Mergeable Summaries

Pankaj K. Agarwal Graham Cormode Zengfeng Haung Jeff M. Phillips
Zhewei Wei Ke Yi

Abstract

We study the *mergeability* of data summaries. Informally speaking, mergeability requires that, given two summaries on two data sets, there is a way to merge the two summaries into a summary on the two data sets combined together, while preserving the error and size guarantees. This property means that the summary can be treated like other algebraic objects such as sum and max, which is especially useful for computing summaries on massive distributed data. Many data summaries are trivially mergeable by construction, most notably those based on linear transformations. But some other fundamental ones like those for heavy hitters and quantiles, are not (known to be) mergeable. In this paper, we demonstrate that these summaries are indeed mergeable or can be made mergeable after appropriate modifications. Specifically, we show that for ε-approximate heavy hitters, there is a deterministic mergeable summary of size $O(1/\varepsilon)$; for ε-approximate quantiles, there is a deterministic summary of size $O(\frac{1}{\varepsilon}\log(\varepsilon n))$ that has a restricted form of mergeability, and a randomized one of size $O(\frac{1}{\varepsilon}\log^{3/2}\frac{1}{\varepsilon})$ with full mergeability. We also extend our results to geometric summaries such as ε-approximations and ε-kernels.

1 Introduction

Data summarization is an important tool for dealing with massive data. There are many reasons to desire a summary in place of the full data: compact summaries enable accurate query answering while requiring much lower resources, computations need less memory, and can be much faster. In some situations, it is not feasible to work with the full data, such as when it is distributed across many different locations, or is observed as a stream of data. Broadly speaking, a *summary* S on a data set (a bag of items) D is any compact data structure from which certain queries on D can be answered. Since S should be much smaller than D, queries are usually answered approximately, and there is a trade-off between the size of S and the approximation error ε. A variety of data summaries have been studied in the past, starting with statistical summaries like heavy hitters, quantile summaries, histograms, various sketches and synopses, to geometric summaries like ε-approximations and ε-kernels, and graph summaries like distance oracles.

There are several models for how summaries can be built from the base data. At the most basic level, we have the data set D accessible in its entirety, and the summary S is constructed offline: this can be thought of as akin to a "bulk-load" operation. More generally, we often want the summary to be maintained in the presence of updates, i.e., when a new record is added to D, S can be updated to reflect the new arrival without recourse to the underlying D. Much progress has been made on incrementally maintainable summaries in the past years, mostly driven by the study of data stream algorithms, as a streaming algorithm with small space implies an incrementally maintainable summary.

In this paper, we study stronger requirements on summaries where there is a process of repeatedly *merging* together two summaries of (separate) data sets to obtain a summary of their union. Such merging is a key requirement in distributed and parallel computations, as we describe subsequently. We distinguish two variants, *full mergeability* and *one-way mergeability*. In the following, we use $S(D,\varepsilon)$ to denote a summary on a data set D with approximation error ε. The exact interpretation of ε will vary depending on the purpose of the summary; for now, we just treat it as a parameter.

Definition 1.1 (Full mergeability)**.** *A summary $S(D,\varepsilon)$ is* fully mergeable *if the size $|S(D,\varepsilon)| \le k(1/\varepsilon, |D|)$ is bounded by an absolute function $k()$, and there exists an algorithm $\mathcal{A}$ that produces $S(D_1 \uplus D_2, \varepsilon)$[1] from any two input summaries $S(D_1,\varepsilon)$ and $S(D_2,\varepsilon)$.*

[1] $\uplus$ denotes multiset addition.

Clearly, a trivial fully mergeable summary is one where $\mathcal{A}$ simply puts $S(D_1, \varepsilon)$ and $S(D_2, \varepsilon)$ together, doubling its size. But such a summary will have size linear in $|D|$ in a repeatedly merging process. Thus we will only be interested in summaries whose size bound $|S(D, \varepsilon)| \leq k(1/\varepsilon, |D|)$ is always sublinear in (and ideally, independent of) $|D|$ at any time in the possibly indefinite merging process. *That is, a fully mergeable summary can be merged, in an arbitrary fashion for an indefinite number of steps, without propagating either the error or the size.* Such a mergeable summary essentially allows us to use it just like some simple aggregate like sum or max.

Definition 1.2 (One-way mergeability)**.** *A summary $S(D, \varepsilon)$ is* one-way mergeable *of size $|S(D, \varepsilon)| \leq k(1/\varepsilon, |D|)$ if there exists two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ such that, (1) $\mathcal{A}_1$ takes any dataset D and produces $S(D, \varepsilon)$; (2) $\mathcal{A}_2$ takes two summaries, $S(D_1, \varepsilon)$ created by $\mathcal{A}_1$ and $S(D_2, \varepsilon)$ created by $\mathcal{A}_2$ or $\mathcal{A}_1$, and produces $S(D_1 \uplus D_2, \varepsilon)$.*

Note that one-way mergeability degenerates to the standard streaming model if $\mathcal{A}_1$ simply takes a single item x and does nothing to it. One-way mergeability is essentially a "batched streaming" model where there is a main algorithm $\mathcal{A}_2$ that maintains the summary, which can take in a batch of items summarized by another algorithm $\mathcal{A}_1$. Generality increases from the streaming model to one-way mergeability, and to full mergeability.

1.1 Applications

Mergeable summaries are needed in a variety of settings. We mention two concrete applications here.

Data aggregation in sensor networks. In a sensor network, the nodes organize themselves into a routing tree rooted at the base station. Each sensor holds some data and the goal of data aggregation is to compute a summary on all the data. Nearly all data aggregation algorithms follow a bottom-up approach [20]: Starting from the leaves, the aggregation propagates upwards to the root. When a node receives the summaries from its children, it merges these with its own summary, and forwards the result to its parent. Depending on the physical distribution of the sensors, the routing tree can take arbitrary shapes.

Computing the heavy hitters (a.k.a. frequent items) in a sensor network has received much attention. Previous solutions have used the MG summary but with a merging algorithm that does not preserve the error. As a result, they must set smaller ε near the leaves of the tree and gradually increase it towards the root. The appropriate ε values at different nodes need an additional round of communication on the sensor network to compute. A heuristic was given in [22] that tries to minimize the largest summary a node sends out, but with no provable bounds. Another solution [21] bounds the sum of sizes of all summaries communicated by $O(k/\varepsilon)$ (k is the number of nodes) for a class of "nice" routing trees. On bad trees, this cost can be as large as $O(k^{5/4}/\varepsilon)$.

For computing quantiles in sensor networks, the q-digest [30] is mergeable but it depends on $\log u$, which can be large. The quantile summary of [18] has size $O(\frac{1}{\varepsilon} \log n \log(h/\varepsilon))$ where h is the height of the routing tree and it needs the knowledge of h in advance.

Cluster computation. A second direct application of mergeable summaries is in the MUD (Massive Unordered Distributed) model of computation [14], which models large-scale distributed systems like MapReduce and Hadoop. In this model, the input data is broken into an arbitrary number of pieces, each of which is potentially handled by a different machine. Each piece of data is first processed by a *local* function, which outputs a message. All the messages are then pairwise combined using an *aggregation* function in an arbitrary fashion, eventually producing an overall message. Finally, a post-processing step is applied.

This exactly corresponds to our notion of full mergeability, where each machine builds a summary of its share of the input, the aggregation function is the merging algorithm, and the post-processing step corresponds to posing queries on the summary. The main result of [14] is that any deterministic streaming algorithm that computes a symmetric function defined on all inputs can be simulated (in theory) by a MUD

algorithm, but it does not hold for indeterminate functions, i.e., functions that may have many correct outputs. All problems considered in this paper are indeterminate, due to the approximation allowed. Their result extends to certain approximation algorithms by treating the (deterministic) approximation algorithm as the function, which will always give one determinate output. Unfortunately, for our problems, all existing streaming approximation algorithms are not symmetric: the output depends on the order in which the items arrive in the stream. Thus, the simulation result of [14] does not apply to any of the problems we consider.

1.2 Problems and Previous Results

The focus of this paper is on several fundamental problems that have existing summaries, but are not known to be mergeable (or have unsatisfactory bounds). We use n to denote $|D|$, the size of the data set being summarized. For randomized algorithms, the bounds we state hold for achieving a constant success probability for answering *all* queries correctly using the summary; the success probability can always be boosted to $1 - \delta$ by building $O(\log \frac{1}{\delta})$ independent summaries. Our results all hold in a comparison model, where only comparisons are used on items in the data sets. In this model we assume each data item, as well as any integer no more than n, can be stored in one unit of storage. Some prior work has more strongly assumed that items in D are drawn from a bounded universe $[u] = \{0, \ldots, u-1\}$ (or from $[u]^d$ in the d-dimensional case) for some $u \geq n$, and any integer less than u takes one unit of storage. In some cases $\log u$ is large, for example when the items are strings or user-defined types, so we seek to avoid such factors. Note that any result in the comparison model also holds in the bounded-universe model, but not vice versa.

Frequency estimation and heavy hitters. For a multiset D, let $f(x)$ be the frequency of x in D. A frequency estimation summary $S(D, \varepsilon)$ can be used to estimate $f(x)$ for any x within an additive error of εn. A heavy hitters summary allows one to extract all frequent items approximately, i.e., for a user-specified ϕ, it returns all items x with $f(x) > \phi n$, no items with $f(x) < (\phi - \varepsilon)n$, while an item x with $(\phi - \varepsilon)n \leq f(x) \leq \phi n$ may or may not be returned.

In the bounded-universe model, the frequency estimation problem can be solved by the Count-Min sketch [12] of size $O(\frac{1}{\varepsilon} \log u)$, which is a linear sketch, and is thus fully mergeable. Since the Count-Min sketch only allows querying for specific frequencies, in order to report all the heavy hitters efficiently, we need a hierarchy of sketches and the space increases to $O(\frac{1}{\varepsilon} \log u \log(\frac{\log u}{\varepsilon}))$. This approach is randomized, but there is also a deterministic linear sketch for frequency estimation [15], but its size is $O((\frac{1}{\varepsilon} \log u)^2 \log n)$.

The counter based summaries, most notably the MG summary [27] and the SpaceSaving summary [26], have been reported [11] to give the best results for both the frequency estimation and the heavy hitters problem (over a data stream of arriving items). They are deterministic, simple, and have the optimal size $O(\frac{1}{\varepsilon})$. They also work in the comparison model. However, only recently were they shown to support one-way merging [7]. In applications which require full mergeability, prior work used a weaker result, that summaries could be merged with a loss in accuracy (i.e., resulting in a larger ε) at each step, and designed schemes to ration out this reduction in precision [21, 22].

Quantile summaries. For the quantile problem we assume that D is a set (i.e., no duplicates); this assumption can be removed by using any tie breaker. For any $0 < \phi < 1$, the *ϕ-quantile* of D is an item x with rank $r(x) = \lfloor \phi n \rfloor$ in D, where the *rank* of x is the number of items in D smaller than x. This is also known as the order statistic of D. An *ε-approximate ϕ-quantile* is an item with rank between $(\phi - \varepsilon)n$ and $(\phi + \varepsilon)n$, and a quantile summary allows us to extract an ε-approximate ϕ-quantile for any $0 < \phi < 1$. It is well known [11] that the frequency estimation problem can be reduced to an ε'-approximate quantile problem for some $\varepsilon' = \Theta(\varepsilon)$, by identifying items which are quantiles for multiples of ε'. Therefore a quantile summary is automatically a frequency estimation summary, but not vice versa.

Quite a number of quantile summaries have been designed [16, 18, 17, 30, 23, 12], but all the mergeable ones either have dependency on $\log u$ (thus work only in the bounded-universe case). The Count-Min sketch (more generally, any frequency estimation summary) can be organized into a hierarchy to solve the quantile

problem, yielding a linear sketch of size $O(\frac{1}{\varepsilon}\log^2 u \log(\frac{\log n}{\varepsilon}))$ [12]. The q-digest [30] has size $O(\frac{1}{\varepsilon}\log u)$; although not linear, it is still fully mergeable. Neither approach scales well when $\log u$ is large. The most popular quantile summary technique is the GK summary [17], which guarantees a size of $O(\frac{1}{\varepsilon}\log(\varepsilon n))$. A merging algorithm has been previously proposed, but the error increases, albeit slightly, with each merge [18]. Viewing the entire merging process as a binary tree, if its height, say h, can be bounded in advance, a desired error bound can still be guaranteed in the final summary at the root of the tree, with a summary size of $O(\frac{1}{\epsilon}\log n \log(h/\epsilon))$ [18]. This merging algorithm does not work in an indefinite merging process, so it is not mergeable by our definition.

Summaries for range counting. Next we consider summaries for approximate range counting, often referred to as *ε-approximations of range spaces*. In one dimension this is essentially equivalent to the quantiles problem: in both cases, we want to approximate the number of data points in D in any query interval within an additive εn error. In higher dimensions, queries are drawn from a family of ranges $\mathcal{A}$ (such as axis-aligned rectangles or halfspaces). An ε-approximation Q is a subset of D that guarantees that for all ranges $R \in \mathcal{A}$, the fraction $|R \cap Q|/|Q|$ is within ε of $|R \cap D|/|D|$, which means that $|D| \cdot |R \cap Q|/|Q|$ estimates the number of points in R within error $\varepsilon|D|$.

A random sample of $O(d/\varepsilon^2)$ points from P [33, 32] is an ε-approximation with constant probability where d is the VC-dimension of the range space. Random samples are easily mergeable, but they are far from optimal. It is known that for d-dimensional axis-aligned rectangles there are ε-approximations of size $O((1/\varepsilon)\log^{2d}(1/\varepsilon))$ [29], and for halfspaces the optimal size is $O(1/\varepsilon^{2d/(d+1)})$ [25]. These deterministic summaries are constructed by partitioning the data to small subsets, and then iteratively merging summaries of these subsets. However, for all existing analysis, the error increases on each merge step; hence these techniques are not known to be mergeable.

ε-kernels. Finally, we consider ε-kernels [2, 1] which are summaries for approximating the convex shape of a point set P. Specifically, they are a specific type of coreset that approximates the width of P within a relative $(1+\varepsilon)$-factor in any direction. These summaries have been extensively studied in computational geometry [8, 34, 9, 3] as they can be used to approximate many other geometric properties of a point set having to do with its convex shape, including diameter, minimum enclosing annulus, and minimum enclosing cylinder. In the static setting in $\mathbb{R}^d$, ε-kernels can always be constructed of size $O(1/\varepsilon^{(d-1)/2})$ in time $O(n + 1/\varepsilon^{d-3/2})$ [8, 34], and sometimes this size is required. In the streaming setting, several algorithms have been developed [2, 8, 4] ultimately yielding an algorithm [35] using $O((1/\varepsilon^{(d-1)/2})\log(1/\varepsilon))$ space.

However, ε-kernels in general, including those maintained by streaming algorithms, are not mergeable. Combining two ε-kernels will in general increase the error by ε in the resulting summary, or double the size. The streaming algorithms rely heavily on processing a single point at a time, so even one-way mergeable algorithms seem to be difficult.

1.3 Our Results

The ability to merge summaries together is a natural requirement, and as such, several types of summaries are known to be mergeable. For example, all *sketches* that are linear functions of D are clearly mergeable (due to their linearity, they usually support arbitrary linear operations such as subtractions as well). These include the AMS sketch [5], the Count-Min sketch [12], the ℓ_1 sketch [13, 28], among many others. Summaries that maintain the maximum or top-k values are also mergeable, most notably summaries for estimating the number of distinct elements [6, 19]. In the rest of the paper we provide several mergeable summaries for more complicated types of summaries.

- In Section 2 we show that the MG and SpaceSaving summaries are fully mergeable; we present a merging algorithm that preserves the size $O(1/\varepsilon)$ and the error parameter ε. Along the way we make the surprising observation that the two summaries are isomorphic, namely, an MG summary can be mapped to a SpaceSaving summary and vice versa, which may be of independent interest.

- In Section 3 we show that the GK summary for ε-approximate quantiles is one-way mergeable, although not fully mergeable. Then we design a randomized quantile summary of size $O(\frac{1}{\varepsilon}\log^{3/2}\frac{1}{\varepsilon})$ that is fully mergeable. This in fact even beats the previous best randomized streaming algorithm for quantiles, which had size $O(\frac{1}{\varepsilon}\log^{3}\frac{1}{\varepsilon})$ [31]. We conjecture that there are no deterministic quantile summaries of size $\text{poly}(\frac{1}{\varepsilon}, \log n)$ that is fully mergeable.
- In the full version we present fully mergeable ε-approximation of range spaces that come close to the size bounds for the deterministic static algorithms. This analysis generalizes the quantiles summaries (for intervals) to more general range spaces. Specifically, for axis-aligned rectangles our mergeable ε-approximation has size $O((1/\varepsilon)\log^{2d+3/2}(1/\varepsilon))$; for halfspaces (and other range spaces with VC-dimension d) the size is $O(1/\varepsilon^{2d/(d+1)} \cdot \log^{2d/(d+1)+1}(1/\varepsilon))$.
- In the full version we provide a fully mergeable ε-kernel for a restricted, but reasonable variant. We assume that we are given a constant factor approximation of the width in every direction ahead of time. This allows us to specify a fixed reference frame, and we can maintain a fully mergeable ε-kernel of size $O(1/\varepsilon^{(d-1)/2})$ with respect to this fixed reference frame. We leave the unrestricted case as an open question.

In the context of data aggregation in sensor networks our results imply several new or improved bounds, and are always oblivious to the size and height of the routing tree. For heavy-hitters our mergeable summaries will have maximum size $O(1/\varepsilon)$ and total size $O(k/\varepsilon)$ simultaneously. It is also simpler than the previous algorithms [22, 21]. For ε-approximate quantiles we improve the size to $O(\frac{1}{\varepsilon}\log^{3/2}\frac{1}{\varepsilon})$, and we believe our results for range counting summaries and ε-kernels are the first of any kind.

2 Heavy Hitters

The MG summary [27] and the SpaceSaving summary [26] are two popular counter-based summaries for the frequency estimation and the heavy hitters problem. We first recall how they work on a stream of items. For a parameter k, a MG summary maintains up to k items with their associated counters. There are three cases when processing an item x in the stream: (1) If x is already maintained in the summary, its counter is increased by 1. (2) If x is not maintained and the summary currently maintains fewer than k items, we add x into the summary with its counter set to 1. (3) If the summary maintains k items and x is not one of them, we decrement all counters by 1 and remove all items with counters being 0. The SpaceSaving summary is the same as the MG summary except for case (3). In SpaceSaving, if the summary is full and the new item x is not currently maintained, we find any item y with the minimum counter, replace y with x, and increase the counter by 1. Previous analysis shows that the MG and the SpaceSaving summaries estimate the frequency of any item x with error at most $n/(k+1)$ and n/k, respectively, where n is the number of items processed. Thus they solve the frequency estimation problem with additive error εn with space $O(1/\varepsilon)$, which is optimal. They can also be used to report the heavy hitters in $O(1/\varepsilon)$ time by going through all counters; any item not maintained cannot have frequency higher than εn.

In this section we show that both MG and SpaceSaving summaries are indeed fully mergeable. We first prove the mergeability of MG summaries by presenting a merging algorithm that preserves the size and error. Then we show that SpaceSaving and MG summaries are fundamentally the same, which immediately leads to the mergeability of the SpaceSaving summary.

We start our proof by observing that the MG summary provides a stronger error bound. Let $f(x)$ be the true frequency of item x and let $\hat{f}(x)$ be the counter of x in MG (set $\hat{f}(x) = 0$ if x is not maintained). For space, we defer proofs in this to the full version

Lemma 2.1. *For any item x, $\hat{f}(x) \le f(x) \le \hat{f}(x) + (n - \hat{n})/(k+1)$, where $\hat{n}$ is the sum of all counters in MG.*

This is related to the result that the MG error is at most $F_1^{res(k)}/k$, where $F_1^{res(k)}$ is the sum of the counts of all items except the k largest [7]. Since each counter stored by the algorithm corresponds to (a subset of)

actual arrivals of the corresponding item, we have that $\hat{n} \leq (n - F_1^{res(k)})$.

Now we present an algorithm that, given two MG summaries with the property stated in Lemma 2.1, produces a merged summary with the same property. More precisely, let S_1 and S_2 be two MG summaries on data sets of sizes n_1 and n_2, respectively. Let $\hat{n}_1$ (resp. $\hat{n}_2$) be the sum of all counters in S_1 (resp. S_2). We know that S_1 (resp. S_2) has error at most $(n_1 - \hat{n}_1)/(k+1)$ (resp. $(n_2 - \hat{n}_2)/(k+1)$). Our merging algorithm is very simple. We first combine the two summaries by adding up the corresponding counters. This could result in up to $2k$ counters. We then perform a prune operation: Take the $(k+1)$-th largest counter, say C_{k+1}, and subtract it from all counters, and then remove all non-positive counters.

Theorem 2.1. *Using the above algorithm, MG summaries are fully mergeable of fixed size $O(1/\varepsilon)$ with at most εn error. Each merge takes $O(1/\varepsilon)$ time.*

Next we show that MG and SpaceSaving are isomorphic. Specifically, consider an MG summary with k counters and a SpaceSaving summary of $k+1$ counters, processing the same stream. Let min^{SS} be the minimum counter of the SpaceSaving summary (set $min^{SS} = 0$ when the summary is not full), and $\hat{n}^{MG}$ be the sum of all counters in the MG summary. Let $\hat{f}^{MG}(x)$ (resp. $\hat{f}^{SS}(x)$) be the counter of item x in the MG (resp. SpaceSaving) summary, and set $\hat{f}^{MG}(x) = 0$ (resp. $\hat{f}^{SS}(x) = min^{SS}$) if x is not maintained.

Lemma 2.2. *After processing n items, $\hat{f}^{SS}(x) - \hat{f}^{MG}(x) = min^{SS} = (n - \hat{n}^{MG})/(k+1)$ for all x.*

Corollary 2.1. *The SpaceSaving summary is fully mergeable.*

3 Quantiles

We first show results for the one-way mergeability of quantiles by generalizing incremental algorithms, including the GK algorithm [17]. Then the bulk of our work is to show a randomized construction which achieves full mergeability by analyzing quantiles through the lens of ε-approximations of the range space of intervals. Let D be a set of n points in one dimension. Let $\mathcal{I}$ be the set of all half-closed intervals $I = (-\infty, x]$. Recall that an ε-approximation S of D (w.r.t. $\mathcal{I}$) is a subset of points of D such that for any $I \in \mathcal{I}$, $n|S \cap I|/|S|$ estimates $|D \cap I|$ with error at most εn. In some cases we may use a weighted version, i.e., each point p in S is associated with a weight $w(p)$. A point p with weight $w(p)$ represents $w(p)$ points in D, and we require that the weighted sum $\sum_{p \in S \cap I} w(p)$ estimates $|D \cap I|$ with error at most εn. Since $|D \cap I|$ is the rank of x in D, we can then do a binary search with x to find an ε-approximate ϕ-quantile for any given ϕ. We will first develop a randomized fully mergeable ε-approximation of size $O((1/\varepsilon)\log(\varepsilon n)\sqrt{\log(1/\varepsilon)})$ inspired by low-discrepancy halving. Then after we review some classical results about random sampling, we combine the random-sample-based and low-discrepancy-based algorithms to produce a hybrid mergeable ε-approximation whose size is independent of n.

3.1 One-way mergeability

Theorem 3.1. *Any quantile summary algorithm which is incrementally maintainable is one-way mergeable.*

For space, this theorem is proved in the full version. An immediate observation is that the GK algorithm [17] (along with other deterministic techniques for streaming computation of quantiles which require more space [23]) meets these requirements, and is therefore one-way mergeable

Corollary 3.1. *The GK algorithm is one-way mergeable, with a summary of size $O(\frac{1}{\varepsilon}\log(\varepsilon n))$.*

This analysis implies a step towards full-mergeability. We can apply the rule of always merging the summary of the smaller data set into the larger. This ensures that in the summarization of n items, any item participates in at most $\log(\varepsilon n)$ one-way merges (we only incur errors for data sets of at least $1/\varepsilon$ points). Thus the total error is $\varepsilon \log \varepsilon n$, and the summary has size $O(\frac{1}{\varepsilon}\log \varepsilon n)$. If we know n in advance, we can rescale ε by a $\log(\varepsilon n)$ factor, and achieve a space of $O(\frac{1}{\varepsilon}\log^2(\varepsilon n))$, matching the result of [18]. However, this does not achieve full mergeability, which does not allow foreknowledge of n.

3.2 Low-discrepancy-based summaries

In this section we mimic the merge-reduce algorithm [24, 10] for constructing deterministic ε-approximations of range spaces, but randomize it in a way so that error is preserved as required by full mergeability. Below we use $\mathsf{abs}(x)$ to denote the absolute value of x; we use $|X|$ to denote the cardinality of a set X.

Same-weight merges. We first consider a restricted merging model where each merge is applied only to two summaries representing data sets of the same size. Let S_1 and S_2 be the two summaries to be merged. Let $S' = S_1 \uplus S_2$, and sort S'. Then let S_e be all even points in the sorted order and S_o be all odd points in the sorted order. We retain either S_e or S_o with equal probability as our merged summary S. We call this a *same-weight merge*. Let us analyze this algorithm.

Lemma 3.1. *For any interval $I \in \mathcal{I}$, $2|I \cap S|$ is an unbiased estimator of $|I \cap S'|$ with error at most 1.*

We note that prior work has included similar ideas, but in a deterministic setting; since the worst-case error is bounded, it can be applied multiple times to build a summary of a larger data set [23]. However, in their case the error parameter ε grows with the merges. Suri *et al.* [31] also use a randomized merging process, but their analysis still allows the error parameter to increase after each level of merges. Below we describe a randomized merging process that preserves the error parameter after any number of merges.

Below we generalize the above lemma to multiple merges, but each merge is a *same-weight* merge. We set the summary size to be k_ε; if a data set contains fewer than k_ε points we simply set the summary to be the same as the data set. Let D be the entire data set of size n; note that the algorithm does not have *a priori* knowledge of n. We assume that n/k_ε is a power of 2. If not we can make it so by adding at most n dummy points and running the algorithm with $\varepsilon/2$ in place of ε. Thus, the whole merging process corresponds to a complete binary tree with $m = \log(n/k_\varepsilon)$ levels. Each internal node in the tree corresponds to the (same-weight) merge of its children. Let S be the final merged summary, corresponding to the root of the tree. Note that each point in S represents 2^m points in D.

Lemma 3.2. *If we set $k_\varepsilon = O((1/\varepsilon)\sqrt{\log(1/\delta)})$, then for any interval $I \in \mathcal{I}$ with probability at least $1-\delta$, $\mathsf{abs}(|I \cap D| - 2^m|I \cap S|) \le \varepsilon n$.*

Proof. Fix any I. We prove this lemma by considering the over-count error $X_{i,j}$ (which could be positive or negative) produced by a single merge of two sets S_1 and S_2 to get a set $S^{(j)}$ in round i. Then we consider the error $M_i = \sum_{j=1}^{r_i} X_{i,j}$ of all $r_i = 2^{m-i}$ merges in round i, and sum them over all m rounds using a single Chernoff-Hoeffding bound. The errors for all rounds form a geometric series that sums to at most εn with probability at least $1-\delta$.

Start the induction at round 1, before any sets are merged. Merging two sets S_1 and S_2 into $S^{(j)}$ causes the estimate $2|S^{(j)} \cap I|$ to have over-count error $X_{1,j} = 2|S^{(j)} \cap I| - |(S_1 \uplus S_2) \cap I|$. Now $\mathsf{abs}(X_{1,j}) \le 1 = \Delta_1$, by Lemma 3.1. There are $r_1 = 2^{m-1}$ such merges in this round, and since each choice of even/odd is made independently, this produces r_1 independent random variables $\{X_{1,1}, \ldots, X_{1,r_1}\}$. Let their total over-count error be denoted $M_1 = \sum_{j=1}^{r_1} X_{1,j}$. So, now except for error M_1, the set of r_1 sets $S^{(j)}$, each the result of an independent merge of two sets, can be used to represent $|D \cap I|$ by $2|(\biguplus_j S^{(j)}) \cap I|$.

So inductively, up to round i, we have accumulated at most $\sum_{s=1}^{i-1} M_s$ error, and have $2r_i$ point sets of size k_ε, where $r_i = 2^{m-i}$. We can again consider the merging of two sets S_1 and S_2 into $S^{(j)}$ by a same-weight merge. This causes the estimate $2^i|S^{(j)} \cap I|$ to have error $X_{i,j} = 2^i|S^{(j)} \cap I| - 2^{i-1}|(S_1 \uplus S_2) \cap I|$, where $\mathsf{abs}(X_{i,j}) \le 2^{i-1} = \Delta_i$, by Lemma 3.1. Again we have r_i such merges in this round, and r_i independent random variables $\{X_{i,1}, \ldots, X_{i,r_i}\}$. The total error in this round is $M_i = \sum_{j=1}^{r_i} X_{i,j}$, and except for this error M_i and $M_{i-1}, \ldots, M_1$, we can accurately estimate $|D \cap I|$ as $2^i|(\bigcup_j S^{(j)}) \cap I|$ using the r_i sets $S^{(j)}$.

We now analyze $M = \sum_{i=1}^{m} M_i$ using the following Chernoff-Hoeffding bound. Given a set $\{Y_1, \ldots, Y_t\}$ of independent random variables such that $\mathsf{abs}(Y_j - E[Y_j]) \le \Upsilon_j$, then for $T = \sum_{j=1}^{t} Y_j$ we can bound $\Pr[\mathsf{abs}(T - \sum_{j=1}^{t} E[Y_j]) > \alpha] \le 2e^{-2\alpha^2/(\sum_{j=1}^{t}(2\Upsilon_j)^2)}$. In our case the random variables are m sets of r_i

variables $\{X_{i,j}\}_j$ each with $E[X_{i,j}] = 0$ and $\mathsf{abs}(X_{i,j} - E[X_{i,j}]) = \mathsf{abs}(X_{i,j}) \le \Delta_i = 2^{i-1}$. There are m such sets for $i \in \{1, \ldots, m\}$. Setting $\alpha = h2^m$ for some parameter h, we can write

$$\begin{aligned}
\Pr\left[\mathsf{abs}(M) > h2^m\right] &\le 2\exp\left(-\frac{2\left(h2^m\right)^2}{\sum_{i=1}^{m}\sum_{j=1}^{r_i}(2\Delta_i)^2}\right) = 2\exp\left(-\frac{2\left(h2^m\right)^2}{\sum_{i=1}^{m}(r_i)(2^{2i})}\right) \\
&= 2\exp\left(-\frac{2h^2\left(2^{2m}\right)}{\sum_{i=1}^{m}(2^{m-i})(2^{2i})}\right) = 2\exp\left(-\frac{2h^2\left(2^{2m}\right)}{\sum_{i=1}^{m}2^{m+i}}\right) \\
&= 2\exp\left(-\frac{2h^2}{\sum_{i=1}^{m}2^{i-m}}\right) = 2\exp\left(-\frac{2h^2}{\sum_{i=1}^{m}2^{-i}}\right) < 2\exp\left(-2h^2\right).
\end{aligned}$$

Thus if we set $h = \sqrt{(1/2)\ln(2/\delta)}$, with probability at least $1-\delta$ we have $\mathsf{abs}(M) < h2^m = hn/k_\varepsilon$. Thus for $k_\varepsilon = O(h/\varepsilon)$ the error will be smaller than εn, as desired. □

This scheme can also produce an ε-approximation that is correct for *all* intervals $I \in \mathcal{I}$ with probability at least $1-\delta$. There is a set of $1/\varepsilon$ evenly spaced intervals $\mathcal{I}_\varepsilon$ such that any interval $I \in \mathcal{I}$ has $\mathsf{abs}(|D \cap I| - |D \cap I'|) \le \varepsilon n/2$ for some $I' \in \mathcal{I}_\varepsilon$. We can then apply the union bound by setting $\delta' = \delta\varepsilon$ and run the above scheme with $k_\varepsilon = O((1/\varepsilon)\sqrt{\log(1/\delta')})$. Then with probability at least $1-\delta$, no interval in $\mathcal{I}_\varepsilon$ has more than $\varepsilon n/2$ error, which means that no interval in $\mathcal{I}$ has more than εn error.

Theorem 3.2. *There is a same-weight merging algorithm that maintains a summary of size $O((1/\varepsilon)\sqrt{\log(1/\varepsilon\delta)})$ which is a one-dimensional ε-approximation with probability at least $1-\delta$.*

Uneven-weight merges. We next reduce uneven-weight merges to $O(\log(n/k_\varepsilon))$ weighted instances of the same-weight ones. This follows the so-called *logarithmic technique* used in many similar situations [18].

Set $k_\varepsilon = O((1/\varepsilon)\sqrt{\log(1/\varepsilon\delta)})$ as previously. Let n be the size of data set currently being summarized. We maintain $\log(n/k_\varepsilon)$ layers, where each layer has exactly k_ε points or is empty (we assume n/k_ε is an integer; otherwise we can always store the extra $\le k_\varepsilon$ points exactly without introducing any error). In the 0th layer, each point has weight 1, and in the ith layer, each point has weight 2^i.

We merge two such summaries S_1 and S_2 via same-weight merging, starting from the bottom layer, and promoting retained points to the next layer. At layer i, we may have $0, 1, 2$, or 3 sets of k_ε points each. If there are 0 or 1 such sets, we skip this layer and proceed to layer $i+1$; if there are 2 or 3 such sets we merge any two of them using a same-weight merge, and promote the merged set of k_ε points to layer $i+1$.

The analysis of this logarithmic scheme is straightforward because our same-weight merging algorithm preserves the error parameter ε across layers: Since each layer is produced by only same-weight merges, it is an ε-approximation of the set of points represented by this layer, namely the error is εn_i for layer i where n_i is the number of points being represented. Summing over all layers yields a total error of εn, without the algorithm requiring *a priori* knowledge of n.

Theorem 3.3. *There is a fully mergeable summary of size $O((1/\varepsilon)\sqrt{\log(1/\varepsilon\delta)}\log(\varepsilon n))$ which is a one-dimensional ε-approximation with probability at least $1-\delta$.*

3.3 Hybrid quantile summaries

Random sampling. A classic result [33, 32] shows that a random sample of $k_\varepsilon = O((1/\varepsilon^2)\log(1/\delta))$ points from D is an ε-approximation with probability $1-\delta$. So a fully mergeable summary for quantiles can be obtained by just retaining a random sample of D. A random sample is easily fully mergeable. A standard way of doing so is to assign a random value $u_i \in [0,1]$ for each point $p_i \in D$, and we retain in $S \subset D$ the k_ε elements with the largest u_i values. On a merge of two summaries S_1 and S_2, we retain the set $S \subset S_1 \cup S_2$ that has the k_ε largest u_i values from the $2k_\varepsilon$ points in $S_1 \cup S_2$. It is also easy to show that finite precision ($O(\log n)$ bits with high probability) is enough to break all ties.

Fact 3.1. *A random sample of size $k_\varepsilon = O((1/\varepsilon^2)\log(1/\delta))$ is fully mergeable and is an ε-approximation with probability at least $1-\delta$.*

We next show how to combine the approaches of random sampling and the low-discrepancy-based method to achieve a summary size independent of n. At an intuitive level, for a subset of points, we maintain a random sample of size about $(1/\varepsilon)\log(1/\varepsilon)$. The sample guarantees error about $\sqrt{\varepsilon}$ for any range, so we make sure that we only use this on a small fraction of the points (at most εn points). The rest of the points are processed using the logarithmic method. That is, we maintain $O(\log(1/\varepsilon))$ levels of the hierarchy, and only in the bottom level use a random sample. This leads to a summary of size approximately $(1/\varepsilon)\log(1/\varepsilon)$.

Hybrid structure. We now describe the summary structure in more detail for n points, where $2^{j-1}k_\varepsilon \leq n < 2^j k_\varepsilon$ for some integer j, and $k_\varepsilon = (4/\varepsilon)\sqrt{\ln(4/\varepsilon)}$. Let $g_\varepsilon = (64/\varepsilon^2)\ln(16/\varepsilon)$. For each level l between $i = j - \log_2(g_\varepsilon)$ and $j-1$ we either maintain k_ε points, or no points. Each point at the lth level has weight 2^l. The remaining $m \leq 2^i k_\varepsilon$ points are in a *random buffer* at level i, represented by a random sample of k_ε points (or only m if $m < k_\varepsilon$). Each point in the sample has weight m/k_ε (or 1 if $m < k_\varepsilon$). Note the total size is $O(k_\varepsilon \log(g_\varepsilon)) = O((1/\varepsilon)\log^{1.5}(1/\varepsilon))$.

Merging. Two hybrid summaries S_1 and S_2 are merged as follows. Let n_1 and n_2 be the sizes of the data sets represented by S_1 and S_2, and w.l.o.g. we assume $n_1 \geq n_2$. Let $n = n_1 + n_2$. Let j be an integer such that $2^{j-1}k_\varepsilon \leq n < 2^j k_\varepsilon$, and let $i = j - \log_2(g_\varepsilon)$.

First consider the random buffer in the merged summary; it now contains both random buffers in S_1 and S_2, as well as all points represented at level $i-1$ or below in either S_1 or S_2. Note that if $n_1 \geq 2^{j-1}k_\varepsilon$, then S_1 cannot have points at level $l \leq i-1$. Points from the random buffers of S_1 and S_2 already have u_i values. For every p of weight $w(p) = 2^l$ that was in a level $l \leq i-1$, we insert $w(p)$ copies of p into the buffer and assign a new u_i value to each copy. Then the k_ε points with the largest u_i values are retained.

When the random buffer is full, i.e., represents $2^i k_\varepsilon$ points, then it performs an "output" operation, and outputs the sample of k_ε points of weight 2^i each, which is then merged into the hierarchy at level i. It is difficult to ensure that the random buffer represents exactly $m = 2^i k_\varepsilon$ points when it outputs points, but it is sufficient if this occurs when the buffer has this size in expectation. There are two ways the random buffer may reach this threshold of representing m points:

1. On insertion of a point from the hierarchy of level $l \leq i-1$. Since copies of these points are inserted one at a time, representing 1 point each, it reaches the threshold exactly. The random buffer outputs and then inserts the remaining points in a new random buffer.
2. On the merge of two random buffers B_1 and B_2, which represent b_1 and b_2 points, respectively. Let $b_1 \geq b_2$, and let B be the union of the two buffers and represent $b = b_1 + b_2$ points. If $b < m$ we do not output; otherwise we have $m/2 \leq b_1 < m \leq b < 2m$. To ensure the output from the random buffer represents m points in expectation we either:
   (i) With probability $\rho = (b-m)/(b-b_1)$, we do not merge, but just output the sample of B_1 and let B_2 be the new random buffer.
   (ii) With probability $1-\rho = (m-b_1)/(b-b_1)$, output the sample of B after the merge, and let the new random buffer be empty.

   Note that the expected number of points represented by the output from the random buffer is $\rho b_1 + (1-\rho)b = \frac{b-m}{b-b_1}b_1 + \frac{m-b_1}{b-b_1}b = m$.

Next, the levels of the hierarchy of both summaries are merged as before, starting from level i. For each level if there are 2 or 3 sets of k_ε points, two of them are merged using a same-weight merge, and the merged set is promoted to the next level.

Analysis. First we formalize the upward movement of points; see proof in full version.

Lemma 3.3. *Over time, a point only moves up in the hierarchy (or is dropped): it never decreases in level.*

Now we analyze the error in this hybrid summary. We will focus on a single interval $I \in \mathcal{I}$ and show the over-count error X on I has $\mathsf{abs}(X) \leq \varepsilon n/2$ with probability $1 - \varepsilon/4$. Then applying a union bound will ensure the summary is correct for all $1/\varepsilon$ intervals in $\mathcal{I}_\varepsilon$ with probability at least $3/4$. This will imply that for all intervals $I \in \mathcal{I}$ the summary has error at most εn.

The total over-count error can be decomposed into two parts. First, we invoke Theorem 3.3 to show that the effect of all same-weight merges has error at most $\varepsilon n/4$ with probability at least $1 - \varepsilon/8$. This step assumes that *all* of the data that ever comes out of the random buffer has no error, it is accounted for in the second step. Note that the total number of merge steps at each level is at most as many as in Theorem 3.3, even those merges which are later absorbed into the random buffer. Second, (the focus of this analysis) we show the total error from all points that pass through the random buffer is at most $\varepsilon n/4$ with probability at least $1 - \varepsilon/8$. This step assumes that all of the weighted points put into the random buffer have no error, this is accounted for in the first step. So there are two types of random events that affect X: same-weight merges and random buffer outputs. We bound the effect of each event, independent of the result of any other event. Thus after analyzing the two types separately, we can apply the union bound to show the total error is at most $\varepsilon n/2$ with probability at least $1 - \varepsilon/4$.

It remains to analyze the effect on I of the random buffer outputs. First we bound the number of times a random buffer can output to level l, i.e., output a set of k_ε points of weight 2^l each; proof is deferred to full version. Then we quantify the total error attributed to the random buffer output at level l.

Lemma 3.4. *A summary of size n, for $2^{j-1}k_\varepsilon \leq n < 2^j k_\varepsilon$, has experienced $h_l \leq 2^{j-l} = 2^{i-l}g_\varepsilon$ random buffer promotions to level l within its entire merge history.*

Lemma 3.5. *When the random buffer promotes a set B of k_ε points representing a set P of m' points (where $m/2 < m' < 2m$), for any interval $I \in \mathcal{I}$ the over-count $X = (m/k_\varepsilon)|I \cap B| - |I \cap P|$ has expectation 0 and $\mathsf{abs}(X) \leq 2m$.*

Proof. The expectation of over-count X has two independent components. B is a random sample from P, so in expectation is has the correct proportion of points in any interval. Also, since $E[|P|] = m$, and $|B| = k_\varepsilon$, then m/k_ε is the correct scaling constant in expectation.

To bound $\mathsf{abs}(X)$, we know that $|P| < 2m$ by construction, so the maximum error an interval I could have is to return 0 when it should have returned $2m$, or vice-versa. So $\mathsf{abs}(X) < 2m$. □

Since $m \leq n/g_\varepsilon$ at level i, then $m \leq 2^{l-i}n/g_\varepsilon$ at level l, and we can bound the over-count error as $\Delta_l = \mathsf{abs}(X) \leq 2m \leq 2^{l-i+1}n/g_\varepsilon$. Now we consider a random buffer promotion that causes an over-count $X_{l,s}$ where $l \in [0, i]$ and $s \in [1, h_l]$. The expected value of $X_{l,s}$ is 0, and $\mathsf{abs}(X_{l,s}) \leq \Delta_l$. These events are independent so we can apply another Chernoff-Hoeffding bound on these $\sum_{l=0}^{i} h_l$ events. Recall that $g_\varepsilon = (64/\varepsilon^2)\ln(16/\varepsilon)$ and let $\hat{T} = \sum_{i=0}^{i}\sum_{s=1}^{h_l} X_{l,s}$, which has expected value 0. Then

$$\begin{aligned}
\mathsf{Pr}[\mathsf{abs}(\hat{T}) \geq \varepsilon n/4] &= 2\exp\left(-2\frac{(\varepsilon n/4)^2}{\sum_{l=0}^{i} h_l \Delta_l^2}\right) \leq 2\exp\left(-2\frac{(\varepsilon n/4)^2}{\sum_{l=0}^{i}\left(2^{i-l}g_\varepsilon\right)\left(2^{l-i+1}n/g_\varepsilon\right))^2}\right) \\
&\leq 2\exp\left(-g_\varepsilon\frac{\varepsilon^2}{8}\frac{1}{\sum_{l=0}^{i} 2^{i-l}2^{2(l-i)+2}}\right) = 2\exp\left(-g_\varepsilon\frac{\varepsilon^2}{32}\frac{1}{\sum_{l=0}^{i} 2^{l-i}}\right) \\
&= 2\exp\left(-2\ln(16/\varepsilon)\frac{1}{\sum_{l=0}^{i} 2^{-l}}\right) \leq 2\exp(-\ln(16/\varepsilon)) = 2(\varepsilon/16) = \varepsilon/8.
\end{aligned}$$

Theorem 3.4. *The above scheme maintains a fully mergeable one-dimensional ε-approximation of size $O(\frac{1}{\varepsilon}\log^{1.5}(1/\varepsilon))$, with probability at least $3/4$.*

References

[1] P. K. Agarwal, S. Har-Peled, and K. Varadarajan. Geometric approximations via coresets. *C. Trends Comb. and Comp. Geom. (E. Welzl)*, 2007.

[2] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measure of points. *J. ACM*, 51(4):2004, 2004.

[3] P. K. Agarwal, J. M. Phillips, and H. Yu. Stability of ε-kernels. In *Proceedings of 18th Annual European Symposium on Algorithms*, 2010.

[4] P. K. Agarwal and H. Yu. A space-optimal data-stream algorithm for coresets in the plane. In *Proceedings 23rd Annual Symposium on Computational Geometry*, 2007.

[5] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58:137–147, 1999.

[6] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *RANDOM*, 2002.

[7] R. Berinde, G. Cormode, P. Indyk, and M. Strauss. Space-optimal heavy hitters with strong error bounds. *ACM Transactions on Database Systems*, 35(4), 2010.

[8] T. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Computational Geometry: Theory and Applications*, 35:20–35, 2006.

[9] T. Chan. Dynamic coresets. In *SoCG*, pages 1–9, 2008.

[10] B. Chazelle and J. Matousek. On linear-time deterministic algorithms for optimization problems in fixed dimensions. *J. Algorithms*, 21:579–597, 1996.

[11] G. Cormode and M. Hadjieleftheriou. Finding frequent items in data streams. In *Proc. International Conference on Very Large Data Bases*, 2008.

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

[13] J. Feigenbaum, S. Kannan, M. J. Strauss, and M. Viswanathan. An approximate L1-difference algorithm for massive data streams. *SIAM J. Comput.*, 32(1):131–151, 2003.

[14] J. Feldman, S. Muthukrishnan, A. Sidiropoulos, C. Stein, and Z. Svitkina. On distributing symmetric streaming computations. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 2008.

[15] S. Ganguly and A. Majumder. CR-precis: A deterministic summary structure for update data streams. In *ESCAPE*, 2007.

[16] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. How to summarize the universe: Dynamic maintenance of quantiles. In *Proc. International Conference on Very Large Data Bases*, 2002.

[17] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2001.

[18] M. Greenwald and S. Khanna. Power conserving computation of order-statistics over sensor networks. In *Proc. ACM Symposium on Principles of Database Systems*, 2004.

[19] D. M. Kane, J. Nelson, and D. P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proc. ACM Symposium on Principles of Database Systems*, 2010.

[20] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. In *Proc. Symposium on Operating Systems Design and Implementation*, 2002.

[21] A. Manjhi, S. Nath, and P. B. Gibbons. Tributaries and deltas: efficient and robust aggregation in sensor network streams. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2005.

[22] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston. Finding (recently) frequent items in distributed data streams. In *Proc. IEEE International Conference on Data Engineering*, 2005.

[23] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *Proc. ACM SIGMOD International Conference on Management of Data*, 1998.

[24] J. Matousek. Approximations and optimal geometric divide-and-conquer. In *SToC*, pages 505–511, 1991.

[25] J. Matousek. *Geometric Discrepancy; An Illustrated Guide*. Springer, 1999.

[26] A. Metwally, D. Agrawal, and A. E. Abbadi. An integrated efficient solution for computing frequent and top-k elements in data streams. *ACM Transactions on Database Systems*, 2006.

[27] J. Misra and D. Gries. Finding repeated elements. *Sc. Comp. Prog.*, 2:143–152, 1982.

[28] J. Nelson and D. P. Woodruff. Fast manhattan sketches in data streams. In *Proc. ACM Symposium on Principles of Database Systems*, 2010.

[29] J. M. Phillips. Algorithms for ε-approximations of terrains. In *ICALP*, 2008.

[30] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: New aggregation techniques for sensor networks. In *Proc. ACM SenSys*, 2004.

[31] S. Suri, C. D. Tóth, and Y. Zhou. Range counting over multidimensional data streams. In *Proceedings 20th Symposium on Computational Geometry*, pages 160–169, 2004.

[32] M. Talagrand. Sharper bounds for Gaussian and emperical processes. *Annals of Probability*, 22:76, 1994.

[33] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *The. of Prob. App.*, 16:264–280, 1971.

[34] H. Yu, P. K. Agarwal, R. Poreddy, and K. R. Varadarajan. Practical methods for shape fitting and kinetic data structures using coresets. In *SoCG*, 2004.

[35] H. Zarrabi-Zadeh. An almost space-optimal streaming algorithm for coresets in fixed dimensions. In *ESA*, pages 817–829, 2008.