Learning to Accurately COUNT with Query-Driven Predictive Analytics

Christos Anagnostopoulos
School of Computing Science
University of Glasgow, UK, G12 8QQ
christos.anagnostopoulos@glasgow.ac.uk

Peter Triantafillou
School of Computing Science
University of Glasgow, UK, G12 8QQ
peter.triantafillou@glasgow.ac.uk

***Abstract*—We study a novel solution to executing aggregation (and specifically COUNT) queries over large-scale data. The proposed solution is generally applicable, in the sense that it can be deployed in environments in which data owners may or may not restrict access to their data and allow only 'aggregation operators' to be executed over their data. For this, it is based on predictive analytics, driven by queries and their results. We propose a machine learning (ML) framework for the task (which can be adapted for different aggregates as well). We focus on the widely used set-cardinality (i.e., COUNT) aggregation operator, as it is a fundamental operator for both internal data system optimisations and for aggregation-query analytics. We contribute a novel, query-driven ML model whose goals are to: (i) learn the query space (access patterns), (ii) associate (complex) aggregation queries with the cardinality of their results, (iii) define query similarity and use it to predict the cardinality of the answer set of an ad-hoc incoming query. Our ML model incorporates incremental learning algorithms for ensuring high prediction accuracy even when both the querying patterns and the underlying data change. The significance of contribution lies in that it (i) is the only query-driven solution applicable over general environments which include restricted-access data, (ii) offers incremental learning adjusted for arriving ad-hoc queries, which is well suited for big data analytics, and (iii) offers a performance (in terms of prediction accuracy and time, and memory requirements) that is superior to data-centric approaches. We provide a comprehensive performance evaluation of our model, evaluating its sensitivity and comparative advantages versus acclaimed data-centric methods (self-tuning histograms, sampling, and multidimensional histograms).**

I. Introduction

In modern big data analytics applications, supported by big data infrastructures, predictive analytics [2], [3] and exploratory analysis are commonly based on statistical aggregation operators over the results of 'complex' queries [5]. Such complex, aggregate queries typically involve large datasets (which may themselves be the result of linking of other different datasets) and a number of range predicates over multidimensional vectors, structured, semi- and unstructured data. Query-driven data exploration and predictive learning is becoming increasingly important in the presence of large-scale data [7] since predicting aggregations over range predicate queries is a fundamental data exploration task [8] in big data systems. Frequently, data analysts and statisticians are in search of (approximate and/or partial) answers to such queries over unknown data subspaces (knowledge discovery). Imagine exploratory and predictive analytics [4] based on a *stream* of such aggregation operators over data subspaces being issued, until the scientist extracts sufficient statistics or learns local statistical characteristics, e.g., coefficient of determination and product-moment correlation coefficient, of the subspaces of interest.

In modern big data systems, often data to be analyzed possibly extends over a large number of federated data nodes, perhaps even crossing different administration domains and/or where data owners (nodes) may only permit restricted accesses (e.g., aggregations) over their data. Similarly, in the modern big data era, large datasets are often stored in the cloud. Hence, even when access is not restricted, accesses to raw data needed to answer aggregate queries are costly money-wise. Hence, predictive/exploratory analytics solutions which are widely applicable, even in such scenarios, are highly desirable.

Consider a d-dimensional data space $\mathbf{x} \in \mathbb{R}^d$.

Definition 1: Let a d-dim. box be defined by two boundary vectors $[a_1, \dots, a_d]^\top$ and $[b_1, \dots, b_d]^\top$, $a_i \leq b_i$, $a_i, b_i \in \mathbb{R}$. A predicate range query is represented by the $2d$-dimensional vector $\mathbf{q} = [a_1, b_1, a_2, b_2, \dots, a_d, b_d]^\top$ where a_i and b_i is lower and higher value, respectively, for the i-th dimension. Query $\mathbf{q}$ is a hyper-rectangle with faces parallel to the axes.

Definition 2: Given a predicate range query $\mathbf{q}$ and a dataset $\mathcal{B}$ of data points $\mathbf{x} \in \mathbb{R}^d$, $y \in \mathbb{N}$ is the cardinality of the answer set of those $\mathbf{x} \in \mathcal{B}$ in the interior of the hyper-rectangle defined by query $\mathbf{q}$ satisfying $a_i \leq x_i \leq b_i$, $\forall i$.

The reason we focus on the COUNT aggregation operator is that the answer Set Cardinality Prediction (SCP) of a predicate multidimensional (m-d) range query is a fundamental task, playing a central role in predictive analytics. With m-d range queries, analysts define the subspaces of interest within the overall data space. SCP in such subspaces then becomes important for data mining, query-driven data exploration, time series analysis, and big data visualization tasks [8], [4] of data (sub)spaces of interest. In exploratory and predictive analytics, a data scientist routinely defines specific regions of a large dataset that are worth exploring and wishes to derive and predict statistics over the populations of these regions – which amounts to SCP of the corresponding range queries. In addition to being an important aggregation operator, in database systems SCP (which amounts to the well known selectivity estimation problem) is explicitly used for query processing optimization, empowering query optimizers to choose, for instance, the access plan which produces the smallest intermediate-query results (which have to be retrieved from disks and communicated over the network) saving time, resource waste, and money (e.g., in Clouds). Furthermore, SCP is a core 'operator' in modern big data frameworks: For instance, in Spark [1] one of the five fundamental actions defined is the so-called *count* action, which is executed over the underlying raw data at each data node.

Motivation: Well-established and widely adopted techniques for Approximate aggregation-Query Processing (AQP) based on sampling, histograms, self-tuning histograms, wavelets, and sketches [6] have been proposed. Their fundamental and naturally acceptable assumption is that the underlying data are always accessible and available, thus it is feasible to create and maintain their statistical structures. For instance, histograms [9] require scanning of all data to be constructed and being up-to-date; the self-tuning histograms [10] require additionally the execution of queries to fine tune their statistical structures; the sampling methods [14] execute the queries over the sample to extrapolate the SCP result.

Consider now a big data environment, where a federation of data nodes store large datasets. There are cases where the data access to these nodes' data may be either restricted, (e.g., government medical and DNA databases and demographic and neighborhood statistic datasets). Furthermore, many real-world large-scale data systems may limit the number of queries that can be issued and/or charge for excessive data accesses. For example, there may exist per-IP limits (for web interface queries) or per developer key limits (for API based queries). Even when the (daily) limit is high enough, repeated executions actually have high monetary cost (e.g., in cloud deployments), waste communication overhead due to remote query execution, and computational resources. The accessed data nodes can either fully execute the queries (to produce exact results) or locally deploy an AQP technique to produce estimates. In the latter case, we must rely upon the SCP accuracy provided by the applied traditional AQP technique.

The above discussion raises the following desiderata: it is important to develop AQP techniques that (1) are applicable to all data-environment scenarios (restricted-access or not), (2) are inexpensive (i.e., avoid relying on excessive querying of and communication with the data nodes), while (3) offering high prediction accuracy, and (4) being prudent in terms of compute-network-store resource utilization.

Let us consider an indicative baseline solution for AQP in our environment. One approach is to store, e.g., locally to a central node, all the AQP structures (e.g., histograms, samples, sketches, etc.) from a federation of data nodes. Thus, we can simply locally access this node for SCP. Firstly, this violates our first desideratum, as privacy issues emerge (data access restrictions). Obviously, retaining all AQP structures, provides one with the *whole* valuable information about the underlying data (e.g., in the case of histograms, we obtain the underlying probability data distribution $p(\mathbf{x})$, while in sampling methods we retain actual samples from the remote datasets). Even, in cases where the local accesses to AQP structures were secured (which is again subject to major security concerns), we would have to cope with the problem of AQP structure updates. The maintenance of those structures in the face of updates demands high network bandwidth overhead, cost for data transfer (in a Cloud setting), latency for communicating with the remote nodes during updates of the underlying dataset at these nodes, and scalability and performance bottleneck problems arise at the central node. Therefore, this approach does not scale well and can be expensive, violating our 2nd and 3rd criteria above. An alternative baseline solution would be to do away with the central node and send the query to the data nodes, which maintain traditional AQP statistical structure(s) and send back their results to the querying node. As before, this violates many of our desiderata. It is not applicable to restricted-access scenarios (violating criterion 1) and involves heavy querying of the data node (violating criteria 2 and 4). Even if this was the case (by violating criteria 1, 2, and 4), the construction and maintenance of an AQP structure would become a prohibited solution; we struggle with huge volumes of data (data universe explosion phenomenon; imagine *only* the creation of a m-d histogram over 1 zettabyte). These facts help expose the formidable challenges to the problem at hand, (a significant problem for large-scale predictive analytics) which to the best of our knowledge, has not been studied before. In this work we study a query-driven SCP in a big data system taking into consideration the above-mentioned desiderata. Although significant data-centric AQP approaches for SCP have been proposed [6] a solution for our intended environments of use is currently not available.

There are three fundamental pressures at play here. The first pertains to the development of a solution for SCP that is efficient and scalable (especially for distributed scale-out environments, wherein extra communication costs, remote invocation techniques, and estimation latency are introduced) by the distribution of datasets and computation. The second pertains to the accuracy of SCP results, where as we shall see traditional solutions fall short. The third concerns the wide-applicability of a proposed method, taking into account environments where data accesses may be restricted, We propose a solution that addresses all these tensions. Conceptually, its fundamental difference from related works is that it is query-driven, as opposed to data-driven, and is thus based on a ML model (trained by a number of queries sent to a data node) and later utilized to predict answers to new incoming queries. More specifically, the challenging aim of our approach is to swiftly provide SCP of ad-hoc, new, unseen queries while (i) *avoiding executing them over a data node*, saving communication and computational resources and money, and (ii) not relying on any knowledge on the $p(\mathbf{x})$, and any knowledge about nodes' data. Through our query-driven SCP, an inquisitive data scientist, who explores data spaces, issues aggregate queries, and discovers hidden data insights, can extract accurate knowledge, efficiently and inexpensively.

Related Work: Given a d-dim. data space $\mathbf{x} \in \mathbb{R}^d$ the holy grail approaches focus on: (i) inspecting the (possibly huge) underlying dataset and estimate the underlying probability density function (pdf) $p(\mathbf{x})$. Histograms (typically m-d) as fundamental data summarization techniques are the cornerstone, whereby the estimation of $p(\mathbf{x})$ is highly exploited for SCP of range queries, e.g., [9], [10]. The traditional methods of building histograms do not scale well with big datasets. Histograms need to be periodically rebuilt in order to update $p(\mathbf{x})$ thus, exacerbating the overhead of this approach [17]. Central to our thinking is the observation that a histogram is constructed solely from data, thus obviously being not applicable to our problem for the above-mentioned reasons. Histograms are also inherently unaware on the SCP requests, i.e., query patterns. Their construction method rely neither on query distribution $p(\mathbf{q})$ nor on joint $p(\mathbf{q}, y)$ but only on $p(\mathbf{x})$. As a result, such methods do not yield the most appropriate histogram for a given $p(\mathbf{q})$ [18]. The limitations of this method are also well-known [19], [20]. To partially address some of the above limitations, prior work has proposed self-tuning histograms (STHs) e.g., [10], [19]. The STHs learn a centrally stored dataset from scratch (i.e., starting with no buckets) and rely only on the cardinality result provided by the execution of a query, referred to as Query Feedback Records (QFR). STHs exploit the actual cardinality from QFR and use this information to build and refine traditional histograms. Formally, given a query $\mathbf{q}$ over data with cardinality y, the methods of STHs estimate the conditional $p(\mathbf{x}|y, \mathbf{q})$ since the main purpose is to construct and tune an histogram conditioned on query patterns. Fundamentally, the limitations in STHs in our problem stem from the fact that they estimate $p(\mathbf{x}|y, \mathbf{q})$, thus, having to data access (in m-d STHs, at least one scan of the set $\mathcal{B}$ is required), deal with the underlying data distribution and make certain assumptions of the statistical dependencies of data. Other histogram-based SCP methods utilize wavelets [11], singular value decomposition [12], value transformations [13], and entropy-based [16]; the list is not exhausted. Briefly, the idea is to apply wavelet decomposition to the dataset to obtain a compact data synopsis based on the wavelet coefficients. By nature, wavelets-based AQP relies on the synopsis construction over data thus could not be applied to our problem. Overall, STHs and the other advanced histogram-based approaches, are associated with data access for estimating $p(\mathbf{x})$ or any other $p(\mathbf{x}|\mathbf{q}, \ldots)$ thus not applicable in our problem. There are also SCP methods e.g., [25] that use the dataset's fractal dimensions. They rely on certain assumptions on the density of data points in the dataset $\mathcal{B}$ and require data access to construct their structures. Sampling methods [14], [15] have been also proposed for SCP. They share the common idea to evaluate the query over a small subset of the dataset and extrapolate the observed cardinality. Finally, another approach for AQP answering to SCP is data sketching; we refer the reader to [6] for a useful survey of sketching techniques. Sketching algorithms construct estimators from the raw data and yielding a function of these estimators as the answer to the query. Therefore, as discussed above, we neither have access to data nor to a sample of them, thus yielding the data sketching and sampling methods inapplicable to our problem.

In conclusion, the data-centric approaches in related work are not applicable to our problem since they require explicit access to data to construct their AQP structures and maintain them up-to-date. For this reason, our proposed solution to this novel setting is query-driven. Our model can be highly useful when it is very costly (in time, money, communication bandwidth) to execute aggregation operators over the results of complex range queries (including joins of datasets and arbitrary selection predicates), when data are stored at the cloud, or at federations of data stores, across different administration domains, etc. And, to our knowledge, it is the only work that can address this problem setting. However, to assess the performance of our solution, we provide extensive experiments comparing our approach versus histograms, popular self-tuning histograms, and sampling methods. We show that our query-driven solution, which extracts knowledge from the issued queries and corresponding answers, provides higher SCP accuracy and performance, while being more widely applicable.

II. Challenges & Overview

Naturally, our approach is query-driven. The first requirement (and challenge) of our approach is to incrementally learn the query patterns $p(\mathbf{q})$ at any time, thus being able to (i) detect possible changes to user interests on issuing queries and (ii) reason about the similarity between query patterns. The second requirement (and challenge) is to learn the association $\mathbf{q} \rightarrow y$ between a query $\mathbf{q}$ and its aggregate result (here, cardinality) y, i.e., $p(y|\mathbf{q})$, thus being able to predict the cardinality. The third requirement (and challenge) is to learn such association without relying on the underlying $p(\mathbf{x})$ which in our case is totally unknown and inaccessible. The fourth requirement (and challenge) is to update $p(\mathbf{q})$ and $p(\mathbf{q}, y)$ based on changes in query patterns and to data. Query distributions are known to be non-uniform, with specific portion of the data space being more popular. However, query patterns change with time, reflecting changes of users interests to exploring different sections of the datasets of nodes. Hence, we must swiftly adapt and learn on-the-fly the *new* query patterns, updating $p(\mathbf{q}, y)$ and $p(\mathbf{q})$. Furthermore, updates on the underlying datasets of nodes can independently occur, altering $p(\mathbf{x})$. We must also deal with such mutations, implying the need to maintain the current $\mathbf{q} \rightarrow y$ association, subject to updates of the underlying data. We require a model to meet the above-mentioned requirements.

Overview of COUNT Predictive Learning: Consider a set $\mathcal{Q} = \{(\mathbf{q}_i, y_i)\}_{i=1}^{n}$ of training pairs and a new query $\mathbf{q}$ with actual result y. Our major aim is to predict its result $\hat{y}$ using only $\mathcal{Q}$ without actually executing $\mathbf{q}$. Let us discuss some baseline solutions: A first idea is to keep all pairs $(\mathbf{q}_i, y_i)$ and given $\mathbf{q}$ we find the most *similar* query $\mathbf{q}_j$ with respect to Euclidean distance and predict $\hat{y} = y_j$, with $(\mathbf{q}_j, y_j) \in \mathcal{Q}$. We can also involve the k closest queries to $\mathbf{q}$ and average their cardinalities, e.g., k-nearest neighbors regression. The major problems here are: (i) we must store and search all previous pairs for each new query; $\mathcal{Q}$ can be huge. Deciding which pairs to discard is not a trivial task (a new pair might convey useful information while another new one might be a redundant / repeated query); (ii) when data change (updates on raw data), which impacts the query results, it is not trivial to determine which pairs from $\mathcal{Q}$ and how many to update. Even worse, all pairs may need updating; (iii) when query patterns change (new user interests), then there may be many pairs in $\mathcal{Q}$ that will not contribute to cardinality prediction (the new queries are actually far distant to the previous ones) or even negatively impact the final result. To avoid such problems we extract knowledge from $\mathcal{Q}$ as to how query and cardinality depend on each other. We could cluster similar queries given the Euclidean distance, thus forming a much smaller set $\mathcal{L}$ of representative (prototype) queries $\mathbf{w}$ with $|\mathcal{L}| \ll |\mathcal{Q}|$. For instance, $\mathbf{w} \in \mathcal{L}$ can be the centroid of those queries from $\mathcal{Q}_{\mathbf{w}} \subset \mathcal{Q}$ with distances from $\mathbf{w}$ be the smallest among all other representatives. However, we are not just interested in clustering $\mathcal{Q}$. We should partition $\mathcal{Q}$ aiming at cardinality prediction. An approach could be to assign to each $\mathbf{w}_i \in \mathcal{L}$ a 'representative' cardinality value, e.g., the average cardinality of those queries that belong to $\mathcal{Q}_{\mathbf{w}_i}$. Once this assignment is achieved, we only keep $\mathcal{L}$ and discard $\mathcal{Q}$. Nonetheless, our requirements include incremental learning of the query space in light of cardinality prediction. We require an adaptive clustering algorithm that incrementally, i.e., with only one pass of $\mathcal{Q}$, quantizes $\mathcal{Q}$ but also with respect to minimizing the prediction error.

Also, the adoption of an on-line quantization algorithm, like on-line k-means is not directly applicable in our case as we don't wish to simply quantize the query space; we explicitly require quantization of the query space *in light of cardinality prediction*. Moreover, on-line regression methods, e.g., incremental regression trees [26], on-line support vector regression [27], could not fulfill all requirements. This is because, we also deal with the fact that queries are continuously observed, conveying the way users are interested in data exploration. The capability of the model to adapt to such changes requires explicit information on accessing the very specific regions of the query patterns space; this is neither easily provided nor supported by incremental regression methods. Moreover, the problem here is not only to adapt to changes on the query patterns but, if so, to decide which and how representative(s) or regions of the query patterns space to update upon data and/or query updates.

Our Predictive Learning Approach & Contribution: We address the above problems bearing in mind the requirements of (i) incremental learning of the query patterns and (ii) incremental learning of the association between query and cardinality utilized for prediction. To this end, we attempt to combine, in parallel, two incremental learning processes: adaptive, self-organized vector quantization of the query patterns space (which will also give insights to which regions of the query patterns space to adapt) taking also into account the objective to minimize the SCP error. On the one hand, we learn $p(\mathbf{q})$ and in parallel, on the other hand, we learn $p(y|\mathbf{q})$ reflected by the $\mathbf{q} \rightarrow y$ association. We move beyond a typical on-line quantization (partitioning) of $\mathcal{Q}$ going toward a prediction error-driven, incremental, self-organized cluster formation of each $\mathcal{Q}_{\mathbf{w}}$. Specifically, given a new query $\mathbf{q}$ we predict $\hat{y}$ through locally weighted regression (using Kernel functions) over those assigned representative cardinality values whose corresponding prototypes $\mathbf{w}_i$ are *topologically* close to $\mathbf{q}$. The regression weights quantify a topology information obtained by the self-organizing neural maps [22]. Through this topological information, we can find which prototypes from $\mathcal{L}$ are similar to $\mathbf{q}$, thus, use their assigned cardinalities for prediction. Moreover, by learning the $\mathbf{q} \rightarrow y$ association (a.k.a. heteroassociative learning [22]) we rest on the fact that similar queries correspond to close cardinalities. Based on this, the update of the most similar query prototypes to an incoming query $\mathbf{q}$ (either during learning or adaptation due to data change) takes into account both the query $\mathbf{q}$ itself and the feedback of the prediction error of the actual result of query $\mathbf{q}$ and the currently estimated result of the model. Hence, the query prototypes in $\mathcal{L}$ are partitioned with respect to the cardinality prediction error, thus, yielding a mechanism ideal for our use case.

We introduce a novel Machine Learning (ML) model $\mathcal{M}$ that incrementally extracts information about the $\mathbf{q} \rightarrow y$ association by learning $p(\mathbf{q})$ and, in parallel, $p(y|\mathbf{q})$. Once trained, model $\mathcal{M}$ predicts the cardinality of an *unseen* query without requesting its execution. Our model also swiftly adapts to *new* query patterns, and deals efficiently with data updates. The major technical contributions are:

- a novel incremental, self-organized, prediction error-driven, vector quantization and regression model for predicting the aggregate results of range queries. This model adopts stochastic gradient descent, thus we provide theoretical analysis and proof of convergence over large-scale loss minimization;
- adaptation methods based on stochastic gradient descend in the face of updates to query patterns and/or data updates;
- comprehensive experimental results analyzing the performance of our model and showcasing its benefits vis-à-vis data-centric sampling [15], histograms [9] and STHs [19], [20].

III. Preliminaries & Problem Formulation

We overview the essentials of our ML model, namely Unsupervised Competitive Learning (UCL) [21] for adaptive query vector quantization and Heteroassociative Competitive Learning (HCL) [22] for associating queries to cardinalities. UCL partitions a query pattern space $\mathbb{R}^{2d}$ characterized by an unknown $p(\mathbf{q})$, $\mathbf{q} \in \mathbb{R}^{2d}$. A prototype or *neuron* $\mathbf{w}_j$ represents a local region of $\mathbb{R}^{2d}$ and behaves as a quantization vector. UCL distributes M neurons $\mathbf{w}_1, \ldots, \mathbf{w}_M$ in $\mathbb{R}^{2d}$ to approximate $p(\mathbf{q})$. A UCL model learns as $\mathbf{w}_j$ changes in response to random training patterns. Competition selects which $\mathbf{w}_j$ the training pattern $\mathbf{q}$ modifies. Neuron $\mathbf{w}_j$ wins if it is the closest (based on 2-norm distance $\|\mathbf{q} - \mathbf{w}_j\|_2$) of the M neurons to $\mathbf{q}$. During the learning phase of UCL, patterns $\mathbf{q}$ are projected onto their winning neurons, which competitively and adaptively move around the space to form optimal partitions that minimize the quantity $\int \|\mathbf{q} - \mathbf{w}_j\|_2^2 p(\mathbf{q}) \mathrm{d}\mathbf{q}$, i.e., vector quantization error, with winning neuron $\mathbf{w}_j$ such that $\|\mathbf{w}_j - \mathbf{q}\|_2 = \min_i \|\mathbf{w}_i - \mathbf{q}\|_2$. The neurons upon a t-th training pattern $\mathbf{q}$ are incrementally updated as follows $\Delta\mathbf{w}_j = \beta(t)\,(\mathbf{q} - \mathbf{w}_j)$ and $\Delta\mathbf{w}_i = \mathbf{0}$, if $i \neq j$, where learning rate $\beta(t) \in (0, 1]$ slowly decreases with the update step. Kohonen's self-organizing map (SOM) [22] is an advanced variant of a UCL for vector quantization, in which $\mathbf{w}_j$ corresponds to the j-th position $\mathbf{r}_j = [r_{j1}, r_{j2}]$ of a 2-dim. square lattice/matrix $\mathcal{L}$ (we notate $\mathbf{w}_j \in \mathcal{L}$). In SOM, neurons that are topologically close in the lattice correspond to patterns that are also close in $\mathbb{R}^{2d}$. This way a *topographic* mapping is learned between query pattern and lattice space. This is achieved by adapting not only the winner neuron $\mathbf{w}_j$ of a pattern $\mathbf{q}$ but also its topographical neighbors $\mathbf{w}_i$ to some degree through a Kernel distance function $h(i, j; t)$ over the positions $\mathbf{r}_i$ and $\mathbf{r}_j$ of neurons $\mathbf{w}_i$ and $\mathbf{w}_j$ in $\mathcal{L}$, respectively. Usually, $h(i, j; t)$ is a Gaussian neighborhood function $h(i, j; t) = \exp\left(-\frac{\|\mathbf{r}_i - \mathbf{r}_j\|_2^2}{2\rho^2(t)}\right)$. The topological neighborhood is symmetric around the winner neuron, which has the maximum value of unity. Parameter $\rho(t)$ is the width of the neighborhood with initial value ρ_0 defined as $\rho(t) = \rho_0 \exp(-\frac{t}{T_\rho})$, where T_ρ is a constant. A small width value corresponds to narrow neighborhood. We obtain SOM through an incremental update rule that adapts all neurons that are topographically close to $\mathbf{w}_j$: $\Delta\mathbf{w}_i = \beta(t) h(i, j; t)\,(\mathbf{q} - \mathbf{w}_i)\,, \forall i$. A good choice of $\beta(t)$ improves significantly the convergence of SOM [22]; usually $\beta(t) = \frac{\beta(t-1)}{1+\beta(t-1)}$ with $\beta(0) = 1$. SOM yields a high quality vector quantization from all UCL variants because of producing a structured *ordering* of the pattern vectors, i.e., similar query patterns are projected to similar neurons, making it ideal for our purposes. UCL/SOM does not learn any conditional or joint association between different pattern spaces. In our case, we desire also to estimate an association between $\mathbb{R}^{2d}$ and $\mathbb{N}$, i.e., estimate $p(\mathbf{q}, y)$ with $\mathbf{q} \in \mathbb{R}^{2d}, y \in \mathbb{N}$, HCL comes into play. HCL estimates indirectly an unknown joint $p(\mathbf{q}, y)$, while directly estimates a function $f : \mathbb{R}^{2d} \rightarrow \mathbb{N}$ over random pairs $(\mathbf{q}, y)$. In statistical learning theory [21], HCL refers to a function estimation model $\mathcal{M}(f, \alpha)$ (or simply $\mathcal{M}$) with parameter $\alpha \in \Lambda$ (Λ is a parameter space defined later) for estimating f. The problem of learning $\mathcal{M}$ is that of choosing from a set of functions $f(\mathbf{q}, \alpha), \alpha \in \Lambda$, the one which minimizes the risk function $\mathcal{J}(\alpha) = \int L(y, f(\mathbf{q}, \alpha)) \mathrm{d}p(\mathbf{q}, y)$ given random pairs $(\mathbf{q}, y)$ drawn according to $p(\mathbf{q}, y) = p(\mathbf{q})p(y|\mathbf{q})$ with *loss* or estimation error $L(y, \hat{y})$ between actual y and predicted $\hat{y} = f(\mathbf{q}, \alpha)$, e.g., $L(y, \hat{y}) = |y - \hat{y}|$. The goal for HCL is to learn $\mathcal{M}(f, \alpha_0)$ which minimizes $\mathcal{J}(\alpha)$ subject to unknown $p(\mathbf{q}, y)$, i.e., $\alpha_0 = \arg\min_{\alpha \in \Lambda} \mathcal{J}(\alpha)$. Stochastic gradient descent (SGD) is considered to be one of the best methods for large scale loss minimization and has been experimentally and theoretically analyzed by [28]. Upon the presence of a t-th pattern $(\mathbf{q}, y)$, $\alpha(t)$ is updated by $\Delta\alpha(t) = -\beta(t)\nabla L(y, \hat{y}; \alpha(t))$, where ∇L is the gradient of L at t-th pattern w.r.t. $\alpha(t)$.

Problem Formulation: Consider a model $\mathcal{M}$ that estimates the SCP function $f : \mathbb{R}^{2d} \rightarrow \mathbb{N}$ given training pairs $(\mathbf{q}, y)$ drawn from the unknown $p(\mathbf{q}, y)$, i.e., $y = f(\mathbf{q})$. Model $\mathcal{M}$ learns the mapping from query pattern space to cardinality domain by minimizing the risk function $\mathcal{J}(\alpha)$ with respect to loss $L(y, \hat{y})$. A loss function can be, e.g., λ-insensitive $L(y, \hat{y}) = \max\{|y - \hat{y}| - \lambda, 0\}, \lambda > 0$, 0-1 loss $L(y, \hat{y}) = I(y \neq \hat{y})$ with I be the indicator function, squared loss $(y - \hat{y})^2$, or absolute $|y - \hat{y}|$. The fundamental problems of the ML model for SCP are:

Problem 1: Given set $\mathcal{B}$ and pairs $(\mathbf{q}, y)$, incrementally learn a model $\mathcal{M}$ which minimizes $\mathcal{J}(\alpha)$.

$\mathcal{M}$ should adapt to changes in $p(\mathbf{q})$ and $p(y|\mathbf{q})$ since query distribution is not necessarily static.

Problem 2: Given a trained $\mathcal{M}$ and change in $p(\mathbf{q})$, derive an incremental algorithm for adapting $\mathcal{M}$'s parameter such that $\mathcal{M}$ minimizes $\mathcal{J}(\alpha)$.

$\mathcal{M}$ should be up-to-date as $\mathcal{B}$ gets updated, i.e., $p(\mathbf{q}, y)$ changes due to updates / insertions / deletions.

Problem 3: Given a trained $\mathcal{M}$ and updated $\mathcal{B}$, derive an incremental algorithm for adapting $\mathcal{M}$'s parameter such that $\mathcal{M}$ minimizes $\mathcal{J}(\alpha)$.

IV. SET CARDINALITY PREDICTIVE LEARNING

Our objective is a ML model $\mathcal{M}$ to incrementally (i) quantize (cluster) the query pattern space, (ii) learn the association $\mathbf{q} \rightarrow y$ and (iii) predict the set cardinality given an unseen query. The novelty of our model is the introduction of two *simultaneous* incremental learning tasks: Task 1: incremental query space quantization (UCL/SOM; unsupervised learning); Task 2: incremental *local* learning of the $\mathbf{q} \rightarrow y$ association within the region of these neurons (HCL; supervised learning). The parameters of the Task 1 are neurons $\mathbf{w}_j \in \mathcal{L}$ and the parameters of the Task 2 are the (local) cardinality prototypes y_j associated with each $\mathbf{w}_j$. The y_j reside on a *regression* lattice $\mathcal{C}$ such that the j-th index of $\mathbf{w}_j$ refers to the j-th index of y_j. The overall parameter set for model $\mathcal{M}$ is $\alpha = (\{\mathbf{w}_j\}, \{y_j\}), j = 1, \ldots, M$.

Figure 1 shows the idea of both simultaneous tasks. Consider the presence of a random pair $(\mathbf{q}, y)$. **Projection**: query $\mathbf{q}$ is projected onto its winner neuron $\mathbf{w}_j \in \mathcal{L}$. Based on Task 1, neuron $\mathbf{w}_j$ adapts (moves). **Association**: Simultaneously (Task 2) the corresponding prototype $y_j \in \mathcal{C}$ moves towards (gets updated) to y governed by a feedback update rule based on stochastic negative partial derivative (introduced later). Both tasks simultaneously converge; see Theorem 1. Neuron $\mathbf{w}_j$ converges to the centroid of a local region $\mathbb{D}_j \subset \mathbb{R}^{2d}$ (see Centroid Theorem in [21]) and its corresponding y_j converges to the median of the image of $\mathbb{D}_j$; see Theorem 2. **Prediction**: $\mathcal{M}$ based on $\mathcal{L}$ 'locates' the winner neuron and, based on $\mathcal{C}$, predicts the cardinality using Kernel regression over the cardinality prototypes. **Feedback:** The prediction result feeds $\mathcal{C}$ during learning for updating cardinality prototypes.

We adopt SOM for UCL since based on topology preservation we can claim that: *if queries* $\mathbf{q}$ *and* $\mathbf{q}'$ *are similar due to being projected onto the same neuron of* $\mathcal{L}$*, then their selectivities on* $\mathcal{C}$ *are likely to be similar, too*. This argument cannot be claimed by any other UCL method (e.g., k-means or fuzzy c-means clustering), which does not guarantee topological ordering of quantization vectors.

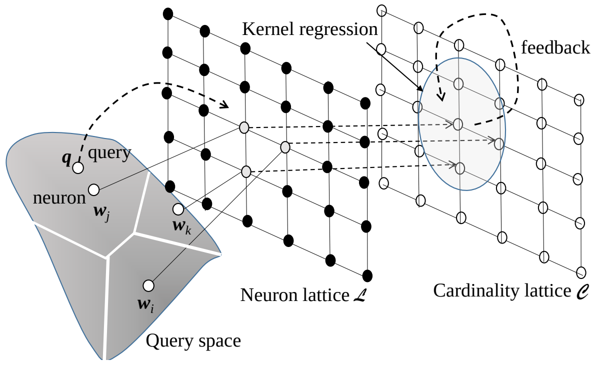


Fig. 1. Projection-association-prediction-feedback: Simultaneous UCL and HCL over lattices $\mathcal{L}$ and $\mathcal{C}$.

We firstly define the distance between queries, the SCP function $\hat{y} = f(\mathbf{q}, \alpha)$ and the loss function $L(y, \hat{y})$. Consider two range queries $\mathbf{q}, \mathbf{q}'$ normalized firstly in $[0, 1]^{2d}$ (only for simplicity in our analysis).

Definition 3: The normalized Euclidean distance between queries $\mathbf{q}$ and $\mathbf{q}'$ is $\|\mathbf{q} - \mathbf{q}'\|_2 = \frac{1}{\sqrt{2d}} \sum_{i=1}^{d} (a_i - a_i')^2 + (b_i - b_i')^2$, where $\frac{1}{\sqrt{2d}}$ is a normalization factor since $0 \leq \|\mathbf{q} - \mathbf{q}'\|_2 \leq \sqrt{2d}$.

When dealing with mixed-type data points, e.g., consisting of categorical & continuous attributes, we can adopt other distance metrics [24]; this does not spoil the generality of quantization. Consider the winner neuron $\mathbf{w}_j \in \mathcal{L}$ and its corresponding cardinality prototype $y_j \in \mathcal{C}$ to a random query $\mathbf{q}$. Neuron $\mathbf{w}_j$ and query $\mathbf{q}$ have the same dimensionality $2d$. The SCP f is not only based on y_j, but also on the contribution of neighboring y_i defined by a topographical neighborhood of winner $\mathbf{w}_j$. This is achieved by a kernel function $\mathcal{K}_\epsilon(\|\mathbf{r}_i - \mathbf{r}_j\|_2)$ over the normalized location vectors $\mathbf{r}_i$ and $\mathbf{r}_j$ (i.e., $\|\mathbf{r}_i\|, \|\mathbf{r}_j\| \leq 1$) of the associated neurons $\mathbf{w}_i$ and $\mathbf{w}_j$ of lattice $\mathcal{L}$, respectively. That is, $\hat{y} = f(\mathbf{q}, \alpha)$ is produced by the Nadaraya-Watson Kernel regression model:

$$\hat{y} = f(\mathbf{q}, \alpha) = \frac{\sum_{i=1}^{M} \mathcal{K}_\epsilon(\|\mathbf{r}_i - \mathbf{r}_j\|) y_i}{\sum_{i=1}^{M} \mathcal{K}_\epsilon(\|\mathbf{r}_i - \mathbf{r}_j\|)} \quad (1)$$

with $j = \arg\min_{\mathbf{w}_i \in \mathcal{L}} \|\mathbf{q} - \mathbf{w}_i\|_2$. Kernel $\mathcal{K}_\epsilon(x) = 0.75 \cdot \left(1 - (\frac{x-0.5}{\epsilon})^2\right) \cdot I(|x - \frac{1}{2}| \leq \epsilon)$ is the Epanechnikov kernel function shifted to 0.5 and scaled by $0 < \epsilon \ll 0.5$. Obviously other kernel functions can be also adopted e.g., uniform, triangular, quadratic, with Epanechnikov being most commonly used kernel. Predicted cardinality is estimated by a kernel smoothing of those cardinality prototypes whose associated neurons are topographically close (w.r.t. ϵ) to winner neuron. Given actual y and predicted $\hat{y}$ as in (1), we adopt the loss $L(y, \hat{y}) = |y - \hat{y}|$ because it is widely used for evaluating the prediction error in SCP as in [9], [18], and [19]. Topographically close neurons w.r.t. location vectors also imply close neurons w.r.t. Euclidean distance. However, the adoption of a Kernel function over the distance of neurons in $\mathbb{R}^{2d}$ could assume query components to be isotropically Gaussian, which is not a general case when d is relatively large. We distinguish three phases for the proposed ML model: the learning, prediction and update phase.

Learning Phase: In the learning phase of $\mathcal{M}$ we are given a sequence of pattern pairs $(\mathbf{q}(1), y(1)), (\mathbf{q}(2), y(2)), \ldots$ Query patterns $\mathbf{q}(t)$ are used for quantizing the query space (over $\mathcal{L}$) and cardinalities $y(t)$ are used for learning the $\mathbf{q} \rightarrow y$ association (over $\mathcal{C}$). Upon the presence of a pattern pair $(\mathbf{q}(t), y(t))$ the winner $\mathbf{w}_j(t) \in \mathcal{L}$ is determined by $j = \arg\min_{\mathbf{w}_i \in \mathcal{L}} \|\mathbf{q}(t) - \mathbf{w}_i(t)\|$. After the projection of $\mathbf{q}$ to winner $\mathbf{w}_j$, the model $\mathcal{M}$ updates in an incremental manner the winner and all its neighbors of lattice $\mathcal{L}$ such that they approach the query pattern $\mathbf{q}$ with a magnitude of $\beta(t)h(i,j;t)$. In the same time, cardinality y is used for updating the corresponding $y_j \in \mathcal{C}$ along with all prototypes $y_i \in \mathcal{C}$ associated with the neighbors of winner neuron $\mathbf{w}_j$ with the same magnitude. Therefore, the update rule for each y_i is governed by the loss function $L(y, \hat{y}) = |y - \hat{y}|$ we aim to minimize with $\hat{y}$ defined in (1).

Theorem 1: Given pattern pair $(\mathbf{q}(t), y(t))$, model $\mathcal{M}$ converges to the optimal parameter α, which minimizes the risk function $\mathcal{J}(\alpha)$ with respect to loss function $L(y, \hat{y}) = |y - \hat{y}|$ and $\hat{y}$ is defined in (1), if neuron $\mathbf{w}_i(t) \in \mathcal{L}$ and its associated prototype $y_i(t) \in \mathcal{C}$ are updated as

$$\Delta \mathbf{w}_i(t) = \beta(t)h(i,j;t)\left(\mathbf{q}(t) - \mathbf{w}_i(t)\right) \quad (2)$$

$$\Delta y_i(t) = \beta(t)h(i,j;t)\frac{\mathcal{K}_\epsilon(\|\mathbf{r}_i - \mathbf{r}_j\|)}{\sum_{k=1}^{M} \mathcal{K}_\epsilon(\|\mathbf{r}_k - \mathbf{r}_j\|)} \operatorname{sgn}\left(y(t) - \hat{y}(t)\right) \quad (3)$$

where sgn$(\cdot)$ is the signum function, $\beta(t)$ is the learning rate and $h(i,j;t)$ is the neighborhood function, j is the index of the winner neuron $\mathbf{w}_j(t)$ of pattern query $\mathbf{q}(t)$ and predicted $\hat{y}(t)$ is determined by (1).

Proof: We derive the analysis of convergence corresponding to lattices $\mathcal{L}$ and $\mathcal{C}$. We verify whether quantization error $\|\mathbf{w} - \mathbf{q}\|_2^2$ and loss $L(y, \hat{y}) = |y - \hat{y}|$ actually decreases as the learning phase proceeds, converging eventually to a stable state. The convergence is evaluated through the average expected loss $\mathcal{E} = \int_{\mathcal{W}} \sum_{\mathbf{w}_i \in \mathcal{L}} h(i,j)\|\mathbf{w}_i - \mathbf{q}\|_2^2 \mathrm{d}p(\mathcal{W}) + \int_{\mathcal{Y}} \sum_{y_i \in \mathcal{C}} h(i,j)|y - \hat{y}|\mathrm{d}p(\mathcal{Y})$ being taken over an infinite sequence of $\mathcal{W} = \{\mathbf{q}(1), \mathbf{q}(2), \ldots\}$ and corresponding $\mathcal{Y} = \{y(1), y(2), \ldots\}$ and $p(\mathcal{W})$, $p(\mathcal{Y})$ is the pdf of $\mathcal{W}$ and $\mathcal{Y}$, respectively. Since both pdfs are unknown and sequences $\mathcal{Y}$ and $\mathcal{W}$ are actually finite we use the Robbins-Monro (RM) [23] stochastic approximation for $\mathcal{E}$ minimization to find an optimal value for each $\mathbf{w}_i, y_i, i = 1, \ldots, M$. Based on RM the stochastic sample $E(t)$ of $\mathcal{E}$ is $E(t) = \sum_{\mathbf{w}_i \in \mathcal{L}} h(i,j;t)\|\mathbf{w}_i(t) - \mathbf{q}(t)\|_2^2 + \sum_{y_i \in \mathcal{C}} h(i,j;t)|y(t) - \hat{y}(t)|$. $E(t)$ has to decrease at each new pattern at t by descending in the direction of its (partial) negative gradient. Hence, the SGD rule for each $\mathbf{w}_i$ is $\Delta \mathbf{w}_i(t) = -\frac{1}{2}\beta(t)\frac{\partial E(t)}{\partial \mathbf{w}_i(t)}$ and for y_i is $\Delta y_i(t) = -\beta(t)\frac{\partial E(t)}{\partial y_i(t)}$, where $\beta(t)$ satisfies $\sum_{t=0}^{\infty} \beta(t) = \infty$ and $\sum_{t=0}^{\infty} \beta^2(t) < \infty$ [21]. From the partial derivatives of $E(t)$ we obtain the update rules (2) and (3) for parameter set α. ■

Note that the update rule (3) for prototypes $y_i(t)$ involves the current prediction $\hat{y}(t)$ of the model during the t-th pattern pair in the learning phase. Naturally we update each $y_i(t)$ in an on-line supervised regression fashion, in which we take the prediction $\hat{y}(t)$ in (1) as feedback. From (3) we observe that neighbor $y_i(t)$ of $y_j(t)$ is adapted by its relative contribution provided by the kernel function, which is rational since $y_i(t)$ contributes with the same magnitude to the SCP. If $y(t) > \hat{y}(t)$, then $y_i(t)$ increases linearly with its contribution to SCP approaching the actual $y(t)$. On the other hand, i.e., $y(t) < \hat{y}(t)$, $y_i(t)$ decreases to move away from $\hat{y}(t)$ and approaches $y(t)$. When the current prediction error is zero, i.e., $L(y(t), \hat{y}(t)) = |y(t) - \hat{y}(t)| = 0$, there is no update on the cardinality prototypes. Neuron $\mathbf{w}_i(t)$ moves toward pattern query $\mathbf{q}(t)$ to follow the trend. Obviously, the more similar a pattern query $\mathbf{q}$ and a neuron $\mathbf{w}_i$ are, the less $\mathbf{w}_i$ gets updated. If ϵ is selected such that $\mathcal{K}_\epsilon(\|\mathbf{r}_i - \mathbf{r}_j\|) = 0, i \neq j$, then we obtain $\Delta y_j \sim \operatorname{sgn}(y - y_j)$ in which only y_j of the winner $\mathbf{w}_j$ is updated. We provide the following theorem:

Theorem 2: If $\tilde{y}_j$ is the median of the partition $\mathbb{Y}_j$ corresponding to the image of region $\mathbb{D}_j$ of winner $\mathbf{w}_j$ then $P(y_j = \tilde{y}_j) = 1$ at equilibrium.

Proof: Let y_j correspond to $\mathbf{w}_j$ and assume the image of $\mathbb{D}_j \subset \mathbb{R}^{2d}$ to subspace $\mathbb{Y}_j \subset \mathbb{N}$ via the SCP $y = f(\mathbf{q})$. The median $\tilde{y}_j$ of $\mathbb{Y}_j$ satisfies $P(y \geq \tilde{y}_j) = P(y \leq \tilde{y}_j) = \frac{1}{2}$. Suppose that y_j has reached equilibrium, i.e., $\Delta y_j = 0$, which holds with probability 1. By taking the expectations of both sides and replacing Δy_j with the update rule sgn$(y - y_j)$:

$$\begin{aligned} E[\Delta y_j] &= \int_{\mathbb{Y}_j} \operatorname{sgn}(y - y_j)p(y)\mathrm{d}y = P(y \geq y_j)\int_{\mathbb{Y}_j} p(y)\mathrm{d}y - \\ & P(y < y_j)\int_{\mathbb{Y}_j} p(y)\mathrm{d}y = 2P(y \geq y_j) - 1. \end{aligned}$$

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

The learning phase of $\mathcal{M}$ is described in Algorithm 1. The input is the training set of pairs $\mathcal{Q} = \{(\mathbf{q}, y)\}$, 2-dim. lattices $\mathcal{L}$ and $\mathcal{C}$ with M entries, and a stopping threshold $\theta > 0$. The algorithm processes successive random pattern pairs until a termination criterion $T_t \leq \theta$. T_t is the 1-norm between successive estimates of neurons and cardinality prototypes $T_t = \sum_{i=1}^{M}(\|\mathbf{w}_i(t) - \mathbf{w}_i(t-1)\|_1 + |y_i(t) - y_i(t-1)|)$ with $\|\mathbf{w}_i\|_1 = \sum_{k=1}^{2d}|w_{ik}|$. The output is parameter set α.

ALGORITHM 1: The Cardinality Learning Algorithm.

```
Input: training set $\mathcal{Q}$, lattices $\mathcal{L}$,$\mathcal{C}$ with $M$ entries,
       stopping threshold $\theta$
Output: parameter set $\alpha$
Initialize $(\mathbf{w}_i(0), y_i(0)), i = 1, \ldots, M$, $t \leftarrow 0$;
repeat
    $t \leftarrow t + 1$;
    Select randomly a pair $(\mathbf{q}(t), y(t)) \in \mathcal{Q}$;
    $j = \arg\min_{\mathbf{w}_i \in \mathcal{L}} \|\mathbf{q}(t) - \mathbf{w}_i(t)\|_2$ /*project*/;
    Update $\mathbf{w}_i(t), \forall i$ /*quantization*/;
    Calculate $\hat{y}(t)$ /*prediction feedback*/;
    Update $y_i(t), \forall i$ /*adaptation*/;
until $T_t \leq \theta$;
```

Prediction Phase Once parameter set α is trained, and thus no more updates are realized on neurons and cardinality prototypes, we predict the cardinality $\hat{y}$ given a random query $\mathbf{q}$ as defined in (1). That is, we proceed with SCP without executing the incoming query $\mathbf{q}$. Firstly, $\mathbf{q}$ is projected onto $\mathcal{L}$ and its winner $\mathbf{w}_j$ is obtained. If y_j is the associated cardinality in lattice $\mathcal{C}$, the predicted `COUNT` value is $\hat{y}$ obtained from (1) under Kernel regression over a region around y_j in lattice $\mathcal{C}$.

Update Phase: Model $\mathcal{M}$ deals with Problems 2 and 3 by updating parameter α thus inducing changes on $\mathcal{L}$ and $\mathcal{C}$. The update rules in Theorem 1 yield $\mathcal{M}$ also capable of being adaptable to both query distribution and data changes. Consider that the query distribution progressively changes. The model treats such change by adapting **only** the winner neuron $\mathbf{w}_j$ and corresponding cardinality y_j upon the presence of a pair $(\mathbf{q}, y)$ using the rules in Theorem 1. The rationale behind such update is that $\mathbf{w}_j$ is moved toward $\mathbf{q}$ to capture the changes in query distribution and, simultaneously, y_j moves toward y. The rate of adaptation $\beta_j \in (0, 1)$ in the update phase depends on the number of times ζ_{j0} neuron $\mathbf{w}_j$ was a winner during the learning phase. In this phase, we define the update rate $\beta_j(t) = \frac{\zeta_{j0}}{\sum_{i=1}^{M}\zeta_{i0}}(1+t)^{-1}$ to prevent neuron and cardinality prototypes from moving too fast and therefore destabilizing the update process. Note, here each winner neuron has its own update rate β_j depending on its 'winning' history during the learning phase. Consider now that updates on set $\mathcal{B}$ occur. In this case, $\mathcal{M}$ updates **only** the associated cardinality prototype y_j and leaves the winner $\mathbf{w}_j$ untouched given a pair $(\mathbf{q}, y)$. That is because, the query distribution is not altered thus $\mathcal{M}$, which has already quantized the query space, proceeds with the update of the image of $\mathbf{w}_j$ only. In this context, y_j is moving toward y as given in (3) with β_j defined above in the query distribution change. Both types of updates require a change detection mechanism to identify a change in $p(\mathbf{q})$ and/or in $p(\mathbf{q}, y)$. In this paper we assume that this mechanism is provided thus proposing a framework that treats the updates. The definition of a change detection mechanism is beyond the scope of this paper and is in our future research agenda.

V. Performance Evaluation

The query and cardinality prototypes constitute a $2d+1$ dim. vector. The learning algorithm requires $O(dM)$ space. The computational cost for prototype updates is $O(dM)$. Since prototypes are updated during learning, the learning phase requires $O(d/\theta)$ [28] iterations to get $T_t \leq \theta$. After learning, we obtain SCP in $O(d \log M)$ by applying an one-nearest neighbor search for the winner using a $2d$-dim. tree structure over prototypes in $\mathcal{L}$. Adaptation given a pair requires also $O(d \log M)$ time for searching for the winner. We will show that by extracting significant knowledge from pairs $(\mathbf{q}, y)$ without relying on underlying data, we achieve better predictions and also adapt to query patterns/data changes.

We now turn to study the model's performance sensitivity over real and synthetic datasets on (i) SCP accuracy, (ii) capability of adapting to query patterns and/or data changes (iii) storage requirements, (iv) number of training pairs, and (iv) construction and prediction time. We also provide a comparative assessment of our query-driven model with data-centric approaches, despite their failure to address our desiderata in our setting. The approaches are: (i) GenHist histograms [9], (ii) the learning framework for STHs [20], (iii) ISOMER STH [19], (iv) STHoles [18], and (v) sampling [15]. The relative percentage SCP error $e = \frac{|y-\hat{y}|}{y}$ is used in [11], [9] and [20] and we adopt it here to enable direct comparisons.

Datasets & Workloads: The following real datasets are used since they were adopted by the competing approaches above, in order to enable objective direct comparison of our model against these works. The real dataset RS2 refers to the forest Cover dataset from UCI Machine Learning Repository (MLR)[1]. We use RS2 for comparison with STHoles ($d = 4$) and GenHist ($d \in \{4, 5, 8, 10\}$). All points in RS2 are normalized as in the GenHist paper [9]. The real dataset RS3 refers to the Census dataset [29] from UCI MLR consisting of 3-dim. points scaled in [0,1] and used for comparison with [20] and STHoles and ISOMER. RS1 is also a real dataset taken from the UCI MLR[2] containing real-valued multivariate points with $d \in \{2, 4\}$. We use RS1 for comparison against the sampling method in [15] and also for our sensitivity analysis. Moreover, we use the SD1 synthetic dataset for our sensitivity analysis, which contains $5 \cdot 10^9$ 3-dim. real points drawn from a mixture of 20 Gaussians with random mean/variance in each dimension. The ranges of the three attributes are 10^2, 10^3, 10^7, respectively, obtaining points with variety in domain ranges. We generate the training set $\mathcal{Q}$ and evaluation set $\mathcal{E}$. Set $\mathcal{E}$ is generated independently of $\mathcal{Q}$, thus assuring completely unseen queries. The $\mathcal{Q}$ size is a very small fraction $\gamma \in [1‰, 1\%]$ of the size of set $|\mathcal{B}|$. Based on $|\mathcal{Q}|$ we define the number of neurons for lattice $\mathcal{L}$ (and $\mathcal{C}$). This is the minimum information required by our model to be initialized, thus, $M = |\mathcal{Q}|$. Also, $|\mathcal{E}|$ is a factor $\delta \in [1, 100]$ of $|\mathcal{Q}|$. $\mathcal{Q}$ contains a number of K query subspaces $\mathbb{Q}_k \subset \mathbb{R}^{2d}$, where each $\mathbb{Q}_k, k = 1, \ldots, K$, is characterized by a query pattern-generator $(\mathbf{c}_k, \mathbf{v}_k, \ell_k)$. The center of each $\mathbf{q}$ of $\mathbb{Q}_k$, for each dimension i, is sampled from a Gaussian distribution $\mathcal{N}(c_{ki}, v_{ki})$ with mean c_{ki}, variance v_{ki}, and radius ℓ_{ki}. That is the lower bound $a_{ki} = x_{ki} - \ell_{ki}$ and upper bound $b_{ki} = x_{ki} + \ell_{ki}$, where center $x_{ki} \sim \mathcal{N}(c_{ki}, v_{ki})$. The volume of each query $2\ell_{ki}$ is drawn uniformly at random from 1% to 25% of the range of the i-th dimension. A random query $\mathbf{q}$ is generated as follows: a $\mathbb{Q}_k$ is selected uniformly at random from K query spaces with equal probability $\frac{1}{K}$.

[1]kdd.ics.uci.edu/summary.data.type.html

[2]http://archive.ics.uci.edu/ml/machine-learning-databases/00235/

TABLE I. PARAMETERS

Parameter	(Default) Value/Range
Data dimension d	$\{2,3,4,5,8,10\}$
Dataset size $\lvert\mathcal{B}\rvert$	2,075,259 (RS1), 545,424 (RS2), 210,138 (RS3) $5 \cdot 10^9$ (SD1)
No. of neurons M	$\gamma\lvert\mathcal{B}\rvert$, $\gamma = \{1‰, 2‰, 5‰, 1\%\}$
Training set size $\lvert\mathcal{Q}\rvert$	M
Evaluation set size $\lvert\mathcal{E}\rvert$	$\delta\lvert\mathcal{Q}\rvert$, $\delta = 1, \ldots, 100$
Query subspaces K	$\{10,100,1000\}$
Convergence threshold θ	10^{-3}

Then, from $\mathbb{Q}_k$ we obtain the lower, upper and volume values for all attributes. Since we wish to study changes in $p(\mathbf{q})$, we alter the mean value and variance of all query spaces in each workload. $\mathcal{M}$ is trained with $\mathcal{Q}$ and at some time during evaluation, we alter all parameters of $\mathbb{Q}_k, \forall k$, thus queries are drawn now from different distributions. We also generate query workloads with the exact same way described in STHoles paper itself [18]. All queries are hyper-rectangles included in a hypercube of volume 1% of the data domain. There are two types of workloads: (i) *Gauss* workload (GWL) in which the query center follows a multi-gaussian distribution independent of the data distribution and has an average volume of 1% of the data domain. (ii) *Uniform* workload (UWL) in which the query center follows a uniform distribution in the data domain and has an average volume of 1% of the data domain. Moreover, the workload WL1 for ISOMER, EquiHist and SpHist [20] is generated with the exact same way as generated in [20] where center $c_i = \frac{a_i+b_i}{2}$ of each dimension $i = 1, \ldots, d$ is selected uniformly at random from [0,1]. A $\mathbf{q}$ is d-dim. hyper-rectangle centered at c_i and with volume $b_i - a_i$ at most 20% of [0,1]. For GenHist, we create two workloads WL2 and WL3 in exactly the same way as generated in [9]. WL2 contains queries whose centers are chosen uniformly at random in the data domain $[0,1]^d$. WL3 contains queries with zero lower bound $a_i = 0, i = 1, \ldots, d$ and upper bound a randomly chosen point $[b_1, \ldots, b_d]^\top \in [0,1]^d$. Table I shows the parameters.

Accuracy of Prediction & Time: Figure 2 shows the percentage SCP error e for our model against number of neurons $M = |\mathcal{Q}| = \gamma|\mathcal{B}|$ (corresponding to $M \in \{2075, 4150, 10376, 20752\}$) and factor δ ($|\mathcal{E}| = \delta|\mathcal{Q}|$) for different number of query subspace K over RS1 with $d = (2,4)$. Our model achieves very low error as M increases, however, an increase in M ($M > 5000$) does not significantly improve the SCP accuracy, thus, there is no need to store a higher number of neurons. A fraction of $\gamma = 2‰$ of $|\mathcal{B}|$ results to SCP error lower than 5% for $d = 2,3,4$. Furthermore, our model depends on the number of query subspaces. For $K = 1000$, the error increases indicating that $\mathcal{M}$ should increase the lattice resolution (increment of M) to capture all variety of query spaces. Nonetheless, for $K \geq 100, \gamma = 2‰$, $\mathcal{M}$ exhibits a robust behavior in terms of error. The error remains constant with an increase in the evaluation set size $|\mathcal{E}|$ for all γ in Figure 2 (right). This is expected since $\mathcal{M}$ has captured the statistical characteristics of query patterns, thus being able to deal with numerous unseen queries. Figure 3 (left) shows the impact of M on error when the range of attributes A_1, A_2 and A_3 in the 3-dim. SD1 varies significantly; $\delta = 100$. In Figure 3(right) we observe that $\mathcal{M}$ obtains constant error against correlation coefficient $\rho \in \{0, 0.3, 0.6, 0.9\}$ among the attributes in SD1. Here, ρ quantifies the Pearson correlation coefficient between A_1 and A_2, and between A_1 and A_3. $\mathcal{M}$ does not depend on the attributes ranges and any correlation among attributes, since it relies only on the query patterns.

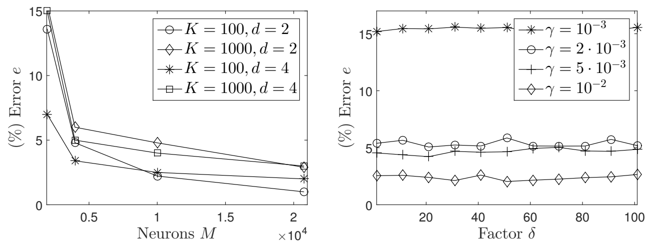


Fig. 2. Error vs. (left) neurons ($d = 2, 4$) and (right) factor δ ($d = 4$); RS1.

We measure the learning time (in seconds) in Figure 4(left) of $\mathcal{M}$ for different number of neurons and query subspaces over RS1 ($d = (2,4)$) on a PC Intel Core i5 CPU at 3.40GHz, 16 GB RAM. Our model stops learning when $T_t > \theta$, which converges quicker and, especially, for $M \leq 5000$, it takes up to five seconds to converge; $M < 5000$ is adequate for obtaining very low error compared to other models (as it will be shown later). The construction time depends on the number query subspaces K (variability of patterns). When K increases, our model must learn a 'richer' query space, thus more training pairs are needed to converge. Moreover, the construction time depends on dimension d ('curse of dimensionality'). The higher the dimension of a neuron (which is $2d$) the more computations are required for updates. The SCP prediction time in milliseconds for a random query $\mathbf{q}$ ranges from 0.06ms to 0.159ms when $10^3 \leq M \leq 2 \cdot 10^4$.

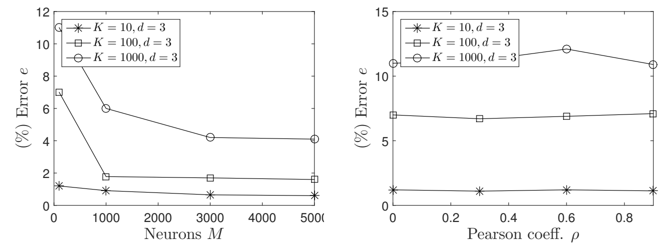


Fig. 3. Error vs. (left) neurons, (right) vs. Pearson's correlation coeff.; SD1.

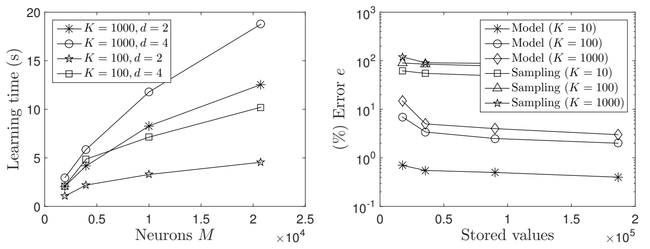


Fig. 4. (left) Learning time vs. neurons over RS1; (right) Error vs. stored values for our model and sampling over RS1; $d = 4$.

Adaptation: Consider the following experimental scenario in Figure 5 (left) repeated 100 times: $\mathcal{M}$ is trained with $K = 100$ query subspaces over SD1. During prediction, $\mathcal{M}$ receives queries one at a time generated by a workload where we randomly change the query pattern-generators' values of all query subspaces thus being different with those used for training. After the first 50 queries, we alter the query subspaces and observe that $\mathcal{M}$ adapts to this change after the next 20 queries. At the 120th query we alter again the query subspaces and observe after the next 30 queries $\mathcal{M}$ has adjusted again to follow the new subspaces and so on. We experiment with dataset updates by randomly changing 25% of data in $\mathcal{B}$ progressively up to 100% total change. Figures 5 (right) shows the error vs. a stream of queries. We alter, consequently, the 25% of $\mathcal{B}$ at the 50th, 120th, 180th and 230th query. $\mathcal{M}$ adapts after (on average) 30 queries.

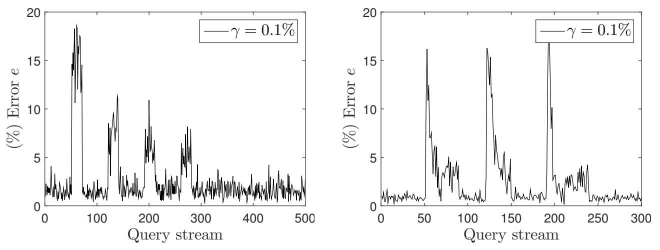


Fig. 5. Error vs. query streams (left) changing $K = 100$ query subspaces, (right) progressively update 25% of the dataset up to 100%; $K = 100$, SD1.

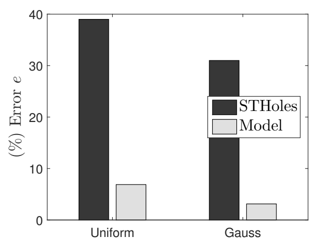
Models Comparison: We provide a comparative assessment against several data-centric approaches. Despite their shortfalls explained earlier, we again aim to demonstrate our model's advantages. Sampling techniques, e.g., [14], [15], sample points from $\mathcal{B}$ randomly and uniformly without replacement, thus obtaining the sample $\mathcal{B}'$. The cardinality values for $\mathcal{B}'$ are used as estimates of SCP over $\mathcal{B}$. We obtain $\mathcal{B}'$ by adopting the *reservoir sampling* algorithm [15]. STHoles [18] uses QFRs to build histograms and requires, for each query and histogram bucket, the computation of the number of rows in the intersection of the query and bucket regions. These detailed row counts are usually not obtainable from the actual queries, thus, STHoles must insert artificial predicates to specify the bucket boundaries. ISOMER [19] is a feedback-driven STH, using the information-theoretic principle of maximum entropy to approximate the $p(\mathbf{x})$. The obtained histogram might have $\Theta(n)$ buckets, given n QFRs, thus getting a histogram with $n' \ll n$, ISOMER heuristically has to eliminate up to $(n - n')$ QFRs; this discards valuable QFR information with impact on SCP accuracy. The framework for STHs in [20] uses QFRs as training pairs by introducing (i) the EquiHist algorithm, which learns a fixed size-bucket Equi-width histogram and (ii) the SpHist algorithm, which adopts Haar wavelets and compressed sensing for treating the histogram learning problem as a sparse-vector recovery problem. GenHist in [9] estimates the $p(\mathbf{x})$ by allowing the buckets to overlap and assuming that within each bucket, $p(\mathbf{x})$ is approximated by the average data density of the bucket. We chose GenHist [9] as a highly regarded histogram method for real-valued data, significantly outperforming the wavelets-based [11], MHIST-2 [12], m-d Kernels and attribute value independence-based methods.

The SCP error of the sampling method shown in Figure 4(right) is high compared with our method over 4-dim. data from RS1 for different query subspaces K given exactly the same number of stored points (we obtain similar results for $d = 2$ not shown for space limitations) with $M = 1\%|\mathcal{B}|, |\mathcal{E}| = 100|\mathcal{Q}|$. The sample size is $|\mathcal{B}'| = M(2 + \frac{1}{d})$ d-dim. points since our model stores M neurons of $2d + 1$ dimension ($2d$ is for query prototype plus the cardinality prototype). The sampling method stores points of d dimensions. Note that sampling-based SCP apart from being inappropriate in data access restricted scenarios, cannot adapt to updates in query patterns / data. Also, the average SCP time per query for sampling is 33.6s with RS1 and 445s with SD1 compared to 0.109ms with RS1 and 0.14ms with SD1 in our model. Note, our model does not depend on the size of set $\mathcal{B}$, which is our fundamental characteristic.

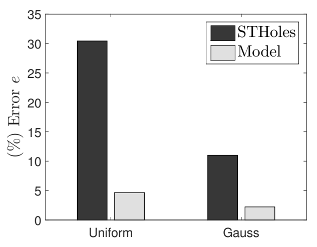
We compare our model with STHoles using the same RS2 and RS3 as used in [18] over both workloads GWL and UWL constructed with the same way as in [18]. Figure 6(a) shows the SCP error for our model and STHoles over RS2 ($d = 4$) using UWL and GWL having the same memory capacity of 1 kilobyte. (A d-dim. STHoles histogram with B buckets requires $2dB$ values for bucket boundaries, B values for frequencies, and $2B$ tree-structure pointers [18]; recall our model requires $M(2d + 1)$ values.) The training workload for both models consists of 1000 pairs and used 1000 different evaluation pairs. For both workloads, our model achieves significantly lower error than STHoles with less error obtained over GWL. That is because, queries in GWL form *clusters* of queries, through which $\mathcal{M}$ can identify such clusters via lattice $\mathcal{L}$. With UWL, queries are uniformly distributed thus resulting to a perfect arrangement of neurons on lattice $\mathcal{L}$ with homogeneously spread neurons. Similar comparison results are obtained using RS3 with GWL and UWL; Figure 6(b).

We now compare our model with EquiHist, SpHist and ISOMER using the same RS3 as used in [20]. We generate a training set $\mathcal{Q} = \gamma|\mathcal{B}|$, $\gamma \in [2‰, 6‰]$, and a different evaluation set $\mathcal{E}$ to compute the SCP error based on WL1, with $|\mathcal{E}| = 2.5\%|\mathcal{B}|$. We compare the performance of all models with SpHist, EquiHist and ISOMER using $B = 64$ buckets. Each of the buckets, in each dimension consists of two boundary values and one value for the frequency. Thus each corresponding histogram stores $B(2d + 1) = 448$ values. Figure 6(c) shows the impact of $|\mathcal{Q}|$ on error for SpHist, EquiHist, ISOMER, and our model (with $M = 64$ neurons thus exactly the same number of stored values). Our model achieves by far the lowest error than the other approaches for all $|\mathcal{Q}|$. Because, as SpHist, EquiHist, and ISOMER attempt to tune a histogram with more QFRs thus estimate $p(\mathbf{x}|y, \mathbf{q})$, our model extracts and learns significant information of WL1 by implicitly estimating $p(y|\mathbf{q})$. Our model can also obtain the same error with fewer pattern pairs thus there is no need for larger numbers of neurons. Figure 6(d) shows the SCP error in RS3 against stored values for our model (corresponding to $M \in [20, 257]$), SpHist, EquiHist, and ISOMER after training them with $|\mathcal{Q}| = 1‰|\mathcal{B}|$. Our model obtains low error with very low storage, while the more information is stored by SpHist, and ISOMER the lower the error they achieve but up to a certain number of stored values.

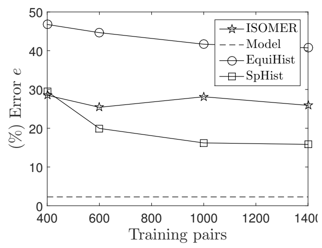
Our model is compared against GenHist over RS2 $d = (5, 10)$ (as used in [9]) and workloads WL2 and WL3. Figure 7 shows the SCP error against stored values using WL2 and WL3 (similar results are obtained over 4- and 8-dim. points but are not shown due to space limitations). We vary the



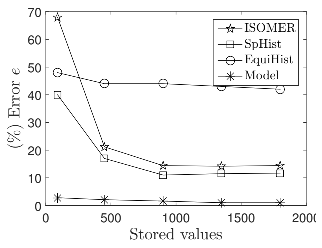
(a) 4-dim. RS2



(b) 3-dim. RS3



(c) 3-dim. RS3



(d) 3-dim. RS3

Fig. 6. Error of STHoles and our model for GWL/UWL over (a) RS2, $d = 4$, (b) RS3, $d = 3$. Error of ISOMER, EquiHist, SpHist, and our model for WL1 over RS3 ($d = 3$) against (c) number of training pairs and (d) stored values.

training set size such that $|\mathcal{Q}| = \gamma|\mathcal{B}|$ with $\gamma \in [2‰, 1\%]$ and $\gamma \in [1‰, 1.5\%]$, for $d = 5$ and $d = 10$, respectively. The evaluation set size $|\mathcal{E}| = \delta|\mathcal{Q}|$ with $\delta \in [5, 50]$ and $\delta \in [65, 1100]$, for $d = (5, 10)$, respectively. In our model, this corresponds to $M = [100, 484]$ and $M = [64, 400]$, respectively. In GenHist, the stored values $B(2d + 1)$ refer to the information stored for B buckets, each one represented by the triple: lower and upper value of the bucket and number of data points in bucket for each dimension. Our model outperforms GenHist by a wide margin for WL2 and for WL3. GenHist is unaware of the way query patterns are generated. This reflects the basic difference of our approach compared to histograms for SCP. Note also that both workloads refer to uniformly at random queries and our model outperforms GenHist in terms of SCP accuracy given the same memory size. An increase in M does not significantly contribute to better SCP accuracy, thus we could use fewer neurons to learn WL2 and WL3.

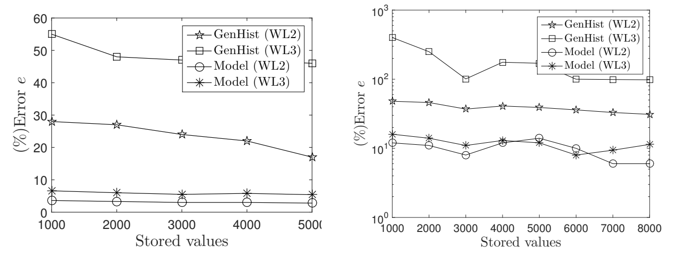


Fig. 7. Error vs. stored values of our model and GenHist (left) $d = 5$, (right) $d = 10$; RS2 using WL2 & WL3.

VI. Conclusions

We introduce a novel perspective and solution for the problem of Set Cardinality Prediction. The fundamental unique characteristic of our approach is that it is query-driven. This is especially important for big data settings, as an increase in the underlying dataset size is largely inconsequential for our method's efficiency and accuracy. The contributed ML model extracts knowledge and learns from previous queries and their results and predicts the cardinality of the answer set based on similar previous queries. The model quantizes the query space and forms a regression plane through learning the $\mathbf{q} \rightarrow y$ association. Our model also efficiently adapts to query-pattern and data updates. Our comprehensive experiments showcased the model's robustness and that our model achieves very small error rates with small memory footprints outperforming the data-centric state-of-the-art. In addition to these advantages, the proposed model represents the only solution applicable to general modern big data environments, which may include data nodes/owners placing access restrictions (e.g., for sensitive data) and/or where data accesses may be too costly to execute (e.g., in a cloud setting).

References

[1] M. Zaharia et al, 'Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing', NSDI, April 2012.

[2] B.T. Ong et al, 'Dynamic pre-training of Deep Recurrent Neural Networks for predicting environmental monitoring data', IEEE Intl. Conf. Big Data, 760–765, 2014.

[3] L. Chieh-Yen Lin et al. 'Large-scale logistic regression and linear support vector machines using Spark', IEEE Intl. Conf. Big Data, 519–528, 2014.

[4] N. Balac et al. 'Large Scale predictive analytics for real-time energy management', IEEE Intl. Conf. Big Data, 657–664, 2013.

[5] A. Atanasov et al.'Query-driven parallel exploration of large datasets', IEEE LDAV'12, pp.23–30.

[6] G. Cormode et al. 'Synposes for Massive Data: Samples, Histograms, Wavelets and Sketches'. now publishers, 2012.

[7] L.J. Gosink et al. 'An Application of Multivariate Statistical Analysis for Query-Driven Visualization', IEEE Trans. Vis. Comp. Graph., 7(3):264–275, 2011.

[8] A. Chaudhuri et al. 2014. 'Efficient Range Distribution Query for Visualizing Scientific Data', IEEE PacificVis, pp.201–208.

[9] D. Gunopulos et al. 'Selectivity estimators for multidimensional range queries over real attributes'. VLDB J. 14(2):137–154, 2005.

[10] A. Aboulnaga et al. 1999. 'Self-tuning histograms: building histograms without looking at data'. ACM SIGMOD '99, pp.181–192.

[11] J.S.Vitter et al. 1998. 'Data cube approximation and histograms via wavelets'. ACM CIKM '98, pp.96–104.

[12] V. Poosala et al. 1997. 'Selectivity estimation without the attribute value independence assumption', 23rd VLDB'97, pp.486–495.

[13] J. Lee et al. 1999. 'Multi-dimensional selectivity estimation using compressed histogram information'. ACM SIGMOD '99, pp.205–214.

[14] F. Olken et al. 1990. 'Random sampling from database files: a survey'. 5th SSDBM '90, pp.92–111.

[15] P.J. Haas et al. 1992. 'Sequential sampling procedures for query size estimation'. ACM SIGMOD'92, 341–350.

[16] H. To et al. 2013. 'Entropy-based histograms for selectivity estimation'. ACM CIKM '13, pp. 1939–1948.

[17] Y. Ioannidis. 2003. 'The history of histograms (abridged)'. 29th VLDB '03, pp.19–30.

[18] N. Bruno et al. 2001. 'STHoles: A multidimensional workload-aware histogram'. ACM SIGMOD'01,211–222.

[19] U. Srivastava et al. 2006. 'Isomer: Consistent histogram construction using query feedback'. IEEE ICDE '06, p.39.

[20] R. Viswanathan et al. 2011. 'A Learning Framework for Self-Tuning Histograms', CoRR abs/1111.7295.

[21] B. Kosko. 1991. 'Stochastic competitive learning'. IEEE Trans. Neur. Netw. 2(5):522–529, Sept. 1991.

[22] T. Kohonen. 2001. 'Self-Organizing Maps' (3rd ed.) Springer, USA.

[23] H. Robbins et al. 1951. 'A Stochastic Approximation Method', Annals of Mathematical Statistics, 22(3):400–407, Sept. 1951.

[24] A. Ahmad et al. 2007 'A k-mean clustering algorithm for mixed numeric and categorical data' Data Knowl Eng, 63(2):503–527.

[25] Y. Tao et al. 'The Power-Law Method: A Comprehensive Estimation Technique for Multi-Dimensional Queries'. CIKM'03, 83–90.

[26] E. Ikonomovska et al. 'Learning model trees from evolving data streams', Data Mining and Knowledge Discovery, pp. 1–41, 2010

[27] J Ma et al. 'Accurate On-line Support Vector Regression', Neural Comput. Nov'03, 15(11):2683–703

[28] O. Bousquet et al. 2007. 'The Tradeoffs of Large Scale Learning', NIPS 2007, pp. 161–168.

[29] C. Blake et al. 1998. UCI repository of ML databases.