaluation Sect. 8.

7.3 Computational complexity

In this section we report on the computational complexity of our model during the training and prediction phases. In the global convergence mode, the model ‘waits’ for the triggering of the criterion in (24) to transit from the training to the prediction phase. Under SGD over the objective minimization functions \mathcal{J} and \mathcal{H} , with the hyperbolic learning schedule in (7), our model requires $O(1/\gamma)$ [15] number of training pairs to reach the convergence threshold γ . This means that the residual difference between the objective function value \mathcal{J}^{t^*} after t^* pairs and the optimal value \mathcal{J}^* , i.e., with the optimal query-LLM parameters, asymptotically decreases exponentially, also known as *linear convergence* [25]. In this mode, there is a clear separation between the training and prediction phases, while the upper bound of the expected excess difference $\mathbb{E}[\mathcal{J}^t - \mathcal{J}^*]$ after t training pairs is $O\left(\sqrt{\frac{\log t}{t}}\right)$ [52], given a hyperbolic learning schedule in (7).

In the prediction phase, which is the operational mode of our model, given a mean-value query Q1 and a linear regression query Q2, we require $O(dK)$ to calculate the neighborhood \mathcal{W} set and deliver the query-LLM functions, respectively, i.e., independent on the data size, thus, achieving scalability. We also require $O(dK)$ space to store the query prototypes and the query-LLM coefficients. The derivation of the data-LLMs is then $O(1)$ given than we have identified the query-LLMs for a given linear regression query.

8 Performance evaluation

8.1 Performance metrics

The proposed methodology deals with two major statistical learning components: *prediction* of the aggregate answer and data output, and data function *approximation* over data

subspaces. For evaluating the performance of our model in light of these components, we should assess the model **predictability** and **goodness of fit**, respectively.

Predictability refers to the capability of a model to *predict* an output given an unseen input, i.e., such input–output pair is not provided during the model’s training phase. Measures of prediction focus on the differences between *values predicted* and *values actually* observed. Goodness of fit describes how well a model *fits* a set of observations, which were provided in the model’s training phase. It provides an understating on how well the selected independent variables (input) explain the variability in the dependent (output) variable. Measures of goodness of fit summarize the discrepancy between actual/observed values during training and the *values approximated* under the model in question.

We compare our statistical methodology against its ground truth counterparts: the multivariate linear regression model over data subspaces, hereafter referred to as REG, and the piecewise linear model (PLR) over data subspaces, both of which have *full access* to the data. Note that the PLR data approximation is the optimal multiple linear modeling over data subspaces we can obtain because it is constructed by accessing the data. Hence, we demonstrate how effectively our data-LLMs *approximate* the ground truth data function g and the optimal PLR data approximation. Specifically, we compare against the REG model using DMS PostgreSQL and the MATLAB and the PLR model using the ARESLab (MATLAB) toolbox⁷ for building PLR models based on the multivariate adaptive regression splines method in [44]. We show that our model is scalable and efficient and as (or even more than) accurate than the REG model, w.r.t. predictability and goodness of fit, and close to the accuracy obtained by the optimal PLR model. Our model is dramatically more scalable and efficient as, unlike REG and PLR models, it does not need access to data, yielding up to six orders of magnitude faster query execution.

8.1.1 Predictability

Predictability in the query space The Mean-Value Accuracy (A1 metric) refers to the answer prediction of the average value \hat{y} given an unseen Q1 query $\mathbf{q} = [\mathbf{x}, \theta]$. Based on the EPE in (5), the A1 metric is the root-mean-square error (RMSE):

$$e = \left(\frac{1}{M} \sum_{i=1}^M (y_i - \hat{y}_i)^2 \right)^{1/2} \tag{34}$$

⁷ <http://www.cs.rtu.lv/jekabsons/>.

where $y = f(\mathbf{x}, \theta)$ and \hat{y} is the actual and the predicted average value of data output u , respectively, from Algorithm 2 given M unseen Q1 queries.

Predictability in the data space The data output accuracy (A2 metric) refers to the prediction of the data output $u = g(\mathbf{x})$ given an unseen input $\mathbf{x} \in \mathbb{R}^d$. Here, query-LLM functions are exploited to predict the data output u by approximating the data function $g(\mathbf{x})$ as in (31) by aggregation of neighboring query-LLMs $f_k(\mathbf{x}, \theta_k)$, i.e., the PLR-based data-LLMs $g_k(\mathbf{x})$. Let u and \hat{u} be the actual and the predicted data output value of $g(\mathbf{x})$ given M unseen points \mathbf{x} . Based on (29) and (31) we predict \hat{u} as:

$$\begin{aligned}\hat{u} &= \sum_{\mathbf{w}_k \in \mathcal{W}(\mathbf{q})} \tilde{\delta}(\mathbf{q}, \mathbf{w}_k) f_k(\mathbf{x}, \theta_k) \\ &= \sum_{\mathbf{w}_k \in \mathcal{W}(\mathbf{q})} \tilde{\delta}(\mathbf{q}, \mathbf{w}_k) g_k(\mathbf{x}),\end{aligned}\quad (35)$$

where $\mathcal{W}(\mathbf{q})$ is the overlapping set for query \mathbf{q} defined in (27). Note that the query-LLM function $f_k(\mathbf{x}, \theta_k)$ provides the intercept y_k and slope $\mathbf{b}_{X,k}$ over the data input space by setting the radius $\theta = \theta_k$ in the function f_k . For a given data input \mathbf{x} , the A2 metric is the RMSE of the predicted output \hat{u} over M unseen inputs:

$$v = \left(\frac{1}{M} \sum_{i=1}^M (u_i - \hat{u}_i)^2 \right)^{1/2}. \quad (36)$$

8.1.2 Goodness of fit

PLR approximation in data space Given an unseen Q2 query $\mathbf{q} = [\mathbf{x}, \theta]$ defined over the data subspace $\mathbb{D}(\mathbf{x}, \theta)$, we evaluate how well our methodology approximates the data function g through data-LLM functions comparing with the REG model and the optimal PLR data approximation model over the same data subspace \mathbb{D} . For *goodness of fit* we adopt the metrics: Fraction of Variance Unexplained (FVU) s and Coefficient of Determination (CoD) R^2 [26]. FVU indicates the fraction of variance of the dependent data output variable u , which cannot be explained, i.e., which is not correctly predicted by the explanatory data input variable \mathbf{x} . Given a data subspace $\mathbb{D}(\mathbf{x}, \theta)$, consider the data pairs $(\mathbf{x}_i, u_i) : \mathbf{x}_i \in \mathbb{D}$, $i \in [n_\theta(\mathbf{x})]$, with outputs $u_i = g(\mathbf{x}_i)$, and approximations \hat{u}_i for each input \mathbf{x}_i . The sum of squared residuals (SSR) and the total sum of squares (TSS) over $\mathbb{D}(\mathbf{x}, \theta)$ are then defined as:

$$\begin{aligned}SSR &= \sum_{i \in [n_\theta(\mathbf{x})]} (u_i - \hat{u}_i)^2, \\ TSS &= \sum_{i \in [n_\theta(\mathbf{x})]} (u_i - \bar{u})^2,\end{aligned}\quad (37)$$

respectively, where \bar{u} is the average output value:

$$\bar{u} = \frac{1}{n_\theta(\mathbf{x})} \sum_{i \in [n_\theta(\mathbf{x})]} u_i. \quad (38)$$

The FVU and CoD are then defined as:

$$s = \frac{SSR}{TSS} \text{ and } R^2 = 1 - s, \quad (39)$$

respectively. The FVU metric indicates how closely the approximation of the data function g over a data subspace \mathbb{D} matches the actual data function g over that data subspace. If the FVU value is greater than 1, the explanatory input variable \mathbf{x} does not convey any information about the output u in the sense that the predictions \hat{u} do not covary with the actual output u . In this case, the data approximation function is a *bad* fit. The approximation is considered *good* when the FVU metric assumes a low value less than 1. Given an unseen query \mathbf{q} over the data subspace $\mathbb{D}(\mathbf{x}, \theta)$, we measure the FVU and CoD metrics for the REG and PLR models, and the average FVU value $s = \frac{1}{|\mathcal{S}|} \sum_{\ell=1}^{|\mathcal{S}|} s_\ell$ of the FVUs s_ℓ (and CoDs) corresponding to the set of data-LLM functions $\mathcal{S} : |\mathcal{S}| \geq 1$ derived by our Algorithm 3. In our experimental evaluation and comparative assessment of our model with the PLR and REG models (pair-wise), in each performance metric, we adopted the paired-sample two-tailed Student t test using a 95% confidence interval, i.e., significance level 0.05.

8.2 Experimental setup

8.2.1 Real and synthetic datasets

Our goal is to evaluate accuracy in terms of predictability and goodness of fit, efficiency and scalability over real and synthetic datasets. For accuracy, using the A1, A2, FVU, and CoD metrics, we intentionally sought multivariate real data functions g that exhibit extreme nonlinearity in many data subspaces. For this reason, to assess the A1 metric for Q1 queries, the A2 metric for data output predictions, and the FVU, CoD metrics for Q2 queries, we used two real datasets R1 from [45] and R3 from [16], and a synthetic dataset referred to as R2.

Real datasets The real dataset R1 consists of 6-dim. feature vectors corresponding to the concentration level of 6 gases, namely, Ethanol (E1), Ethylene (E2), Ammonia (A1), Acetaldehyde (A2), Acetone (A3), and Toluene (T) derived from chemical sensors. The sensors measurements of the dataset R1 were gathered within 36 months in a gas delivery platform facility situated at the ChemoSignals Laboratory in the BioCircuits Institute (BCI⁸), University of California.

⁸ <http://biocircuits.ucsd.edu/>.

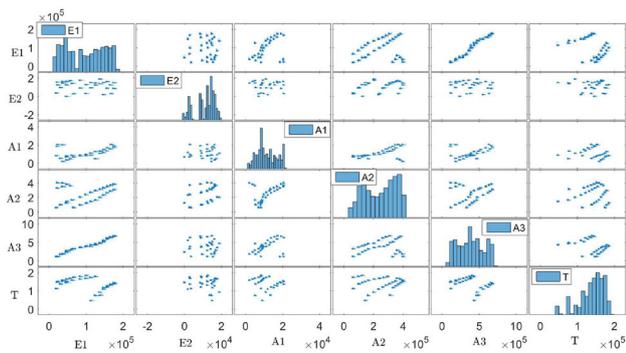


Fig. 10 R1 Dataset Scatter Plot Matrix: Each cell plots the concentration level of gas X against gas Y, with: E1:Ethanol, E2:Ethylene, A1:Ammonia, A2:Acetaldehyde, A3:Acetone, and T:Toluene; the diagonal plots the histogram of each gas concentration

We expand the R1 size by adding extra 6-dim. vectors with Gaussian noise, thus, in total the R1 dataset contains $15 \cdot 10^6$ multi-dimensional data vectors of gases concentration levels. With the R1 dataset we wished to delve into accuracy issues and this dataset was chosen because its data exhibits nonlinear relationships among features. All d -dim. real-valued vectors are scaled and normalized in $[0,1]$ ($d \in \{1, \dots, 6\}$) with significant nonlinear dependencies among the features, evidenced by a high FVU = 4.68. This indicates that a linear approximation of the entire data space is definitely to no avail, presenting a challenging dataset for our approach. Figure 10 shows the R1 scatter plot matrix for all gases concentrations (before scaling and normalization) depicting the dependencies between gases and the corresponding histograms of each dimension. We obtain significant correlations among many gases, indicatively E1 with A1, A2 and A3 with Pearson correlation coefficient 0.41, 0.23, and 0.98 ($p < 0.05$), respectively, and E2 with A2 having correlation 0.36 ($p < 0.05$). By further analyzing the R1 dataset, the first three Principal Components (PCs) explain the 99.73% of the total variance by 73.57%, 23.94%, and 2.22%, respectively, which are used for Q1 and Q2 analytics queries (prediction of the mean-value and model fitting).

The R3 real dataset contains environmental sensed data used for data-driven predictive models for the energy use of appliances [16]. The data include measurements of temperature and humidity sensors from a wireless network in a house located in Stambruges, Belgium. The sensors read contextual data every 10 min for about 4.5 months. The house temperature and humidity conditions were monitored with an in-house ZigBee wireless sensor network built with XBee radios, Atmega328P micro-controllers and DHT-22 sensors. The digital DHT-22 sensors have an accuracy of ± 0.5 Celsius for temperature and $\pm 1\%$ for relative humidity. The environmental parameters are: temperature in kitchen area (T1), humidity in kitchen area (H1), temperature in living room area (T2), humidity in living room area (H2), temperature in

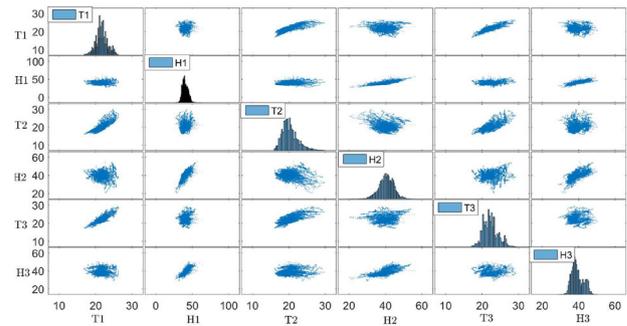


Fig. 11 R3 Dataset Scatter Plot Matrix: Each cell plots the contextual/environmental parameter level of dimension X against dimension Y, with: temperature in kitchen (T1), humidity in kitchen (H1), temperature in living room (T2), humidity in living room (H2), temperature in laundry room (T3), and humidity in laundry room area (H3); the diagonal plots the histogram of each environmental parameter

laundry room area (T3), and humidity in laundry room area (H3). The real dataset R3 consists of 6-dim. feature vectors corresponding to the above-mentioned six contextual parameters. We expand the R3 size by adding extra 6-dim. vectors with Gaussian noise, thus, in total the R3 dataset contains $10 \cdot 10^6$ multi-dimensional data vectors of temperature and relative humidity of different areas within the house. All d -dim. real-valued vectors are scaled and normalized in $[0,1]$ ($d \in \{1, \dots, 6\}$) with significant nonlinear dependencies among the features, evidenced by a FVU = 7.32 indicating that a single linear approximation of the entire data space is not an option. Figure 11 shows the R3 scatter plot matrix for all dimensions (before scaling and normalization) along with their dependencies and histograms. The first four Principal Components (PCs) explain the 99.08% of the total variance by 67.82%, 19.60%, 9.29%, and 2.37%, respectively, used for Q1 and Q2 analytics queries (prediction of the mean-value and model fitting).

Synthetic dataset To further evaluate scalability and efficiency along with accuracy, we now use a *big* synthetic dataset deriving from a benchmark function to ensure also significant nonlinearity. The R2 synthetic dataset of input–output pairs (u, \mathbf{x}) contains 10^{10} d -dim. real data generated by the Rosenbrock function [46] $u = g(\mathbf{x})$ and $d \in \{1, \dots, 6\}$. This is the popular benchmark function for testing nonlinear, gradient-based optimization algorithms. It has a global minimum inside a long, narrow, parabolic shaped flat valley, where convergence to the global minimum, however, is extremely non-trivial [46]. We obtain the Rosenbrock $u = g(\mathbf{x}) = \sum_{i=1}^{d-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$, $\mathbf{x} = [x_1, \dots, x_d]$, attribute domain $|x_i| \leq 10$ and global minimum is 0 at $x_i = 1, \forall d$. Obviously, there is no linear dependency among features in the data space evidenced by a FVU = 12.45. In addition, we generate 10^{10} vectors adding noise $\epsilon \sim \mathcal{N}(0, 1)$ to each dimension. For illustration purposes, Fig. 12 shows the R2 dataset of the Rosenbrock function $u = g(x_1, x_2)$

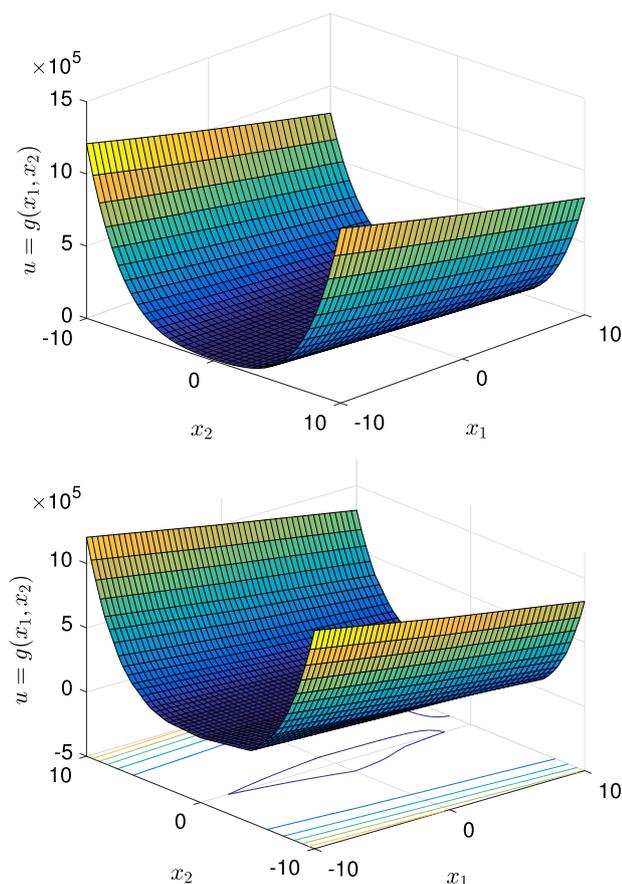


Fig. 12 (Upper) R2 Synthetic Dataset of the Rosenbrock function $u = g(x_1, x_2)$ with two ($d = 2$) variables x_1 and x_2 ; (lower) the PLR approximation of the Rosenbrock function ($d = 2$) through $K = 23$ LLMs and the corresponding contour plot

with two ($d = 2$) variables x_1 and x_2 and its corresponding PLR approximation through $K = 23$ LLMs.

System implementation The real datasets R1 and R3 and the synthetic dataset R2 are stored in a PostgreSQL server with 2x Intel Xeon E5645, RAM 96 GB, HD: Seagate Constellation 1TB, 32MB cache. We use R1, R2, and R3 to assess the query Q1 prediction accuracy of the aggregate answer y (A1 metric), corresponding to the average of the dimensions of the first three and four PCs in R1 and in R3, respectively, and to the average data output u of the Rosenbrock in R2. The PLR approximation of the data function g regarding Q2 queries over the R1, R2, and R3 datasets is conducted over data dimensions $d \in \{2, 3, 5, 6\}$ for the metrics: FVU, CoD and data output prediction accuracy metric A2. For scalability and efficiency, our method compared against the PostgreSQL with a B-tree index on input vector \mathbf{x} ($d \in \{2, 5, 6\}$ over Q1 queries, $d = 2$ over Q2 queries) and MATLAB ($d = 5$ and $d = 6$) over Q2 queries using the `regress` function on the server and the ARESLab tool for PLR data approximation.

8.2.2 Query workloads, training and testing sets

Query workload Firstly, we generate certain query workloads to train and test our model. The random queries $\mathbf{q} = [\mathbf{x}, \theta]$ with centers \mathbf{x} and radii θ over the data subspaces are generated with uniformly distributed centers $\mathbf{x} \in [0, 1]^d$ for the R1 and R3 datasets and in $[-10, 10]^d$ for the R2 dataset (recall that the data vectors in R1 and R3 are scaled and normalized in $[0, 1]$). That is, the query centers can uniformly at random appear over all the data space defined by the domains of the datasets dimensions $d \in \{1, \dots, 6\}$. The query radius θ affects the training time and the prediction quality (both in predictability and goodness of fit). In brief, a larger (smaller) θ implies shorter (longer) training times as will be elaborated later. For each query, the radius $\theta \sim \mathcal{N}(\mu_\theta, \sigma_\theta^2)$ is generated from a Gaussian distribution with mean μ_θ , variance σ_θ^2 . We set random radius $\theta \sim \mathcal{N}(0.1, 0.01)$ for the R1 and R3 datasets and $\theta \sim \mathcal{N}(1, 0.25)$ for the R2 dataset, covering $\sim 20\%$ in each feature data range; the justification for this setting is discussed later. Section 8.7 provides an extensive experimental and theoretical analysis of the impact of θ on the model performance. Based on this set up, we generated random queries \mathbf{q} that are issued over the data spaces of the R1, R2 and R3 datasets. We use these queries for training and testing our models as follows.

Training and testing sets We describe how we generate the training and testing query–answer sets from the above-mentioned query workload methodology. To train our model, we generate training files \mathcal{T} consisting of random queries \mathbf{q} as described above along with their actual aggregate answers y after executing them. To test the performance of our models, we generate different testing files \mathcal{V} dedicated only for predictions containing random queries of various sizes: $|\mathcal{T}| \in \{10^3, \dots, 10^4\}$ and $M = |\mathcal{V}| \in \{10^3, \dots, 2 \cdot 10^4\}$, respectively. Specifically, the training sets \mathcal{T} and testing sets \mathcal{V} contain pairs of queries and answers, i.e., (\mathbf{q}, y) , where the queries were executed over the R1, R2, and R3 datasets (see also Fig. 4). We adopted the cross-validation technique [51] to evaluate all the predictive models by partitioning the original query–answer sets into a training set to train the models, and a test set to evaluate them. We use 10-fold cross-validation, where the original query–answer set is randomly partitioned into 10 equal size subsets. Of the 10 subsets, a single subset is retained as the validation dataset for testing the models, and the remaining 9 subsamples are used as training data. The cross-validation process is then repeated 10 times (the folds), with each of the 10 subsamples used exactly once as the validation data. The 10 results from the folds are then be averaged to produce a single estimation of the above-mentioned performance metrics.

8.3 Model training and convergence

We train our model with the training set \mathcal{T} and then evaluate and compare it with the ground truths REG (ProstgreSQL and MATLAB) and PLR (MATLAB) with the testing set \mathcal{V} examining the statistical significance in accuracy with respect to paired-sample two-tailed Student t test with significance level 0.05. Note, the \mathcal{T} and \mathcal{V} sets contain explicitly different queries as discussed in Sect. 8.2.2. The granularity of quantization for our model is tuned by the percentage coefficient $a \in [0.05, 1]$, involved in vigilance parameter $\rho = a(d^{1/2} + 1)$ (see Sect. 5 and Remark 7). Specifically, a value of quantization coefficient $a = 1$ corresponds to the generation of only one prototype, i.e., $K = 1$ (that is, coarse quantization), while any value of coefficient $a < 1$ (that is, fine grained quantization) corresponds to a variable number of prototypes $K > 1$ depending on the underlying (unknown) data distribution. The default value for the quantization percentage coefficient is $a = 0.25$ in our experiments. The model is adapting its parameters/prototypes in a stochastic manner every time a new query–answer pair $(\mathbf{q}, y) \in \mathcal{T}$ is present. We set model convergence threshold $\gamma = 0.01$ in Algorithm 1 to transit from the training phase to the prediction phase. Moreover, the hyperbolic learning rate schedule for the t -th training pair is: $\eta_t = (1 + t)^{-1}$ [14] for the stochastic training of the prototypes as query–answer pairs (\mathbf{q}, y) are retrieved (one at a time) from the training set \mathcal{T} . Notably, to fairly compare against the optimal PLR data approximation, we set its maximum numbers of the automatically discovered linear models (in the forward building phase of the PLR algorithm) equal to K and the generalized cross-validation penalty per PLR knot to 3 as suggested in [44]. The proposed statistical methodology requires training and prediction phases. We also introduce the intermediate phase in Sect. 7.2, which is controlled by the consensual threshold $r = 0.7$ of the partially converged query prototypes involved in queries. That is the model starts providing predictions in the intermediate phase when 70% of the query prototypes have converged. We examine firstly the global convergence of the training phase of our model and, then, study the variant of partial convergence w.r.t. the landmarks t_{\top} and t^* and the impact on the required number of training pairs and accuracy. Table 1 shows the experimental parameters and their range/default values.

Figure 13 examines the termination criterion of the training Algorithm 1 $\Gamma = \max(\Gamma^{\mathcal{J}}, \Gamma^{\mathcal{H}})$ against the number of training pairs (\mathbf{q}, y) in training set \mathcal{T} for $d \in \{2, 5\}$ over R1 and R2 datasets with quantization coefficient $a = 0.25$; similar results are obtained from R3 dataset. The training phase terminates at the first instance t^* when $\Gamma \leq \gamma$, which is obtained for $|\mathcal{T}| \approx 5300$ training pairs. The total average training time, which includes both Q1 execution time and model updates time, is (0.41, 0.36, 2.38) h for R1, R3,

Table 1 Experimental Parameters

| Parameters | Range/value |
|---------------------------------------|--|
| Data dimensionality d | {2, 3, 5, 6} |
| Real dataset R1 [45] | $15 \cdot 10^6$ vectors in $[0, 1]^d$ |
| Synthetic dataset R2 | 10^{10} Rosenbrock in $[-10, 10]^d$ |
| Real dataset R3 [16] | $10 \cdot 10^6$ vectors in $[0, 1]^d$ |
| Vigilance coefficient a | [0.05, 1] |
| Consensual threshold r | 0.7 |
| Convergence threshold γ | 0.01 |
| Training dataset size $ \mathcal{T} $ | $[10^3, \dots, 10^4]$ |
| Testing dataset size $ \mathcal{V} $ | $[10^3, \dots, 2 \cdot 10^4]$ |
| Initial learning rate η_0 | 0.5 [14] |
| Query center/point | Uniform vectors in $[0, 1]^d$ |
| Query radius θ | Gaussian values $\mathcal{N}(\mu_{\theta}, \sigma_{\theta}^2)$ |
| Query mean radius μ_{θ} | [0.01, 0.99] |
| Query radius dev. σ_{θ} | 0.01 |

and R2 datasets, respectively. This should not be perceived as overhead of our approach, as 99.62% of the training time is devoted to executing the queries over the DMS/statistical system, which we cannot avoid that anyway even in the typical case as shown in Fig. 4. Any traditional approach would thus also pay 99.62% of this cost. This only affects how early our approach switches to using the trained model versus executing the queries against the system. Our experiments show that excellent quality results can be produced using a reasonable number of past executed queries for training purposes. Obviously, this can be tuned by setting different model convergence threshold γ values. We set $\gamma = 0.01$, where Γ is (stochastically) trapped around 0.0046, with deviation 0.0023 in R1 and 0.0012 in R3. In R2, the Γ is strictly less than γ for $|\mathcal{T}| > 5300$.

Figure 14 shows the relation between the percentage of the number of the training pairs $|\mathcal{T}|%$ used for a specific percentage of query prototypes to partially converge given the intermediate phase for $d \in \{2, 5\}$ over R1 dataset with quantization coefficient $a = 0.25$. Specifically, we observe that with only 35% of the training pairs, i.e., with almost 1800 query–answer pairs (landmark $t_{\top} \approx 1800$), we obtain a model convergence of the 70–80% of the query-LLM prototypes. This indicates that the entire model has partially converged to a great portion w.r.t. number of query-LLM prototypes requiring a relatively small number of training pairs. In this case, the intermediate phase is deemed of high importance for delivering predictions to the analysts while the model is still being in a ‘quasi-training’ mode. The model converges with a high rate as more training pairs from the training set \mathcal{T} are observed after the convergence of the 70% of the query-LLM prototypes. This suggests to set the consensual threshold for the intermediate phase $r = 0.7$. However,

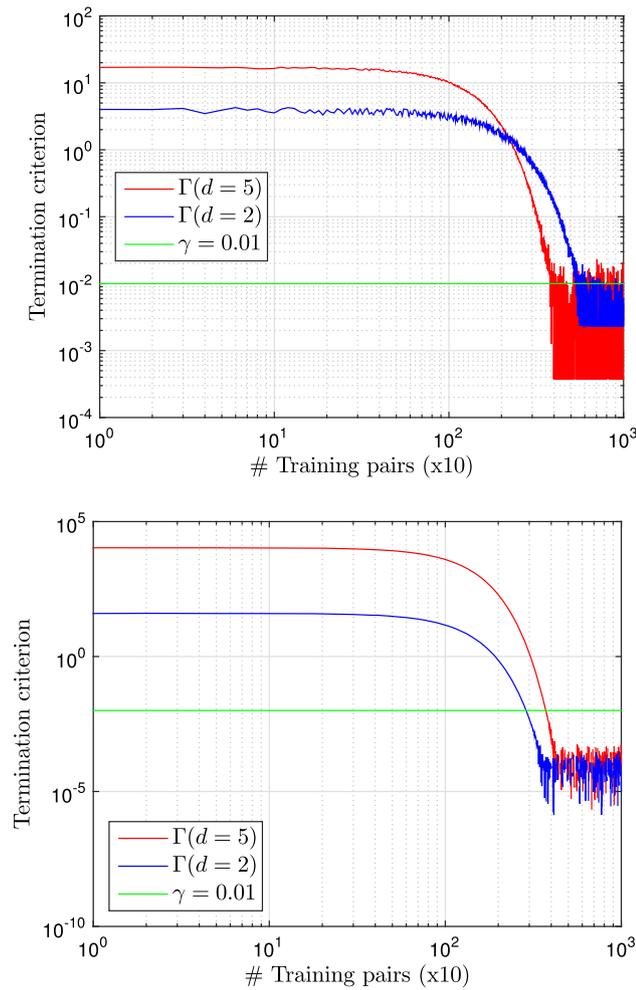


Fig. 13 Learning termination criterion $\Gamma = \max(\Gamma^{\mathcal{J}}, \Gamma^{\mathcal{H}})$ of Algorithm 1 versus number of training pairs $|\mathcal{T}|$ for (upper) R1 and (lower) R2; $d \in \{2, 5\}$

during this phase, the delivered predictions to the analysts have to be assessed w.r.t. prediction accuracy, as will be discussed in Sect. 8.4.

Figure 15(upper) shows the evolution of the joint objective functions \mathcal{J} and \mathcal{H} and Fig. 15(lower) shows the evolution of the individual norm difference of each query prototype from the *current average*, i.e., the individual convergence criterion, against the percentage of training pairs. We observe that after 37% of training pairs of the training set \mathcal{T} , all the prototypes start to transit from their training phase to the prediction phase, while minimizing their deviations from the average convergence trend of the model. This indicates the flexibility of our model in being in the prediction phase thus proceeding with prediction for those data subspaces where the corresponding query prototypes have already converged and, in the same time, being in the training phase for those query prototypes which have not yet converged, thus, keeping in a learning mode. After the global convergence, i.e., when the

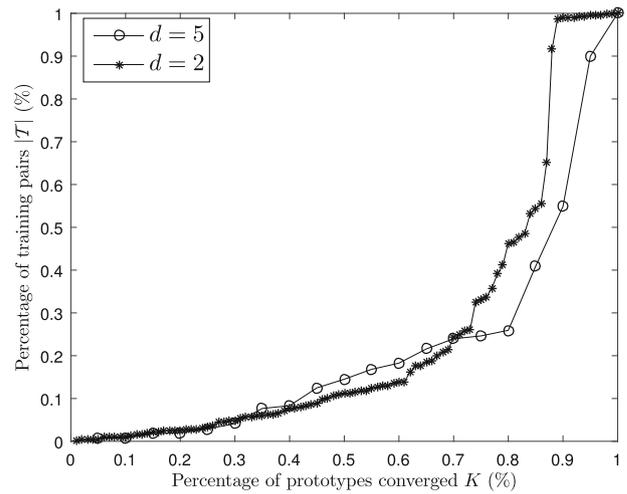


Fig. 14 Relation between the percentage of the training pairs $|\mathcal{T}|$ % used for a specific percentage of query prototypes K % to partially converge given the intermediate phase for $d \in \{2, 5\}$

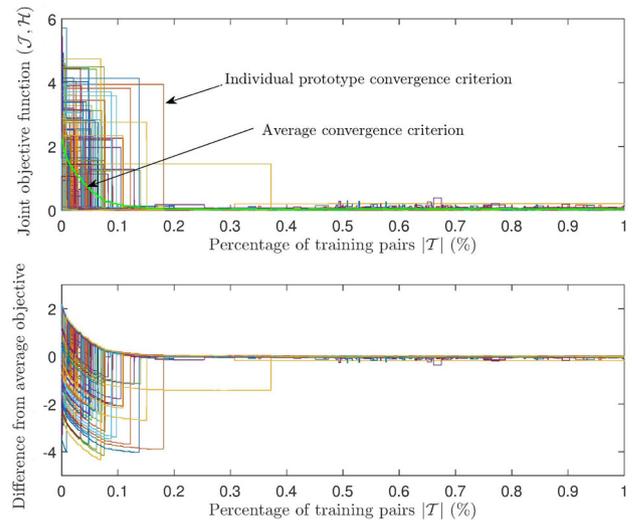


Fig. 15 (Upper) Evolution of the joint objective functions \mathcal{J} and \mathcal{H} and (lower) evolution of the difference of the individual convergence criterion per prototype against the percentage of number of training pairs $|\mathcal{T}|$; dataset R1; $a = 0.25$

consensual ratio reaches the unity, the model transits entirely in the prediction phase thus achieving significantly fast query execution without accessing the data. This signifies the scalability of our query-driven approach.

8.4 Evaluation of Q1 query: predictability and scalability

Figures 16 and 18(upper) show the RMSE e of the predicted answer y (A1 metric) against the resolution of quantization (coefficient) a over R1, R2, and R3 datasets, respectively, for Q1 queries using the generated testing set \mathcal{V} (see Sect. 8.2.2).

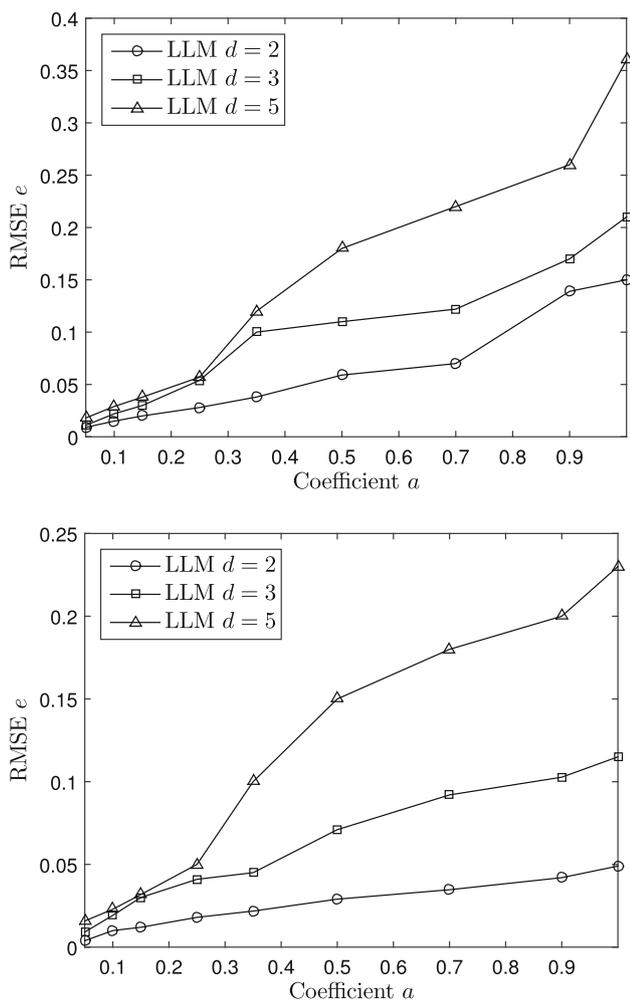


Fig. 16 Q1: RMSE of *y* of LLM versus coefficient *a* over (upper) R2 and (lower) R1; $d \in \{2, 3, 5\}$

For different quantization coefficient *a* values, our model identifies subspaces where the function $f(\mathbf{x}, \theta)$ behaves almost linearly, thus, the query-LLM functions approximate such regions with high accuracy. Interestingly, by quantizing the query space by adopting a small coefficient *a* value, i.e., fine-grained resolution of the query space, then high accuracy is achieved (low RMSE *e* values) with obvious non-linear dependencies among dimensions. This is due to the fact that with small coefficient *a* values, we focus on very specific query subspaces where linear approximations suffice to approximate the query function *f*. This expects to result in a high number of prototypes *K* in order to build many query-LLM functions to capture all the possible nonlinearities of the query function *f* as will be discussed below.

Figure 23(lower) shows the number of query prototypes formed during the query space quantization. Indicatively, we obtain $K = 450$ prototypes for quantization coefficient $a = 0.25$. This indicates that only 450 prototypes are required to accurately predict the aggregate answer *y* for data

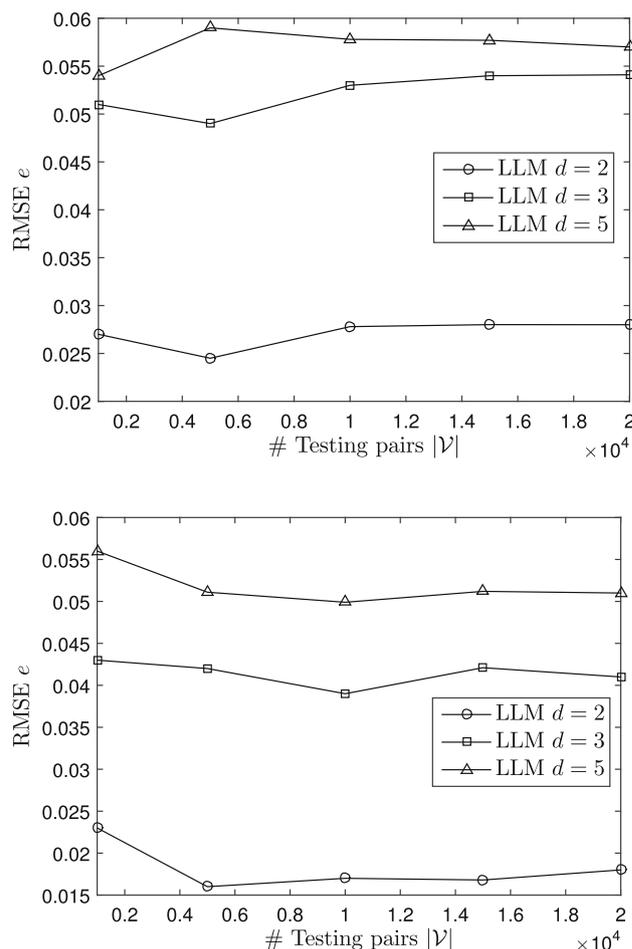


Fig. 17 Q1: RMSE of *y* of LLM versus number of testing pairs ($|\mathcal{V}|$ size) over (upper) R2 and (lower) R1; $d \in \{2, 3, 5\}$, $a = 0.25$

dimension $d = 5$, that is, it is required 450 query-LLM functions to capture the curvature of the query function *f* over the query subspaces. As the quantization coefficient $a \rightarrow 1$, then we quantize the query function *f* into fewer query-LLMs approximations, thus, yielding higher RMSE values as expected due to coarse approximation of the function *f*.

Figures 17 and 18(lower) show the robustness of the our model w.r.t. predictability with various testing file sizes $|\mathcal{V}|$ for R1, R2, and R3 datasets, respectively. Once the LLM model has converged, it provides a low and constant prediction error in terms of RMSE for different data dimensions *d*, indicating the robustness of the training and convergence phase of the proposed model. This means that the model after transiting into the prediction phase can accurately predict the aggregate answer *y* via the identified and optimized query-LLM functions thus no query processing and data access is needed at that phase.

To assess the efficiency and scalability for the mean-value prediction query Q1, Fig. 24(upper) shows in log scale the average Q1 execution time over the dataset R2 for LLM (with

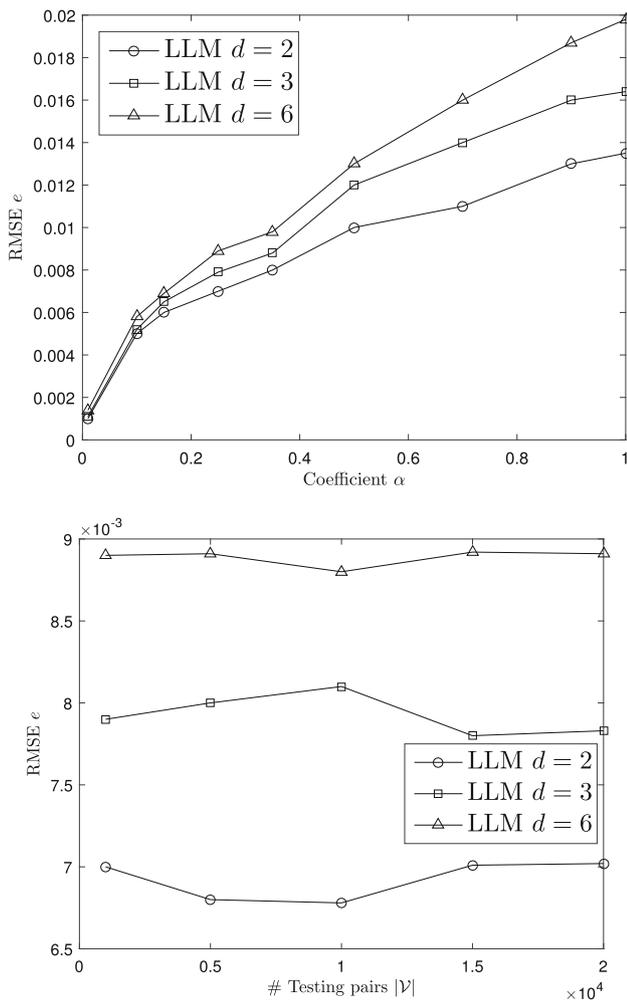


Fig. 18 Q1: RMSE of y of LLM versus (upper) coefficient α over R3 and (lower) number of testing pairs ($|\mathcal{V}|$ size) over R3; $d \in \{2, 3, 6\}$, $a = 0.25$

quantization coefficient $a = 0.25$) corresponding to $K = 92$ and $K = 450$ query prototypes for dimensions $d \in \{2, 5\}$, respectively. Our method requires just 0.18 ms per query over massive data spaces in its prediction phase, offering up to five orders of magnitude speedup (0.18 ms vs up to 10^5 ms/query). This is expected, since the LLM-based model predicts the Q1’s outputs and does not execute the query over the data during prediction achieving high prediction accuracy.

We now examine the impact of the model partial convergence on the predictability, i.e., when the model is in the intermediate phase between the training and the prediction phases. Figure 19 (upper) shows the *partial* RMSE \tilde{e} of the predicted aggregate answer y (A1 metric) during the intermediate phase of the model and the achieved RMSE e during the prediction phase against the percentage of training pairs for consensual threshold $r = 0.7$ over dimension $d \in \{2, 5\}$ for the dataset R1. Similar results are obtained from R2 and R3 datasets. Specifically, the partial RMSE \tilde{e} is obtained only

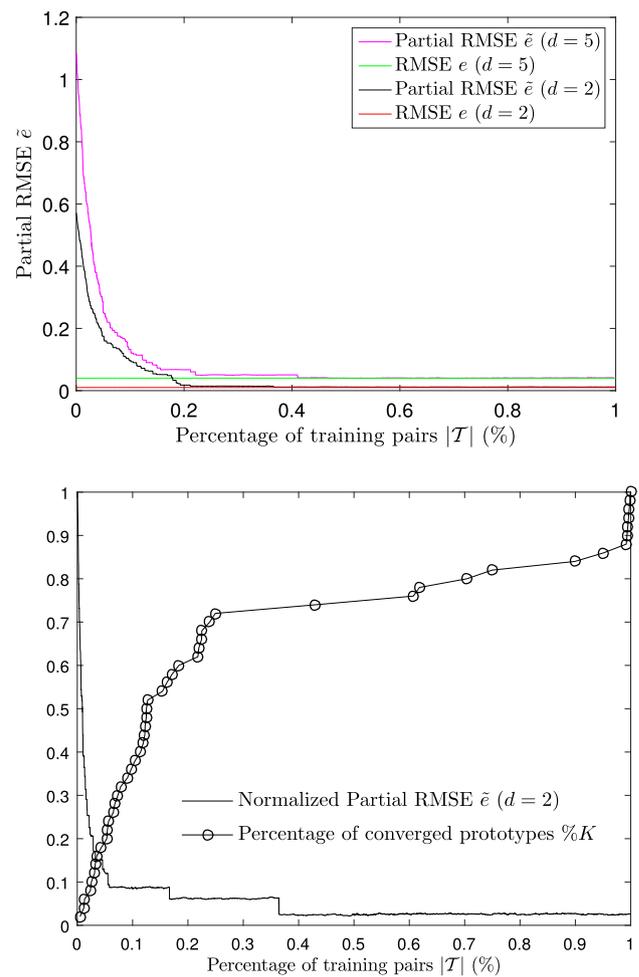


Fig. 19 (Upper) *partial* RMSE \tilde{e} (during intermediate phase) and achieved RMSE e (during the prediction phase) against the percentage of training pairs for consensual threshold $r = 0.7$; (lower) normalized partial RMSE \tilde{e} in $[0, 1]$ against the percentages of converged prototypes and training pairs; $d \in \{2, 5\}$ over dataset R1

from the converged query prototypes during the intermediate phase as described in Case A.I in Sect. 7.2 for $r = 0.7$. That is, from those query prototypes whose any additional training pair (\mathbf{q}, y) does not significantly *move* the query prototypes in the query space. We observe the predictability capability of our model w.r.t. number of training pairs such that with almost 35% for $d = 2$ (and 45% for $d = 5$) of the observed training pairs, the model achieves a RMSE value close to the RMSE value obtained in the fully prediction phase, i.e., after observing 100% of the observed training pairs from \mathcal{T} . This indicates the flexibility of our model to proceed with accurate predictions even being in the intermediate phase, where some of the query prototypes are still in a training mode until the model entirely converges.

More interestingly, Fig. 19(lower) shows the efficiency of our model in achieving high prediction accuracy even during the intermediate phase describe above. The model being

in the intermediate phase can provide RMSE values close to that at the end of the training phase by having 70% of the prototypes converged after observing 37% of the training pairs from the training set \mathcal{T} . This demonstrates the fast convergence of the model and its immediate application for delivering predictions to the analysts and real-time predictive analytics applications while not yet being fully converged.

Remark 10 The RMSE in Fig. 19(lower) is normalized in $[0,1]$ for comparison reasons with the percentages of converged prototypes and training pairs.

8.5 Evaluation of Q2 query: PLR data approximation and scalability

We evaluate the Q2 queries by using our query-/data-LLMs model against the REG and PLR models and show the statistical significance of the derived accuracy metrics. The explanation over the linear/nonlinear behaviors of data function g is interpreted by the variance explanation and model fitting metrics fraction of the variance unexplained FVU and coefficient of determination CoD against the quantization resolution coefficient a and the model prototypes K . Figures 20 and 21(upper) show the sum of squared residuals SSR between the actual answers and the predicted answers for the data-LLMs and REG model with $d \in \{2, 5, 6\}$ over the datasets R1, R2, and R3 with $p < 0.05$.

Figures 22(upper) and 21(lower) show that the fraction of the variance unexplained of the function approximation FVU < 1 for our model, while for the REG model we obtain FVU > 1 , $p < 0.05$. This indicates the capability of our model to capture the nonlinearities of the underlying data function g over all the data subspaces, compared with the REG model provided in the modern DMS. Specifically, as the query space quantization is getting coarse, that is a low resolution with $a \rightarrow 1$, our model approaches the fraction of unexplained variances FVU of the model fitting as that of the REG model, i.e., resulting to a few number of data-LLM functions. This is because, we enforce our model to generate fewer data-LLM functions, thus, cannot effectively capture the nonlinearities of the data function g over all possible data subspaces. Indicatively, we obtain only one data-LLM when $a = 1$, i.e., only a global linear model approximates the data function g . As expected, the optimal PLR model achieves the lowest FVU by capturing the nonlinearity of the data function g with multiple linear basis functions. For quantization coefficient $a < 1$, we achieve a low FVU and our model captures effectively the nonlinearity of data function g by autonomously deriving multiple data-LLMs, which is very close to the actual PLR approximation for quantization coefficient $a < 0.1$ with $p < 0.05$. As the Rosenbrock function g is nonlinear, our model attempts to analyze it into data-LLMs and provide a fine-grained explanation on the behavior of the data function

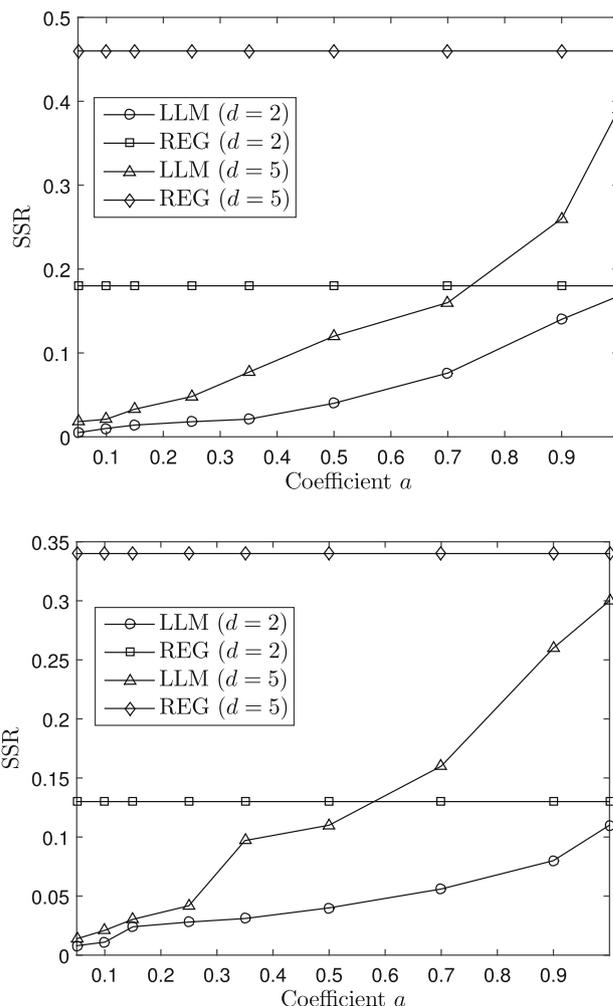


Fig. 20 Q2: SSR of REG and LLM versus coefficient a over (upper) R2 and (lower) R1; $d \in \{2, 5\}$

g . This cannot be achieved by the in-DMS model REG, since the Rosenbrock function cannot be expressed by a ‘global’ line within the entire data space $\mathbb{D}(\mathbf{x}, \theta)$. The PLR model shows statistically superior FVU performance ($p < 0.05$), while being dramatically inefficient compared to our model, as shown in Fig. 24(lower). On the other hand, our model conditionally quantizes the data function g into data-LLMs, thus, providing the list \mathcal{S} of local lines that significantly better explain the data subspace, without accessing the data and then providing high accurate model fitting.

In Fig. 22(lower) and in Fig. 21(lower) the data function g in R1 and R3 datasets does not behave linearly in all the random data subspaces. This is evidenced by the FVU metric of the REG model, which is relatively close to/over 1 for $d = 2$, $d = 5$, and $d = 6$ with $p < 0.05$. This information is unknown a priori to analysts, hence the results using the REG model would be fraught with approximation errors indicating ‘bad’ model fitting. It is worth noting that, the average

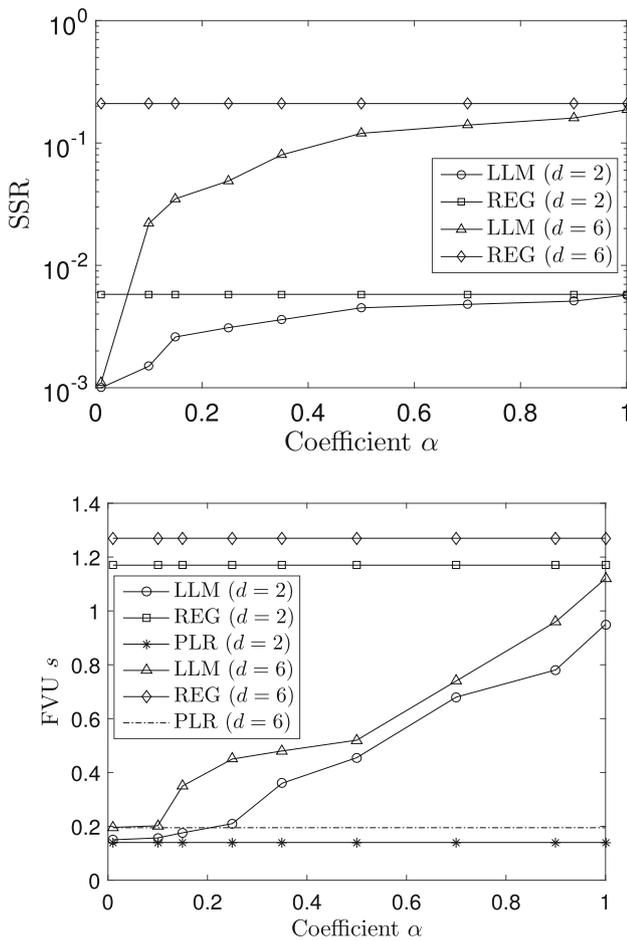


Fig. 21 Q2: (Upper) SSR of REG and LLM versus coefficient a over R3; (lower) FVU_s of REG, PLR, and LLM versus coefficient a over R3, $d \in \{2, 6\}$

number of data-LLM functions that are returned to the analysts for all the issued testing queries in the testing set \mathcal{V} is $|\mathcal{S}| = 4.62$ per query with variance 3.88. This denotes the nonlinearity behavior of data function g and the fine-grained and accurate explanation of the function g within a specific data subspace $\mathbb{D}(\mathbf{x}, \theta)$ per query $\mathbf{q} = [\mathbf{x}, \theta]$. Here, the PLR model achieves the lowest FVU value, i.e., best model fitting as expected ($p < 0.05$), but note that this is also achieved by our data-LLM functions with a quantization coefficient $a < 0.1$.

Figure 23(upper) shows the coefficient of determination CoD R^2 for the LLM, REG, and PLR models over the R1 dataset (similar results are obtained for datasets R2 and R3) having a significance level of 5%. A positive value of R^2 close to 1 depicts that a linear approximation is a good fit for the unknown data function g . While, a value of R^2 close to 0, and especially, a negative value of R^2 indicates a significantly *bad* fit signaling inaccuracies in function approximation. In our case, with $K > 60$ query prototypes, our model achieves high and positive R^2 indicating that our model better explains

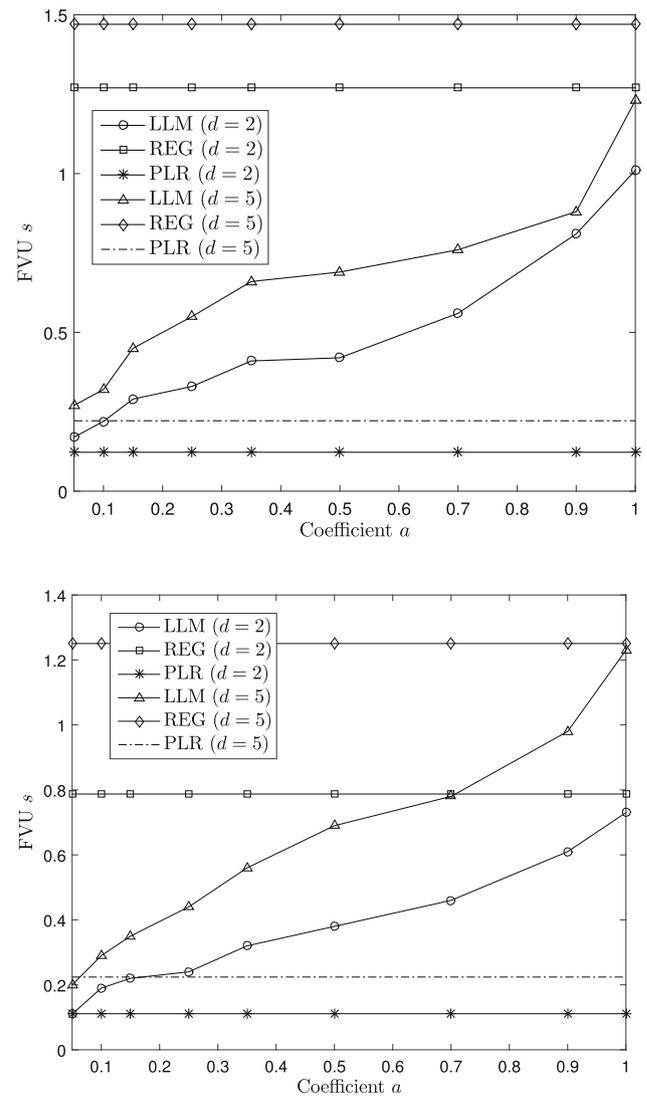


Fig. 22 Q2: FVU_s of REG, PLR, and LLM versus coefficient a over (upper) R2 and (lower) R1; $d \in \{2, 5\}$

the random queried data subspaces $\mathbb{D}(\mathbf{x}, \theta)$ compared with the obtained explanation of the current in-DMS REG model over exactly the same data subspaces and $p < 0.05$. The REG model achieves low R^2 values, including negative ones, thus it is inappropriate for predictions and function approximation. This indicates that the underlying data function g highly exhibits nonlinearities, which are not known to the analysts a-priori. By adopting our model, the analysts progressively learn the underlying data function g and also via the derived data-LLM functions capture the hidden nonlinearities over the queried data subspaces. Such subspaces could never be known to the analysts unless exhaustive data access and exploration takes place. This capability is only provided by our model. Notably, as the quantization coefficient $a \rightarrow 1$, our model increases significantly the coefficient of variation R^2 value indicating a better capture of the specificities of the

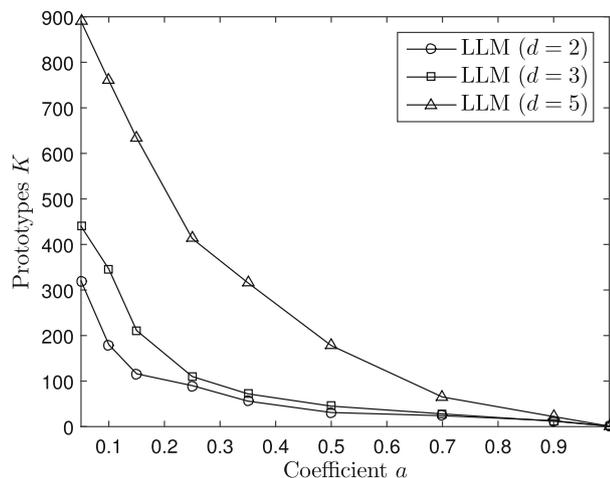
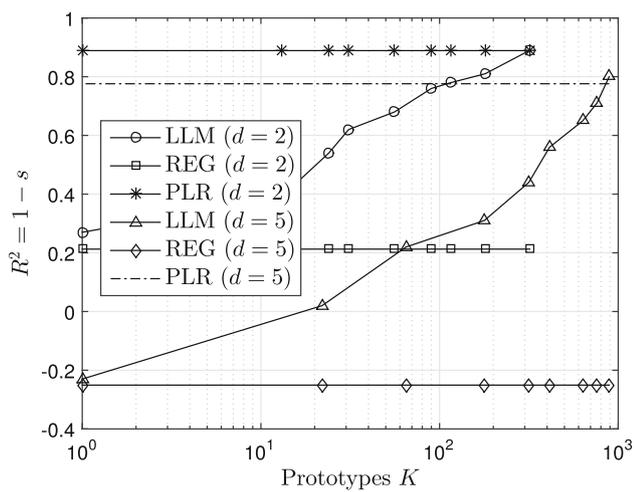


Fig. 23 Q2: (Upper) CoD R^2 of LLM, PLR, and REG versus prototypes K for R1; (lower) Prototypes K versus coefficient a over R1; $d \in \{2, 5\}$

underlying data function g , thus, providing more accurate linear models. Again, the data-access exhaustive PLR model achieves the highest CoD values, however at the cost of high insufficiency; see Fig. 24(lower). Regardless, note that our model can catch the PLR’s CoD value by simply increasing K , i.e., the granularity of query space quantization.

Figures 24(lower) and 26(lower) show the Q2 execution time over the dataset R2 and R3, respectively, for data-LLM ($a = 0.25$, i.e., $K = (92, 450)$ for $d = (2, 5)$) through Algorithm 3, the REG model from PostgreSQL ($d = 2$) (REG-DBMS), the REG model from MATLAB ($d = 5$) (REG-MATLAB), and the optimal PLR against dataset size. The derived results are statistically significant with $p < 0.05$. Our model is highly scalable (note the flat curves) in both datasets and highly efficient, achieving 0.56 ms/query and 0.78 ms/query (even for massive datasets)—up to six orders of magnitude better than the REG and PLR models for R2 and R3 datasets, respectively. The full picture is then that our model provides ultimate scalability by being independent of

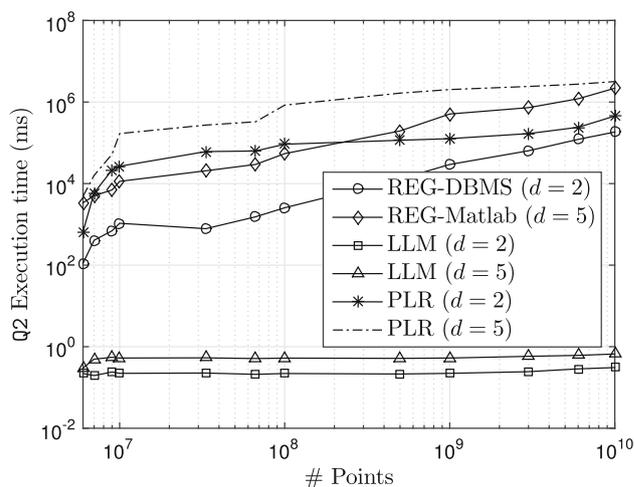
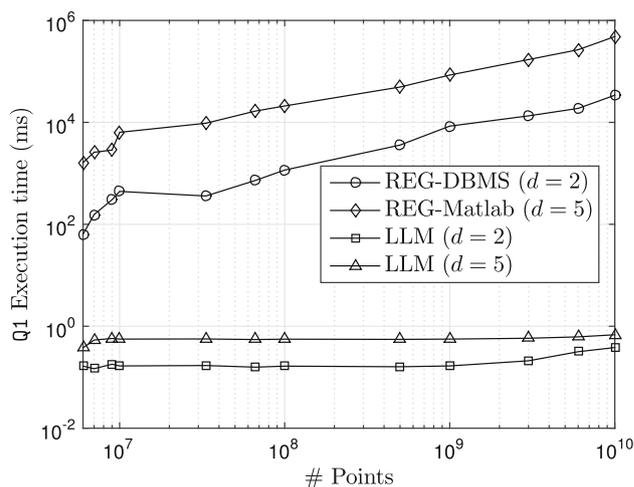


Fig. 24 Query execution time (ms) versus # points for (upper) Q1 and (lower) Q2 for LLM, PLR, and REG over R2; $d \in \{2, 5\}$

the size of the dataset) and many orders of magnitude higher efficiency, while it ensures great goodness of fit (CoD,FVU), similar to that of PLR.

The PLR model with data sampling techniques could also be considered as an effective efficiency-accuracy trade-off. Figure 24, however, shows the efficiency limitations of such an approach. The PLR model, even over a very small random sample of size $10^6 = 0.01\%$ of the 10^{10} dataset, is shown to be > 3 orders of magnitude less efficient than our model. Also, recall that PLR here is implemented over MATLAB, with all data in memory, hiding the performance costs of a full in-DBMS implementation for the selection operator (computing the data subspace); the sampling of the data space; and the PLR algorithm (whose performance is shown in Fig. 24). All of this is in stark contrast to the $O(1)$ cost of our model. Finally, note that, to our knowledge, PLR is not currently implemented within DMSs, regardless of its cost.

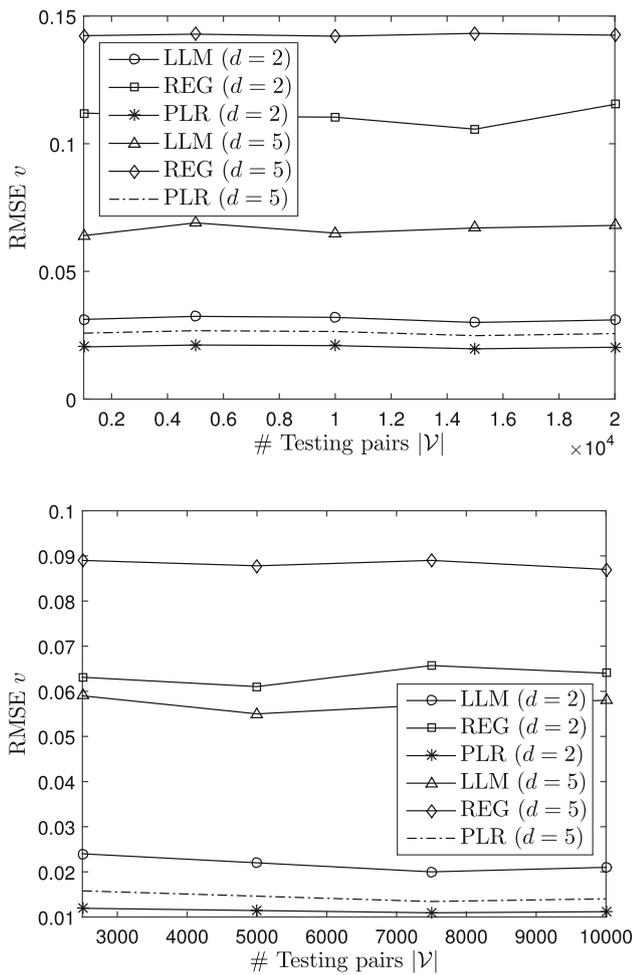


Fig. 25 Q2: RMSE v for data output u of LLM, PLR, and REG versus number of testing pairs ($|\mathcal{V}|$ size) over (upper) R2 and (lower) R1; $d \in \{2, 5\}$, $a = 0.25$

8.6 Data output predictability

We compare our data-LLM functions against the in-RDBMS REG and PLR models for providing accurate data output predictions w.r.t. the A2 metric, i.e., the data prediction performance with statistical level of significance 5%. We use our data-LLM functions for providing data output u predictions over unseen data subspaces $\mathbb{D}(\mathbf{x}, \theta)$ using (31) and (29) for prediction. Figures 25 and 26(upper) show the RMSE v for the LLM, REG, and PLR models over the R1, R2, and R3 datasets against the testing set size $|\mathcal{V}|$.

The LLM model can successfully predict the data output u by being statistically robust in terms of number of testing pairs $|\mathcal{V}|$ ($p < 0.05$) and assume comparable or, even, lower prediction error than the REG model. This denotes that our model, by fusing different data-LLM functions which better capture the characteristics of the underlying data function g , provides better data output u prediction than a ‘global’ REG model over random queried data subspaces \mathbb{D} . Evidently,

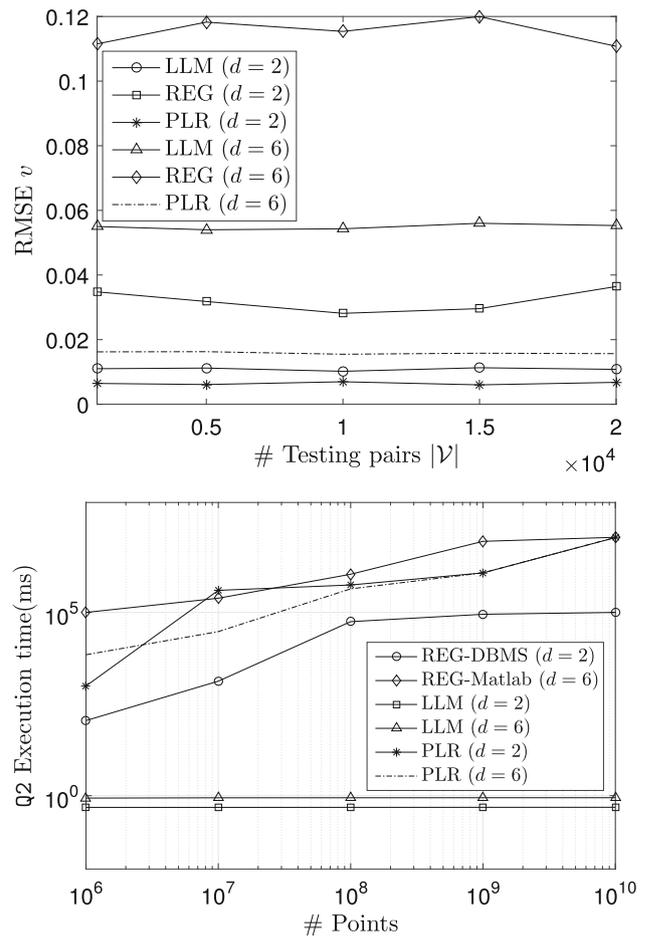


Fig. 26 Q2: (Upper) RMSE v for data output u of LLM, PLR, and REG versus number of testing pairs ($|\mathcal{V}|$ size) over R3; (lower) query execution time versus # points for LLM, PLR, and REG over R3; $d \in \{2, 6\}$, $a = 0.25$

the PLR model achieves the lowest RMSE value by actually accessing the data and captures the actual nonlinearity of the data function g through linear models. However, this is achieved with relatively high computational complexity, higher than the REG model including polynomially data-access process [44]. Note, the data output prediction times for the LLM, REG, and PLR models in this experiment are the same presented in Fig. 24: The LLM model executes our Algorithm 2 by replacing $\theta = \theta_k$ in (30), $\forall \mathbf{w}_k \in \mathcal{W}(\mathbf{q})$, the REG model creates the linear approximation over the data space \mathbb{D} , and the PLR adaptively finds the best linear models for data fitting in each prediction request.

Overall, the proposed LLM model through the training, intermediate and prediction phases achieves statistically significant scalability and accuracy performance compared with the in-RDBMS REG model and the data-access intensive PLR model ($p < 0.05$). The scalability of the proposed model in the predictive analytics era is achieved by predicting the query answers and delivering to analysts the statistical

behavior of the underlying data function without accessing the raw data and without processing/executing the analytics queries, as opposed to the data-driven REG and PLR models in the literature.

8.7 Impact of radius θ

In this section we examine the impact of the query radius θ on the predictive and scalability performance of our query-driven approach. Consider that the query function f is approximated by some function \hat{f} . Then, the expected prediction error (EPE) in (5) for a random query \mathbf{q} with actual aggregate answer y is decomposed as:

$$\mathbb{E}[\hat{f}(\mathbf{x}, \theta) - f(\mathbf{x}, \theta)]^2 + \mathbb{E}[\hat{f}(\mathbf{x}, \theta) - \mathbb{E}[\hat{f}(\mathbf{x}, \theta)]]^2 + \sigma^2. \quad (40)$$

The first term is the squared bias, i.e., the difference between the true function f and the expected value of the estimate $\mathbb{E}[\hat{f}]$, where the expectation averages the randomness in the dataset. This term will most likely increase with radius θ , which implies an increase in the number of input data points $n_\theta(\mathbf{x})$ from the dataset \mathcal{B} . The second term is a variance that decreases as the query radius θ increases. The third term is the irreducible error, i.e., the noise in the true relationship that cannot fundamentally be reduced by any model with variance $\sigma^2 = \text{Var}(\epsilon)$. The θ value controls the influence that each neighbor query point has on the aggregate answer prediction. As radius θ varies there is a bias-variance trade-off. An increase in radius θ results in smoother aggregate answer prediction but increasing the bias. On the other hand, the variance goes to zero since, for instance, with a high radius θ , the prediction $\hat{f}(\mathbf{x}, \theta) \approx \mathbb{E}[y]$, i.e., unconditioned to the query \mathbf{q} . Imagine queries whose radii include all data points \mathbf{x}_i in the entire data space. In that case, we obtain a constant aggregate answer for each issued query, i.e., the aggregate answer $y = 1/n \sum_{i=1}^n u_i$, which is the average of all data outputs u_i thus, no need to predict the aggregate answer y . By selecting a radius θ such that $n_\theta(\mathbf{x}) \approx n$, then the aggregate answer prediction is a relatively smooth function of query \mathbf{q} , but has little to do with the actual positions of the data vectors \mathbf{x}_i 's over the data space. Evidently, the variance contribution to the expected error is then small. On the other hand, the prediction to a particular query \mathbf{q} is systematically biased toward the population response, regardless of any evidence for local variation in the data subspace $\mathbb{D}(\mathbf{x}, \theta)$. The other extreme is to select a radius θ such that $n_\theta(\mathbf{x}) = 1$ for all queries. We can expect less bias and, in this case, it goes to zero if $n \rightarrow \infty$ [32]. Finally, Given the true model and infinite data, we should be able to reduce both the bias and variance terms to 0. That is, as the number of input data points n and $n_\theta(\mathbf{x}) \rightarrow \infty$ such that $\frac{n_\theta(\mathbf{x})}{n} \rightarrow 0$, then $\hat{f}(\mathbf{x}, \theta) \rightarrow \mathbb{E}[y|\mathbf{x}, \theta]$. However, in real world with approximate models and finite

data, the radius θ plays the trade-off between minimizing the bias and minimizing the variance.

We experiment with different mean values μ_θ of the radius $\theta \sim \mathcal{N}(\mu_\theta, \sigma_\theta^2)$ having a fixed variance σ_θ^2 to examine the impact on the model training, quality of aggregate answer prediction, and PLR approximation of the underlying data function g . We examine the number of training pairs, $|\mathcal{T}|$, where our method requires to reach the convergence threshold $\gamma = 0.01$. We also examine the impact of radius θ on the RMSE and CoD metrics. Hence, three factors ($|\mathcal{T}|$, RMSE, and CoD) are influenced by the radius θ . We experiment with mean radius $\mu_\theta \in \{0.01, \dots, 0.99\}$ over the R1 dataset (similar results are obtained in R2 and R3 datasets). Consider the queries with high radius θ drawn from Gaussian $\mathcal{N}(\mu_\theta, \sigma_\theta^2)$ with high mean radius μ_θ . Then, radius θ nearly covers the entire input data range and aggregate answer y is close to the average value of output u for all queries, i.e., n_θ contains all \mathbf{x} input data points. In this case, all query prototypes \mathbf{w}_k correspond to constant query-LLM functions $f_k(\mathbf{x}, \theta) \approx y_k = y$, where aggregate answer $y = \mathbb{E}[u]$ unconditioned to \mathbf{x} and radius θ . Hence, the training and convergence of all LLMs is *trivial* since there is no any specificity to be extracted from each query-LLM function f_k . Our method converges with a low number of training pairs $|\mathcal{T}|$ as shown in Fig. 27(lower). On the other hand, a small θ value refers to learning ‘meticulously’ all the specificities for all LLMs. In this case, our method requires a relatively high number of training query–answer pairs $|\mathcal{T}|$ to converge; see Fig. 27(lower).

In terms of accuracy, the higher the radius θ is, the lower the RMSE e becomes. With high radius θ , all query-LLM functions refer to constant functions with the extreme case where $f_k \approx \mathbb{E}[u], \forall k$ as discussed above, thus, the RMSE $e = \sqrt{\frac{1}{M} \sum (y_i - \hat{y}_i)^2} \rightarrow 0$ with $y_i \approx \mathbb{E}[u]$ due to the fact that n_θ contains all input data and, thus, $\hat{y}_i \approx \mathbb{E}[u]$ (see Fig. 27(upper) with $|\mathcal{T}| = 5359$ training pairs required for convergence w.r.t. $\gamma = 0.01$). However, this comes at the expense of a low CoD R^2 since the data function g is approximated ‘solely’ by a constant approximate function $g(\mathbf{x}) \approx \mathbb{E}[u]$ (see Fig. 27(lower)). When radius θ is small, we attempt to estimate the query function f over (\mathbf{x}, θ) and, thus, approximate the data function $g(\mathbf{x})$. This, however, requires many training query–answer pairs $|\mathcal{T}|$; see Fig. 27(lower). Overall, there is a trade-off in the number of training pairs $|\mathcal{T}|$ with approximation and accuracy capability. To obtain quality approximation, the CoD metric should be strictly greater than zero. This is achieved by setting the mean radius value $\mu_\theta < (0.4, 0.5)$ for $d \in (5, 2)$. Then, we can compensate the RMSE and training time (number of training pairs $|\mathcal{T}|$) as shown in Figs. 27 and 28. In addition, there is a trade-off between training effort and predictability. As shown in Figs. 27, 28 and as explained above, a low μ_θ value results to a high RMSE and training effort in terms of $|\mathcal{T}|$ size. By

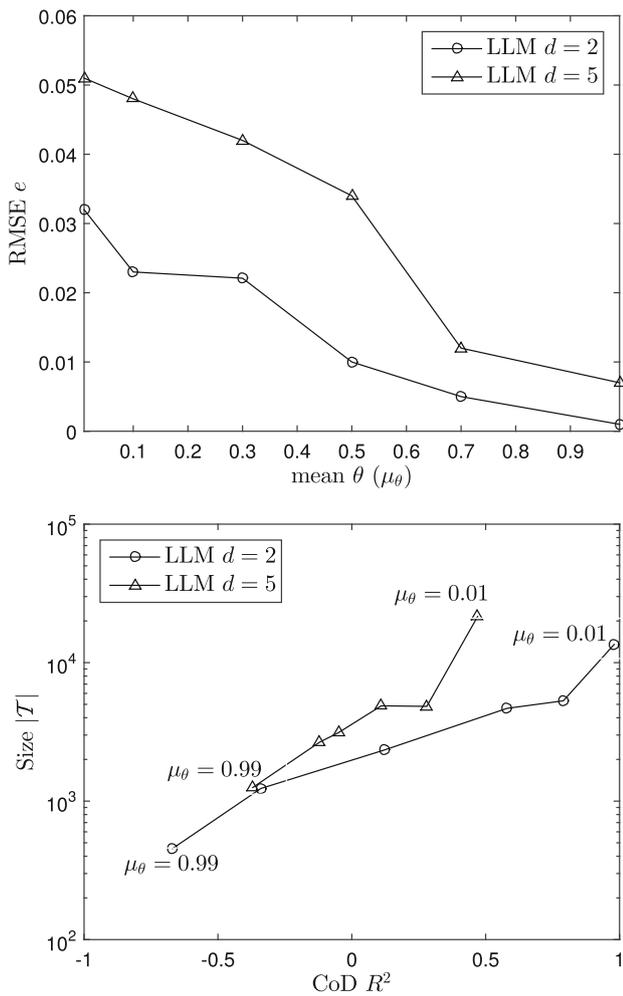


Fig. 27 Trade-off: (upper) RMSE e versus mean θ (μ_θ); (lower) size $|T|$ versus CoD R^2 with μ_θ ; $d \in \{2, 5\}$, $a = 0.25$

combining those trade-offs, for a reasonable training effort, to achieve low RMSE and high goodness of fit, i.e., a high positive CoD value, we set mean radius value $\mu_\theta = 0.1$, which corresponds to $\sim 20\%$ of the data range for $\sigma_\theta^2 = 0.01$. Finally, Fig. 29 shows the impact (trajectory) of the mean radius μ_θ on the training set size $|T|$, the prediction accuracy w.r.t RMSE e , and the goodness of fit w.r.t CoD R^2 metric for $d \in (2, 5)$ for R1; we obtain similar results for R2 and R3 datasets.

9 Conclusions and future plans

We focused on the inferential task of piecewise linear regression and predictive modeling which are central to in-DMS predictive analytics. We introduced an investigation route, whereby answers from previously executed aggregate and regression queries are exploited to train novel statistical learning models which discover and approximate the

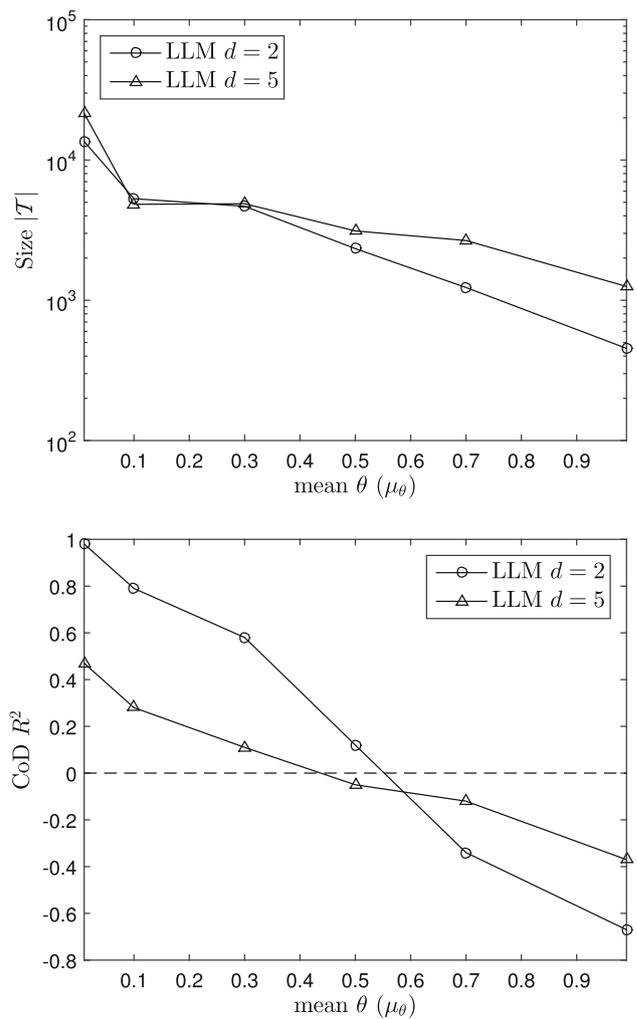


Fig. 28 (Upper) Number of training pairs $|T|$ versus mean θ (μ_θ); (lower) CoD R^2 versus mean θ (μ_θ); $d \in \{2, 5\}$, $a = 0.25$

unknown underlying data function with piecewise linear regression planes, predict future mean-value query answers, and predict the data output. We contribute with a statistical learning methodology, which yields highly accurate answers and data function approximation based only on the query-answer pairs and avoiding data access after the model training phase. The performance evaluation and comparative assessment revealed very promising results.

Our methodology is shown to be highly accurate, extremely efficient in computing query results (with sub-millisecond latencies even for massive datasets, yielding up to six orders of magnitude improvement compared to computing exact answers, produced by piecewise linear regression and global linear approximation models), and scalable, as predictions during query processing do not require access to the DMS engine, thus being insensitive to dataset sizes.

Our plans for future work focus on developing a framework that can dynamically and optimally switch between

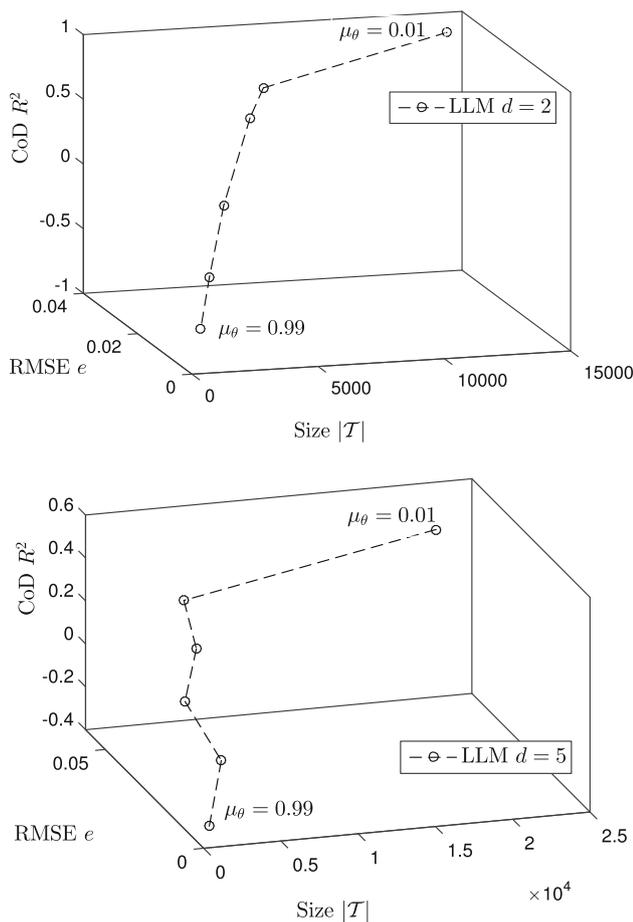


Fig. 29 Impact of μ_θ size $|T|$, RMSE e , and CoD R^2 for $d = 2$ (upper) and $d = 5$ (lower) on R1; $a = 0.25$

the training/intermediate phases and query prediction phases as analysts interests shift between data subspaces. Moreover, the developing framework is expected to cope with nonlinear approximations by evolving and expanding the fundamental representatives of both: data and query subspaces for supporting robust query subspace adaptation and for dealing with data spaces with online data updates.

Acknowledgements The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This work is funded by the EU H2020 GNFUV Project RAWFIE–OC2–EXP–SCI (Grant#645220), under the EC FIRE+ initiative.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

Appendix

See Table 2.

Table 2 Nomenclature

| Notation | Explanation |
|---|--|
| d | Data dimensionality |
| $\mathbf{x} \in \mathbb{R}^d$ | Data input d -dimensional vector |
| $u \in \mathbb{R}$ | Data output value (dependent variable) |
| $\mathbf{q} \in \mathbb{R}^{d+1}$ | Query vector |
| $\theta \in \mathbb{R}_+$ | Query radius |
| $\mathbb{Q} \subset \mathbb{R}^{d+1}$ | Query subspace |
| $\mathbb{D} \subset \mathbb{R}^d$ | Data subspace |
| $\mathbb{D}(\mathbf{x}, \theta) \subset \mathbb{R}^d$ | Data subspace defined by query $\mathbf{q} = [\mathbf{x}, \theta]$ |
| $\mathbf{b}_k \in \mathbb{R}^d$ | d -Dimensional linear regression coefficient (slope; PLR prototype) |
| $y_k \in \mathbb{R}$ | Intercept linear regression coefficient (PLR prototype) |
| $\ \mathbf{x}\ $ | Euclidean norm of vector \mathbf{x} |
| $u = g(\mathbf{x}) \in \mathbb{R}$ | Underlying real data function |
| $y = f(\mathbf{x}, \theta) \equiv f(\mathbf{q}) \in \mathbb{R}$ | Mean-value query function |
| $y \in \mathbb{R}$ | Mean-value answer |
| \mathcal{L}_K | Family of piecewise linear regression functions with K line segments |
| $\mathbf{w}_k \in \mathbb{R}^{d+1}$ | Query prototype |
| $f_k(\mathbf{x}, \theta)$ | Query-LLM |
| $g_k(\mathbf{x})$ | Data-LLM derived from f_k query-LLM |
| K | Number of query-LLM prototypes |
| $\rho \in \mathbb{R}_+$ | Vigilance parameter |
| $\gamma \in \mathbb{R}_+$ | Convergence threshold |
| $\Gamma \in \mathbb{R}$ | Vector norm difference of the optimization parameters |
| $r \in (0.5, 1)$ | Consensual threshold |
| \mathcal{F} | Statistical methodology |
| \mathcal{T} | Training set of (query–answer) pairs |
| \mathcal{V} | Testing set of (query–answer) pairs |
| \mathcal{B} | Data set of (\mathbf{x}, u) pairs |
| \mathcal{J} | Objective function for minimizing the mean squared distortion error |
| \mathcal{H} | Objective function for minimizing the mean squared prediction error |
| $\mathcal{W}(\mathbf{q})$ | Overlapping set |
| $\mathcal{C}(\mathbf{q})$ | Set of converged query parameters (intermediate phase) |
| $\mathcal{U}(\mathbf{q})$ | Set of un-converged query parameters (intermediate phase) |
| \mathcal{S} | Set of data-LLMs |
| $\alpha_k = (\mathbf{w}_k, y_k, \mathbf{b}_k)$ | k -th Query-LLM parameters set |
| $n_\theta(\mathbf{x}) \in \mathbb{N}$ | Number of data input vectors in a data subspace $\mathbb{D}(\mathbf{x}, \theta)$ |
| $\eta_t \in (0, 1)$ | Hyperbolic schedule/learning rate in SGD |
| $v(\mathbf{x})$ | Vector quantizer function of vector \mathbf{x} |
| $k \in [K]$ | Compact notation of $k = 1, \dots, K$ |
| $a \in (0, 1)$ | Vigilance coefficient |
| $e \in \mathbb{R}$ | Root-mean-squared error (RMSE) |

Table 2 continued

| Notation | Explanation |
|--|---|
| $\tilde{\epsilon} \in \mathbb{R}$ | Partial root-mean-squared error (RMSE in intermediate phase) |
| s | Coefficient of determination (R^2) metric |
| $\delta(\mathbf{q}, \mathbf{q}') \in (0, 1)$ | Degree of query overlapping |
| $\tau, \tau^*, \tau_{\top}$ | τ -th Observation of query–answer pair, convergence landmark, intermediate phase landmark |
| $\mathcal{N}(\mu_{\theta}, \sigma_{\theta}^2)$ | Gaussian distribution for radius θ with mean μ_{θ} and variance σ_{θ}^2 |
| $ \mathcal{T} $ | Cardinality of set \mathcal{T} |

References

- Abbott, D.: Applied Predictive Analytics: Principles and Techniques for the Professional Data Analyst, 1st edn. Wiley, Hoboken (2014)
- Adjeroh, D.A., Lee, M.C., King, I.: A distance measure for video sequence similarity matching. In: Proceedings International Workshop on Multi-Media Database Management Systems (Cat. No.98TB100249), pp. 72–79 (1998)
- Amirian, P., Basiri, A., Morley, J.: Predictive analytics for enhancing travel time estimation in navigation apps of apple, google, and microsoft. In: Proceedings of the 9th ACM SIGSPATIAL International Workshop on Computational Transportation Science, IWCTS '16, pp. 31–36. ACM, New York (2016)
- Anagnostopoulos, C.: Quality-optimized predictive analytics. Appl. Intell. **45**(4), 1034–1046 (2016)
- Anagnostopoulos, C., Kolomvatsos, K.: Predictive intelligence to the edge through approximate collaborative context reasoning. Appl. Intell. **48**(4), 966–991 (2018)
- Anagnostopoulos, C., Savva, F., Triantafillou, P.: Scalable aggregation predictive analytics: a query-driven machine learning approach. Appl. Intell. **48**, 2546 (2018). <https://doi.org/10.1007/s10489-017-1093-y>
- Anagnostopoulos, C., Triantafillou, P.: Learning set cardinality in distance nearest neighbours. In: 2015 IEEE International Conference on Data Mining, pp. 691–696 (2015)
- Anagnostopoulos, C., Triantafillou, P.: Efficient scalable accurate regression queries in in-dbms analytics. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 559–570 (2017). <https://doi.org/10.1109/ICDE.2017.111>
- Anagnostopoulos, C., Triantafillou, P.: Query-driven learning for predictive analytics of data subspace cardinality. ACM Trans. Knowl. Discov. Data **11**(4), 47 (2017). <https://doi.org/10.1145/3059177>
- Ari, B., Gvenir, H.A.: Clustered linear regression. Knowl. Based Syst. **15**(3), 169–175 (2002)
- Avron, H., Sindhvani, V., Woodruff, D.P.: Sketching structured matrices for faster nonlinear regression. In: Proceedings of the 26th International Conference on Neural Information Processing Systems, NIPS'13, pp. 2994–3002. Curran Associates Inc. (2013)
- Bagirov, A., Clausen, C., Kohler, M.: An algorithm for the estimation of a regression function by continuous piecewise linear functions. Comput. Optim. Appl. **45**(1), 159–179 (2010)
- Bai, J., Perron, P.: Estimating and testing linear models with multiple structural changes. Econometrica **66**(1), 47–78 (1998)
- Bottou, L.: Stochastic gradient descent tricks. In: Montavon, G., Orr, G.B., Müller, K.R. (eds.) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol. 7700, 2nd edn, pp. 421–436. Springer, Berlin (2012)
- Bousquet, O., Bottou, L.: The tradeoffs of large scale learning. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S.T. (eds.) Advances in Neural Information Processing Systems, vol. 20, pp. 161–168. Curran Associates Inc, Red Hook (2008)
- Candanedo, L.M., Feldheim, V., Deramaix, D.: Data driven prediction models of energy use of appliances in a low-energy house. Energy Build. **140**, 81–97 (2017)
- Chatterjee, S., Guntuboyina, A., Sen, B.: On risk bounds in isotonic and other shape restricted regression problems. Ann. Stat. **43**(4), 1774–1800 (2015)
- Cherkassky, V., Lari-Najafi, H.: Constrained topological mapping for nonparametric regression analysis. Neural Netw. **4**(1), 27–40 (1991)
- Choi, C.H., Choi, J.Y.: Constructive neural networks with piecewise interpolation capabilities for function approximations. IEEE Trans. Neural Netw. **5**(6), 936–944 (1994)
- Choi, J.Y., Farrell, J.A.: Nonlinear adaptive control using networks of piecewise linear approximators. IEEE Trans. Neural Netw. **11**(2), 390–401 (2000)
- Cohen, J., Dolan, B., Dunlap, M., Hellerstein, J.M., Welton, C.: Mad skills: new analysis practices for big data. Proc. VLDB Endow. **2**(2), 1481–1492 (2009)
- Dean, J., Corrado, G.S., Monga, R., Chen, K., Devin, M., Le, Q.V., Mao, M.Z., Ranzato, M., Senior, A., Tucker, P., Yang, K., Ng, A.Y.: Large scale distributed deep networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems, NIPS'12, pp. 1223–1231. Curran Associates Inc. (2012)
- Deshpande, A., Madden, S.: Mauvedb: Supporting model-based user views in database systems. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, SIGMOD '06, pp. 73–84. ACM, New York (2006)
- Di Blas, N., Mazuran, M., Paolini, P., Quintarelli, E., Tanca, L.: Exploratory computing: a comprehensive approach to data sense-making. Int. J. Data Sci. Anal. **3**(1), 61–77 (2017)
- Dennis Jr., J.E., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Prentice Hall Series in Computational Mathematics. Prentice Hall, Upper Saddle River (1983)
- Fan, J., Gijbels, I.: Local Polynomial Modelling and Its Applications. Monographs on Statistics and Applied Probability Series, vol. 66. Chapman & Hall, London (1996)
- Feng, X., Kumar, A., Recht, B., Ré, C.: Towards a unified architecture for in-rdbms analytics. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, SIGMOD '12, pp. 325–336. ACM, New York (2012)
- Ferrari-Trecate, G., Muselli, M.: A new learning method for piecewise linear regression. In: Artificial Neural Networks—ICANN 2002, International Conference, Madrid, 28–30 Aug 2002, Proceedings, pp. 444–449 (2002)
- Freedman, D.: Statistical Models: Theory and Practice. Cambridge University Press, Cambridge (2005)
- Grossberg, S.: Adaptive resonance theory: how a brain learns to consciously attend, learn, and recognize a changing world. Neural Netw. **37**, 1–47 (2013)
- Harth, N., Anagnostopoulos, C.: Quality-aware aggregation predictive analytics at the edge. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 17–26 (2017)
- Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Series in Statistics. Springer, New York (2001)
- Jeffreys, H., Jeffreys, B.S.: 'Taylor's Theorem' Paragraph. Methods of Mathematical Physics, vol. 1.133, 3rd edn, pp. 50–51. Cambridge University Press, Cambridge (1988)
- Jordan, M.I.: On statistics, computation and scalability. Bernoulli **19**(4), 1378–1390 (2013)

35. Jordan, M.I.: Computational thinking, inferential thinking and “big data”. In: Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS '15, pp. 1–1. ACM, New York (2015)
36. Khattree, R., Bahuguna, M.: An alternative data analytic approach to measure the univariate and multivariate skewness. *Int. J. Data Sci. Anal.* (2018). <https://doi.org/10.1007/s41060-018-0106-1>
37. Kyng, R., Rao, A., Sachdeva, S.: Fast, provable algorithms for isotonic regression in all p -norms. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, NIPS' 15, pp. 2719–2727. MIT Press, Cambridge (2015)
38. Li, X., Anselin, L., Koschinsky, J.: Geoda web: enhancing web-based mapping with spatial analytics. In: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL '15, pp. 94:1–94:4. ACM, New York (2015)
39. Meyer, M.C.: Inference using shape-restricted regression splines. *Ann. Appl. Stat.* **2**(3), 1013–1033 (2008)
40. Moustra, M., Avraamides, M., Christodoulou, C.: Artificial neural networks for earthquake prediction using time series magnitude data or seismic electric signals. *Expert Syst. Appl.* **38**(12), 15032–15039 (2011)
41. Mukherji, A., Lin, X., Toto, E., Botaish, C.R., Whitehouse, J., Rundensteiner, E.A., Ward, M.O.: Fire: a two-level interactive visualization for deep exploration of association rules. *Int. J. Data Sci. Anal.* **2018**, 1–26 (2018)
42. Nakayama, K., Hirano, A., Kanbe, A.: A structure trainable neural network with embedded gating units and its learning algorithm. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, vol. 3, pp. 253–258 (2000)
43. Nothaft, F.A., Massie, M., Danford, T., Zhang, Z., Laserson, U., Yeksigian, C., Kottalam, J., Ahuja, A., Hammerbacher, J., Linderman, M., Franklin, M.J., Joseph, A.D., Patterson, D.A.: Rethinking data-intensive science using scalable analytics systems. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, SIGMOD '15, pp. 631–646. ACM, New York (2015)
44. O'Sullivan, F.: Discussion: multivariate adaptive regression splines. *Ann. Stat.* **19**(1), 99–102 (1991)
45. Rodríguez-Lujan, I., Fonollosa, J., Vergara, A., Homer, M., Huerta, R.: On the calibration of sensor arrays for pattern recognition using the minimal number of experiments. *Chemom. Intell. Lab. Syst.* **130**, 123–134 (2014)
46. Rosenbrock, H.H.: An automatic method for finding the greatest or least value of a function. *Comput. J.* **3**(3), 175 (1960)
47. Salloum, S., Dautov, R., Chen, X., Peng, P.X., Huang, J.Z.: Big data analytics on apache spark. *Int. J. Data Sci. Anal.* **1**(3), 145–164 (2016)
48. Schleich, M., Olteanu, D., Ciucanu, R.: Learning linear regression models over factorized joins. In: Proceedings of the 2016 International Conference on Management of Data, SIGMOD '16, pp. 3–18. ACM, New York (2016)
49. Schneider, P., Biehl, M., Hammer, B.: Adaptive relevance matrices in learning vector quantization. *Neural Comput.* **21**(12), 3532–3561 (2009)
50. Thiagarajan, A., Madden, S.: Querying continuous functions in a database system. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08, pp. 791–804. ACM, New York (2008)
51. Trippa, L., Waldron, L., Huttenhower, C., Parmigiani, G.: Bayesian nonparametric cross-study validation of prediction methods. *Ann. Appl. Stat.* **9**(1), 402–428 (2015)
52. Vapnik, V.N., Chervonenkis, A.Y.: On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.* **16**(2), 264–280 (1971)
53. Venkataraman, S., Yang, Z., Franklin, M., Recht, B., Stoica, I.: Ernest: Efficient performance prediction for large-scale advanced analytics. In: Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation, NSDI'16, pp. 363–378. USENIX Association, Berkeley (2016)
54. Yamamoto, Y., Perron, P.: Estimating and testing multiple structural changes in linear models using band spectral regressions. *Econom. J.* **16**(3), 400–429 (2013)
55. Yeh, E., Niekrasz, J., Freitag, D.: Unsupervised discovery and extraction of semi-structured regions in text via self-information. In: Proceedings of the 2013 Workshop on Automated Knowledge Base Construction, AKBC '13, pp. 103–108. ACM, New York (2013)
56. Zheng, L., Wang, S., Liu, Y., Lee, C.H.: Information theoretic regularization for semi-supervised boosting. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, pp. 1017–1026. ACM, New York (2009)
57. Zhou, X., Zhou, X., Chen, L., Shu, Y., Bouguettaya, A., Taylor, J.A.: Adaptive subspace symbolization for content-based video detection. *IEEE Trans. Knowl. Data Eng.* **22**(10), 1372–1387 (2010)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.