Aggregate Query Prediction under Dynamic Workloads

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

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

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

***Abstract*—Large organizations have seamlessly incorporated data-driven decision making in their operations. However, as data volumes increase, expensive big data infrastructures are called to rescue. In this setting, analytics tasks become very costly in terms of query response time, resource consumption, and money in cloud deployments, especially when base data are stored across geographically distributed data centers. Therefore, we introduce an adaptive Machine Learning mechanism which is light-weight, stored client-side, can estimate the answers of a variety of aggregate queries and can avoid the big data backend. The estimations are performed in milliseconds and are inexepensive as the mechanism learns from past analytical-query patterns. However, as analytic queries are ad-hoc and analysts' interests change over time we develop solutions that can swiftly and accurately detect such changes and adapt to new query patterns. The capabilities of our approach are demonstrated using extensive evaluation with real and synthetic datasets.**

***Index Terms*—life-long learning, approximate query processing, machine learning, concept drift detection, change-point detection**

I. Introduction

With the rapid explosion of data volumes and the adoption of data-driven decision making, organizations have been struggling to process data efficiently. Because of that a number of companies is turning to popular cloud providers that have created large-scale systems capable of storing and processing large quantities of data. However, the problem still remains in that multiple analytics queries are issued by multiple analysts (Figure 1) which often overburden data clusters and are costly. Data analysts should be able to extract information

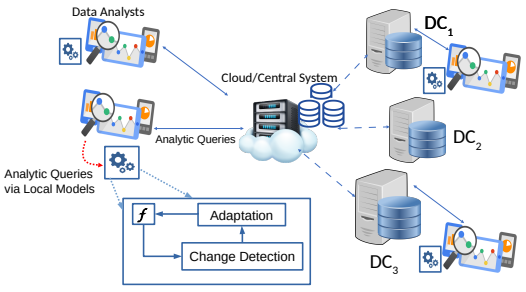

Fig. 1. Aggregate analytics eco-system with analysts' devices, data centers, and local adaptive ML models.

without significant delays so as not to violate the *interactivity constraint* set around 500ms [15]. Anything over that limit can negatively affect analysts' experience and productivity. This constraint is particularly important in the context of *exploratory analysis* [13]. Such analyses are an invariable step in the process of understanding data and creating solutions to support business decisions.

Vision: Depicted at Figure 1 is our vision for an aggregate analytics learning & prediction system that is light-weight, stored on an Analyst's Device (AD) and adaptive to dynamic query workloads. This allows the exploratory process to be executed locally at ADs providing predictions to aggregate queries thus not overburdening the Cloud/Central System (CS). Prediction-based aggregate analytics is expected to save computational and communication resources, which could be devoted to cases where accurate answers to aggregate queries are demanded. From the CS's perspective, our system acts as a *pseudo*-caching mechanism to reduce communication overhead and computational load when it is necessary, thus, allowing for other tasks/processes to run.

Our system offers a learning-based, prediction-driven way of performing aggregate analytics in ADs accessing *no data*. It neither requires data transmission from CS to ADs nor from ADs to CS. What makes such a system possible is the exploitation of previously executed queries and their answers sitting in log files. We adopt Machine Learning (ML) regression models that learn to associate past executed queries with their answers, and can in turn locally predict the answers of *new* queries. Subsequent aggregate queries are answered in milliseconds, thus, fulfilling the interactivity constraint.

Furthermore, our framework can directly adapt to analysts' (dynamic) query workloads by monitoring the analysts' query patterns and adjusting their parameters. Shown at Figure 1 are the ML models f and mechanisms developed for detecting and adapting to changes in query patterns. Both of them are discussed in Sections III and IV.

Challenges & Contribution: A large number of analysts exist within an organization with diverse analytics interests thus their query patterns are expected to differ, accessing different parts of the whole data-space. We are challenged to support model training over vastly different patterns, which are to be drastically changing or expanding in dynamic environments. Moreover the models have to be up-to-date w.r.t. pattern changes, which require early query pattern change detection and efficient adaptation. Given these challenges, our contributions are:

1) A novel query-driven mechanism and query representation that associates queries with their respective answers and can be used by ML models.
2) A local change detection mechanism for detecting changes in query patterns based on our prediction-error approximation.
3) A reciprocity-based adaptation mechanism in light of novel query patterns, which efficiently engages the CS to validate the prediction-to-adaptation states transition and guarantees system convergence.
4) Comprehensive assessment of the system performance and sensitivity analysis using real and synthetic data and query workloads.

II. Fundamentals of Query-Driven Learning

The fundamentals of query-driven mechanism for analytics are: (1) transforming analytic queries in a real-valued vectorial space, (2) quantization of vectorial space, extracting query patterns and (3) training of local regression models for predicting query answers using past issued queries. Principally, we learn to associate the result of a query using the derived query patterns and linking these patterns with local regression models. Given an unseen query, we project it to the closest query pattern we have learned and then predict its corresponding result without executing the query over the data in a DC/CS.

Definition 1: A dataset $\mathcal{B} = \{\mathbf{a}\}_{i=1}^{n}$ is a set of n row data vectors $\mathbf{a} = [a_1, \ldots, a_d] \in \mathbb{R}^d$ with real attributes $a_i \in \mathbb{R}$.

Predicates over attributes define a data subspace over $\mathcal{B}$ formed by a sequence of logical conjunctions using (in)equality constraints ($\leq, \geq, =$). A *range-predicate* restricts an attribute a_i to be within range $[l_i, u_i]$: $a_i \geq l_i \wedge a_i \leq u_i$. We model a range query over $\mathcal{B}$ through conjunctions of predicates, i.e., $\bigwedge_{i=1}^{d}(l_i \leq a_i \leq u_i)$ represented as a vector in $\mathbb{R}^{2d}$.

Definition 2: A (range) row query vector is defined as $\mathbf{q} = [l_1, u_1, \ldots, l_d, u_d] \in \mathbb{R}^{2d}$ corresponding to the range query $\bigwedge_{i=1}^{d}(l_i \leq a_i \leq u_i)$. The distance between two queries $\mathbf{q}$ and $\mathbf{q}'$ is defined as the L_2^2 norm (Euclidean distance): $\|\mathbf{q}-\mathbf{q}'\|_2^2 = \sum_{i=1}^{d}(l_i - l_i')^2 + (u_i - u_i')^2$.

This representation is flexible enough to accommodate a wide variety of queries. As the dimensionality of the query vector is proportional to the data vector, queries with predicates bounding the values of different attributes can be used by the same ML algorithm. This means that only a number of (l_i, u_i) values are set w.r.t the number of predicates for a given query. In addition, we make no assumptions as to the back-end system as what is being parsed are the filters in a query. This allows the mechanism to be deployed in parallel to multiple analytic systems.

In query-driven learning, we learn to associate a query with its corresponding aggregate result (a scalar $y \in \mathbb{R}$). This is achieved using a $training$ set of query-result pairs $\mathcal{T} = \{(\mathbf{q}_i, y_i)\}_{i=1}^{N}$ obtained after executing N queries over dataset $\mathcal{B}$. The goal is to develop an ML model based on $\mathcal{T}$ to minimize the expected prediction error between actual y and predicted $\hat{y}$, $\mathbb{E}[(\hat{y} - y)^2]$ and predict the result of any unseen query *without* executing it over $\mathcal{B}$.

A. Query Space Clustering

Recent research analyzing analytics workloads from various domains has shown that queries within analytics workloads share patterns and their results are *similar* having various degrees of overlap [23]. Based on this evidence, we mine query logs (the $training$ set) and discover clusters of queries (in the vectorial d-dimensional space), having similar predicate parameters. This partitioning is fundamental to get accurate ML models for predictive analytics, as we then associate different ML predictive models with different clusters. In this way, learning different data sub-sets is proven to be more efficient in terms of *explainability / model-fitting* and *predictability* than having one global ML model learn everything and is also known as *ensemble learning* [10].

To put this in context, consider a discrete time domain $t \in \mathbb{T} = \{1, 2, 3, \ldots\}$, where at each time instance t an analyst issues a query $\mathbf{q}_t$. The query is executed and an answer y_t is obtained, forming the pair $(\mathbf{q}_t, y_t)$. The issued queries are stored in a growing set $\mathcal{C}_t = \{(\mathbf{q}_1, y_1), \ldots, (\mathbf{q}_t, y_t)\} = \mathcal{C}_{t-1} \cup \{(\mathbf{q}_t, y_t)\}$. Given this set, we incrementally extract knowledge from the query vectors and then train *local* ML models that predict the associated outputs given new, unseen queries. This is achieved by on-line partitioning the vectors $\{\mathbf{q}_1, \ldots, \mathbf{q}_t\} \in \mathcal{C}_t$ into disjoint clusters that represent the query patterns of the analysts (fundamentally, within each cluster the queries are much more *similar* than the queries in other clusters). The distance between queries quantifies how close the predicate parameters are in the vectorial space. Close queries $\mathbf{q}$ and $\mathbf{q}'$ are grouped together into K[1] clusters w.r.t. $\|\mathbf{q} - \mathbf{q}'\|_2^2$. The objective is to minimize the expected quantization error $\mathbb{E}[\min_{k=1,\ldots,K}\|\mathbf{q} - \mathbf{w}_k\|_2^2]$ of all queries to their closest cluster *representative* $\mathbf{w}_k$, which reflects the analysts query patterns and best represents each cluster. The K query representatives $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$ optimally quantize $\mathcal{C}_t$ minimizing the expected quantization error while each query $\mathbf{q}$ is projected onto its closest representative $\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathcal{W}}\|\mathbf{q} - \mathbf{w}\|_2^2$. Based on the partitioning of $\mathcal{C}_t$, we produce K query-disjoint sub-sets such that $\mathcal{C}_k \cap \mathcal{C}_l \equiv \emptyset$ for $k \neq l$ and $\mathcal{C}_k = \{(\mathbf{q}, y) \in \mathcal{C}_t | \mathbf{w}_k = \arg\min_{\mathbf{w}\in\mathcal{W}}\|\mathbf{q} - \mathbf{w}\|_2\}$. A *local* ML model is then trained over each subset using the pairs in $\mathcal{C}_k, k \in [K]$.

B. Query-Answer Predictive Model

Each aggregate result y from the pair $(\mathbf{q}, y) \in \mathcal{C}_t$ represents the $exact$ answer produced by the CS. Essentially, y is produced by an unknown function g which we wish to learn. Such function produces query-answers w.r.t an unknown distribution $p(y|\mathbf{q})$. Our aim is to approximate the *true* functions g for aggregate functions (descriptive statistics) e.g., count, average, max, sum etc. Regression algorithms are trained using query-answer pairs from $\mathcal{C}_t$ to minimize the expected prediction

[1]The number of clusters K is automatically identified by the clustering algorithm used. [2]

error between actual $y = g(\mathbf{q})$, from the true function g, and predicted $\hat{y}$ from an approximated function $\hat{g}$, i.e., $\mathbb{E}[(g(\mathbf{q}) - \hat{g}(\mathbf{q}))^2]$. After having partitioned the query space into clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$, we therein train K local ML models, $\mathcal{M} = \{\hat{g}_1, \ldots, \hat{g}_K\}$ that associate queries $\mathbf{q}$ belonging to cluster $\mathcal{C}_k$ with their outputs y. Each ML model $\hat{g}_k$ is trained from query-response pairs $(\mathbf{q}, y) \in \mathcal{C}_t$ from those queries $\mathbf{q}$ which belong to $\mathcal{C}_k$ such that $\mathbf{w}_k$ is the closest representative to those queries. The originally trained ML models in DC/CS are then sent to ADs to be used for predicting answers. Given a query $\mathbf{q}$ only the most representative model $\hat{g}_k$ is used for prediction, corresponding to the closest $\mathbf{w}_k$:

$$\hat{y} \quad = \quad \sum_{k=1}^{K} \mathcal{I}_k \hat{g}_k(\mathbf{q}) \tag{1}$$

where $\mathcal{I}_k = 1$ if $\mathbf{w}_k = \arg\min_{\mathbf{w} \in \mathcal{W}} \|\mathbf{q} - \mathbf{w}\|_2^2$; 0 otherwise.

III. Query Pattern Change Detection

Suppose that all trained ML models $\{\hat{g}_k\}_{k=1}^K$ are delivered to the analysts from CS, indicating that the mechanism enters its *prediction mode*. That is, for each incoming query, it predicts the answer and delivers it back to the analysts *without* query execution. However assuming a stationary query pattern distribution is not realistic. It is highly likely that *analysts interests* change over time (e.g., during exploratory analytics tasks, which are considered as ad-hoc processes [13]). So, dynamic workloads may render the $\{\hat{g}_k\}_{k=1}^K$ models obsolete, as they were trained using *past* query patterns following distributions which may now be different. When $p(y|\mathbf{q})$ changes to $p'(y|\mathbf{q})$, it is highly likely that any previous approximation would produce high-error answers, unless $p(y|\mathbf{q}) \approx p'(y|\mathbf{q})$. We capture such dynamics as *concept drift* [9], [21] – many methods have been developed for adjusting when this arises [9], [11].

We introduce a Change Detection Mechanism (CDM) and an Adaptation Mechanism (ADM) (shown at Figure 1) addressing this concern raising a number of challenges: (1) How to detect a query pattern change (2) What kind of action should we take in case that happens (3) How should we notify users, analysts about such change(s) We explore these challenges and describe the decisions we take in tackling them in the remainder.

A. Change Detection Mechanism

Our approach can be best understood by first assuming that the CDM maintains an on-line average of the prediction error $(y - \hat{y})^2$ such that : $u_k \approx \mathbb{E}[(y - \hat{y})^2|\mathbf{q}]$. This is done for each query representative $\mathbf{w}_k$ across different users. Should the expected error u_k escalate significantly, then this may signal that a query pattern has shifted around the 'region' represented by the representative $\mathbf{w}_k$. But, recall that during the prediction mode, the actual y is unknown since our goal is to predict accurate answers but without executing the query itself. Hence, we develop an approximation mechanism for change detection, not requiring query executions over CS/DC.

Once we have trained the individual ML models $\mathcal{M}$ and calculated their expected prediction accuracy (using an independent test sample drawn from the original set of queries) we obtain the Expected Prediction Error (EPE), which will be constant across all possible queries associated with a particular query representative defined as: $EPE = \mathbb{E}\Big[\big(g(\mathbf{q}) - \sum_{\kappa=1}^{K} \mathcal{I}_\kappa \hat{g}_\kappa(\mathbf{q})\big)^2\Big]$. Using the EPE, we wish to find a fine-grained estimate of the true prediction error rather just assuming this is constant for each and every unseen query.

To do this, we have analyzed the error behaviour under changing query patterns[2]. Our findings reveal an interesting fact: The Euclidean distance $d(\mathbf{q}, \mathbf{w}_k) = \|\mathbf{q} - \mathbf{w}_k\|_2^2$ of a random query $\mathbf{q}$ from its closest query representative $\mathbf{w}_k$ is strongly correlated with the associated prediction error $(y - \hat{y})^2$.[3] Considering the correlation between $d(\mathbf{q}, \mathbf{w}_k)$ and the local u_k, we define a distance-based prediction error $\tilde{u}_k$ of a query $\mathbf{q}$ as:

$$\tilde{u}_i = \ln\left(1 + d(\mathbf{q}, \mathbf{w}_k) - \min_{\mathbf{q}_\ell \in \mathcal{C}_k} d(\mathbf{w}_k, \mathbf{q}_\ell)\right) \cdot u_k, \tag{2}$$

where the natural-log operator acts as a penalizing/discount factor for queries given their distance from the closest representative $\mathbf{w}_k$. The second term within the natural-log operator, $\min_{\mathbf{q}_\ell \in \mathcal{C}_k} d(\mathbf{w}_k, \mathbf{q}_\ell)$ is the minimum distance between the query representative $\mathbf{w}_k$ and the associated queries $\mathbf{q} \in \mathcal{C}_k$.

Consider the incoming unseen (random) queries $(\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_t)$ arriving in a sequence during prediction mode. They are answered by specific local ML models $(\hat{g}_0, \hat{g}_1, \ldots, \hat{g}_k)$, generating a series of distance-based error estimations $\{\tilde{u}_t\}$. The CDM monitors this series and, based on a specific threshold, signals the existence of *concept drift*, i.e., checks whether the probability distribution of the queries has changed. Based on the series of error estimations, we learn two query distributions: (1) the *expected* query distribution, which is represented by the query representatives and (2) the *novel* query distribution, which cannot be represented by the current query representatives. The expected distribution $p_0(\tilde{u})$ is estimated given a training period from $\tilde{u}_k$ values corresponding to queries with closest representative $\mathbf{w}_k$. The novel distribution $p_1(\tilde{u})$ is estimated from $\tilde{u}_m$ values corresponding to error values derived from the *rival* representatives $\mathbf{w}_m$ of queries with closest $\mathbf{w}_m$ and $k \neq m$. Based on this, we estimate the distribution of the error values generated from representatives which were not the closest to the queries, thus, approximating novel error values. Both distributions were approximated by fitting the $p(\tilde{u}) \sim \Gamma(e_1, e_2)$ distribution with scale e_1 and shape e_2.

Given a $\tilde{u}_t$ value, we calculate the likelihood ratio $s_t = \log \frac{p_1(\tilde{u}_t)}{p_0(\tilde{u}_t)}$ and the cumulative sum of s_t up to time t, $U_t = \sum_{\tau=0}^{t} s_\tau$. Based on the sequential ratio monitoring for a progressive concept drift in distribution [12] from p_0 to p_1,

[2]The datasets used are described in the Experimental Section

[3]A 0.3 Pearson's Correlations was obtained on a real dataset described in the Experimental Section.

a decision function is introduced for signaling a potential concept drift expressed as:

$$G_t \quad = \quad U_t - \min_{0 \leq \tau \leq t} U_{\tau - 1}. \tag{3}$$

The decision function in (3) indicates the current cumulative sum of ratios minus its current minimum value. This denotes that the *change time* estimate t^* is the time following the current minimum of the cumulative sum, i.e., $t^* = \arg\min_{0 \leq \tau \leq t} U_\tau$.

Hence, a concept drift of query patterns projected over the query representatives space is detected at time t_D: $t_D = \min\{t \geq 0 : G_t > h\}$. The parameter h is usually set $3\sigma \leq h \leq 5\sigma$ with σ the standard deviation of $\tilde{u}$. The cumulative sum of ratios exceeds the threshold h as soon as queries are issued from an unknown distribution as the error estimates become steadily larger and are not just random fluctuations in errors. As soon as a change is detected the CDM signals the ADM component, that new query patterns have been detected. In turn, the ADM signals the *Prediction Component* (containing the $\mathcal{M}$ and $\mathcal{W}$) to be put in `BUFFERING` mode since the prediction component can no longer provide reliable answers for *all* queries. However, the AD can still leverage the complete system to ask queries following known distributions. By entering `BUFFERING` mode our ADM starts to adjust for the new query patterns under the AD until *converging*. At that point it signals the *Prediction Component* to switch back to `PREDICTION` mode, resuming normal operation.

IV. Model Adaptation

A. Model Adaptation & Reciprocity

In the CS, when a query is selectively forwarded from an AD[4], the process of model adaptation has as follows: for adapting to new query patterns, we rely on the principle of *explicit partitioning* [9], [21], as a natural extension of our strategy using an ensemble of local ML models. To adjust to new query patterns, we train a new model $\hat{g}_{K+1}$ using executed queries and their answers in CS. This is the optimal strategy for expanding the current $\mathcal{M}$ as other methods might lead to *catastrophic forgetting* [11].

The adaptation process is performed with parameters: the K query prototypes $\mathcal{W}$ and their associated ML models $\mathcal{M}$. Recall that the analyst's device has cached models $\mathcal{M}$ and the CS adapts the received parameters by learning the new underlying query patterns and based on these trains the new ML model. Let the queries series $\{\mathbf{q}_1, \mathbf{q}_2, \ldots\}$ coming from the AD to CS based on selective forwarding. This means that most likely a query $\mathbf{q}_t$ conforms to new query patterns thus sent to CS for execution. Once query $\mathbf{q}_1$ is executed and its actual answer y_1 is obtained, it is then considered as a new (initial) representative $\mathbf{w}_{K+1}$ for $\mathcal{M}_{K+1}$. The pairs $(\mathbf{q}_t, y_t)$ are then used to incrementally update $\mathbf{w}_{K+1}$ and then buffered in $\mathcal{Q}$, which will be the training set for $\hat{g}_{K+1}$. The adaptation of $\mathbf{w}_{K+1}$ to follow the new query pattern is achieved by Stochastic Gradient Descent (SGD) [7], which is widely used in statistical learning for training in an on-line manner considering one training example (query-answer) at a time. We focus on the convergence of the query distribution by moving the new query representative towards the estimated median of the queries in $\mathcal{Q}$ and not the corresponding centroid. This is introduced so that the new representative converges to a robust statistic, free of outliers and more reliable than the centroid (mean vector). The convergence to the median denotes with high reliability *convergence to the distribution*, which is what we desire for model convergence. In this case, we provide the adaptation rule of the new query representative to converge to the median of the forwarded queries, as provided in Theorem 1.

Theorem 1: The novel representative $\mathbf{w}_{K+1}$ converges to the median vector of the queries executed in the DC w.r.t. update rule $\Delta\mathbf{w}_{K+1} \propto \gamma sgn(\mathbf{q} - \mathbf{w}_{K+1})$, $\gamma \in (0, 1)$; $sgn(\cdot)$ is the signum function.

Proof 1: Proof is omitted and can be found at [19]

The convergence of the representatives is checked by the subsequent adjustments in positions that $\mathbf{w}_{K+1}$ makes. If that change is lower than a threshold c then convergence has been achieved. After the convergence of the query representative, the CS trains the new models $g_{\hat{K+1}}$ and $\{\hat{g}_j\}$, using $\mathcal{Q}$. The new ML models and new representatives are then delivered to AD.

V. Evaluation Results

A. Implementation & Experimental Environment

To implement our algorithms we used `XGBoost` [8] and an implementation of the `Growing-Networks` algorithm [17]. We performed our experiments on a desktop machine with a Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz and 16GB RAM. **Real datasets**: We use the **Crimes** dataset from [1] and the **Sensors** dataset from [3]. The Crimes dataset contains $|\mathcal{R}_1| = 6.6 \cdot 10^6$ and the Sensors dataset $|\mathcal{R}_2| = 2.3 \cdot 10^6$ data vectors. We created synthetic query workloads over these datasets as real query workloads do not exist for this purpose as also attested by [23].

B. Predictability

We experiment with real datasets to demonstrate the applicability of our system under real conditions. As evident from Figure 2 our system provides estimations for descriptive statistics over different types of real datasets with relative error below 10%. (A relative error below 10% is the target of modern state of the art approximate-answer production systems [14]). We also tested these datasets usng VerdictDB [18], a state-of-the-art system in Approximate Query Processing. The errors obtained varied from $1\% - 14\%$ with sampling ratio of $1\% - 10\%$. These results are comparable to ours and show that our system can be reliably used in parallel to such engines when local access is needed and resource consumption at CS is to be minimized. After training the system with more than ca. 2000 queries the relative error starts approaching its minimum value rather swiftly for both

[4]Specifics of selectively forwarding queries can be found at https://arxiv.org/abs/1908.04772

datasets (named **Crimes** and **Sensors**). This demonstrates the capabilities of the proposed learning approach to offer high accuracy estimates for approximating analytical query answers with only a fairly small number of training queries. Note that typical industrial-strength in-production big data analytics clusters used for approximating answers to such analytical queries receive several million of queries per day [14]. Therefore, one can expect that a system employing our approach would receive a few thousand of training queries in a just few tens of seconds.

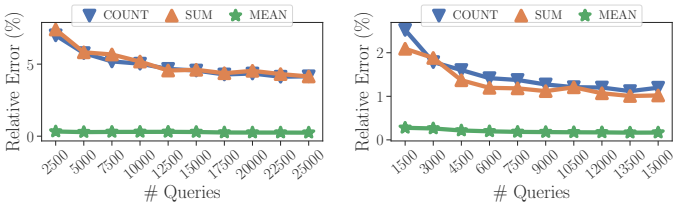

Fig. 2. Relative error vs. number of training queries; (left) Crimes (right) Sensors.

C. Performance

We examine the performance and storage requirements of our system. This is important as our solution has to be light-weight both in terms of storage overhead for ADs and efficient in transferring models through the network. We examine all the above-mentioned models to identify the most efficient ones in training and prediction. A synthetic workload with 50 predicates and 100 columns is used for training all the models. For *Prediction Time* (PT) in Table I we report on the mean and standard deviation of prediction time. The central takeway here is that PTs are negligible – much less than a millisecond, thus, guaranteeing efficient statistics estimation. Even though there are multiple models trained, to account for varying query patterns, the time complexity associated is $\mathcal{O}(K)$ with K usually being small.

For measuring the training time of individual models, we varied the number of training samples and examined the expected model Training Time (TT) in Table I. We used 11 training samples varying in size wrt $\{4 \cdot 10^2, \ldots, 4 \cdot 10^5\}$. We note that its TT is no more than 53 seconds without using its multi-threading capabilities.

We also, examine the model Size in KB shown in Table I and observe that it refers to a minimal cost fulfilling our initial requirements about the system being light-weight to reside in the ADs memory. The results are for one individual model therefore the resulting storage cost is K times the initial one. However, the cost incurred is not preventive as the benefit of decreasing latency times, offloading queries otherwise issued to the cluster and no extra monetary cost are far greater.

	Size (KB)	TT (s)	PT (ms)
XGB	65.42 ± 4	52.58 ± 90	0.008 ± 0.005201

TABLE I
PERFORMANCE AND STORAGE RESULTS ACROSS MODELS

D. Adaptivity

To examine our CDM, ADM due to concept drift we devised the following experiment: consider a $\mathcal{M}_i$ that has already learned a particular distribution of y being deployed to answer queries. Our aim is to examine whether CDM detects a query pattern shift from one distribution to another. If remained undetected, it will cause detrimental problems in accuracy due to different distributions of y. We first set the detection threshold $h = 3\sigma_{\tilde{u}}$ and convergence threshold $c = 0.008$. We gradually introduce new query patterns and compare our system with an approach where no adaptation is deployed. Figure 3 shows the different queries being processed by our mechanism and the associated *true* prediction error. We first measure the error of queries using the *known* distribution until $t = 66$. From that point onward, we shift to the unknown distribution and evidently the error increases dramatically should no adaptation mechanism be employed. On the other hand, CDM detects that a shift has happened and transits the system from prediction mode to buffering mode until the exiting criteria are met. At the end, a new model is introduced which is trained using the new distribution as evidenced by the decreased error at $\mathcal{M}_{new}$.

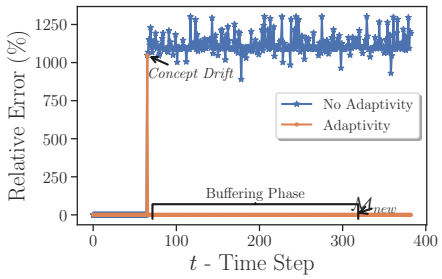

Fig. 3. Error with concept drift detection/adaptation.

VI. Related Work

Our work is related to prior work in analytical-query processing and in applied ML research communities and to prior work focusing on the benefits of the query-driven approach in analytical query processing and tuning [5], [6], [16]. Analytical queries nowadays are executed over underlying systems that provide either exact answers [20] or approximate answers [4], [14], [18] working over large big data clusters in CS requiring several orders of magnitude longer query response times. The contributions in this work are largely complementary to all this work. Specifically, during the training phase and in the adapting phase the system proposed here can be supported either by an exact or an approximate query processing engine. In addition, what makes our solution different is that it can be stored locally on an analyst's device as it has low storage overhead and also requires no communication to the cluster.

Query-driven models are largely being deployed for both aggregate estimation [5], [6] and for hyper-tuning [22] database systems. Unlike [5], [6] our focus is on a wide variety

of aggregate operators and not just COUNT for selectivity estimation. Furthermore, we address the crucial problem of detecting query pattern changes and adapting to them, which (to our knowledge) has not been addressed in this context before. Hence,our framework can be leveraged by all query-driven implementations in cases of dynamic/non-stationary workloads.

Moreover, concept drift adaptation is well understood [11], [21], mostly dealing with classification tasks, where classifiers adapt to new classes. We utilise concept drift in the context of query-driven analytical processing, relying on *explicit partitioning* [11], ensuring it avoids destructive forgetting while remaining accurate of previously learned query patterns. It is also favorable given our initial off-line design which already uses partitioning for clustering the query patterns and learning local models in given sub-spaces. Our work contributes with monitoring and detecting real-time query patterns change based on *approximating* the prediction error, which differentiates with the previous concept drift methods by measuring the actual error; evidently, this is not applicable in our case. Finally, we propose a novel reciprocity-driven adaptation mechanism in which we set a mechanism deciding when a new model should be trained engaging the knowledge derived from other possibly changing models in the CS.

VII. Conclusions

In this work we contribute a novel framework for adapting trained models under concept drift. We focus on models used for estimating analytical query answers efficiently and accurately, however we note that the framework is applicable in other domains as well. The contributions centre on a novel suit of ML models, which mine past and new queries and incrementally build models over quantized query-spaces using a vectorial representation. The described mechanisms (ADM and CDM) bear the ability to adapt under changing analytical workloads, while maintaining high accuracy of estimations. As shown by our evaluation (using real and synthetic datasets), the proposed approach enjoys high accuracy (well below 10% relative error) across all aggregate operators, with low response times (well below a millisecond) and low storage and training-time overheads. The contributed CDM,ADM mechanisms are able to detect changes using estimated errors and swiftly adapt. Furthermore, as more queries are processed our system has the potential to reach global convergence as no more query patterns remain undiscovered. This can significantly reduce unnecessary communication to cloud providers thus reduce network load and monetary costs.

VIII. Acknowledgement

We would like to thank Dr Kostas Kolomvatsos for his contributions to this work. This research received funding from the European's Union Horizon 2020 research and innovation programme under the grant agreement No. 745829. The author is funded by an EPSRC scholarship.

References

[1] Crimes - 2001 to present. URL: https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2, 2018. Accessed: 2018-08-10.

[2] Growing networks. URL: https://github.com/Skeftical/GrowingNetworks, 2018. Accessed: 2018-08-10.

[3] Intel lab data. URL: http://db.csail.mit.edu/labdata/labdata.html, 2018. Accessed: 2018-08-10.

[4] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. Blinkdb: queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 29–42. ACM, 2013.

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

[6] C. Anagnostopoulos and P. Triantafillou. Query-driven learning for predictive analytics of data subspace cardinality. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(4):47, 2017.

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

[8] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[9] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar. Learning in non-stationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.

[10] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[11] A. Gepperth and B. Hammer. Incremental learning algorithms and applications. In *European Symposium on Artificial Neural Networks (ESANN)*, 2016.

[12] O. A. Grigg, V. Farewell, and D. Spiegelhalter. Use of risk-adjusted cusum and rsprtcharts for monitoring in medical contexts. *Statistical methods in medical research*, 12(2):147–170, 2003.

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

[14] S. Kandula, A. Shanbhag, A. Vitorovic, M. Olma, R. Grandl, S. Chaudhuri, and B. Ding. Quickr: Lazily approximating complex adhoc queries in bigdata clusters. In *Proceedings of the 2016 International Conference on Management of Data*, pages 631–646. ACM, 2016.

[15] Z. Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. *IEEE Transactions on Visualization & Computer Graphics*, (1):1–1.

[16] L. Ma, D. Van Aken, A. Hefny, G. Mezerhane, A. Pavlo, and G. J. Gordon. Query-based workload forecasting for self-driving database management systems. In *Proceedings of the 2018 International Conference on Management of Data*, pages 631–645. ACM, 2018.

[17] S. Marsland, J. Shapiro, and U. Nehmzow. A self-organising network that grows when required. *Neural networks*, 15(8-9):1041–1058, 2002.

[18] Y. Park, B. Mozafari, J. Sorenson, and J. Wang. Verdictdb: universalizing approximate query processing. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1461–1476. ACM, 2018.

[19] F. Savva, C. Anagnostopoulos, and P. Triantafillou. Adaptive learning of aggregate analytics under dynamic workloads. *arXiv preprint arXiv:1908.04772*, 2019.

[20] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2):1626–1629, 2009.

[21] A. Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106(2), 2004.

[22] D. Van Aken, A. Pavlo, G. J. Gordon, and B. Zhang. Automatic database management system tuning through large-scale machine learning. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1009–1024. ACM, 2017.

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