Measuring Independence of Datasets

Vladimir Braverman[*]
UCLA
vova@cs.ucla.edu

Rafail Ostrovsky[†]
UCLA
rafail@cs.ucla.edu

ABSTRACT

Approximating pairwise, or k-wise, independence with sublinear memory is of considerable importance in the data stream model. In the streaming model the joint distribution is given by a stream of k-tuples, with the goal of testing correlations among the components measured over the entire stream. Indyk and McGregor (SODA 08) recently gave exciting new results for measuring pairwise independence in this model.

Statistical distance is one of the most fundamental metrics for measuring the similarity of two distributions, and it has been a metric of choice in many papers that discuss distribution closeness. For pairwise independence, the Indyk and McGregor methods provide $\log n$-approximation under statistical distance between the joint and product distributions in the streaming model. Indyk and McGregor leave, as their main open question, the problem of improving their $\log n$-approximation for the statistical distance metric.

In this paper we solve the main open problem posed by Indyk and McGregor for the statistical distance for pairwise independence and extend this result to any constant k. In particular, we present an algorithm that computes an (ϵ, δ)-approximation of the statistical distance between the joint and product distributions defined by a stream of k-tuples. Our algorithm requires $O\left(\left(\frac{1}{\epsilon}\log(\frac{nm}{\delta})\right)^{(30+k)^k}\right)$ memory and a single pass over the data stream.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY

[*]Supported in part by NSF grants 0716835, 0716389, 0830803, 0916574.

[†]Supported in part by IBM Faculty Award, Xerox Innovation Group Award, the Okawa Foundation Award, Lockheed Martin, Intel, Teradata, NSF grants 0716835, 0716389, 0830803, 0916574, BSF grant and U.C. MICRO grant.

General Terms

Algorithms, Theory

Keywords

Data Streams, Randomized Algorithms, Dimension Reduction, Theory of Computation.

1. INTRODUCTION

Finding correlations between columns of a table is a fundamental problem in databases. Virtually all commercial databases construct query plans for queries that employ cross-dimensional predicates. The basic step is estimating "selectivity" (i.e., the number of rows that satisfy the predicate conditions) of the complex predicate. Without any prior knowledge, the typical solution is to compute selectivity of each column separately and use the multiplication as an estimate. Thus, optimizers make a "statistical independence assumption" which sometimes may not hold. Incorrect estimations may lead to suboptimal query plans and decrease performance significantly. Identifying correlations between database columns by measuring a level of independence between columns has a long history in the database research community (see, e.g., Poosala and Ioannidis [26]). For data warehouses, it is important to find correlated columns for correct schema construction, as Kimball and Caserta note in [23]. In practice, typical solutions for finding correlations between columns are either histograms (see e.g., [26]) or sampling (see e.g., Ilyas, Markl, Haas, Brown, Aboulnaga [19]). These methods have their natural disadvantages, i.e., they do not tolerate deletions and may require several passes over the data. When it comes to very large data volumes, it is critical to maintain sublinear in terms of memory solutions that do not require additional passes over the data and can tolerate incremental updates of the data, e.g., deletions.

For these purposes, a theoretical *data stream model* can be useful. For data warehouses, the "loading" phase of the ETL process (see e.g., Kimball and Caserta [23]) can be seen as a data stream. When reading a database table, the process can be considered as a stream of data tuples. Thus, the data stream model represents another setting where approximating pairwise or k-wise independence with sublinear memory is of considerable importance.

1.1 Precise Definition of the Problem

The natural way to model database tables in a streaming model is by considering a stream of tuples. In this paper we consider a stream of k-tuples $(i_1, \ldots, i_k)$ where $i_l \in [n]$. (For simplicity, we assume that elements of all columns are drawn from the same domain, even though our approach trivially extends to a general case of different domains.) As pointed out in [26, 19, 21], the natural way to define a joint distribution of two (or more) columns is given by the frequencies of all combinations of coordinates. Similarly, the distribution of each column is defined by the corresponding set of frequencies; the definition of a product distribution follows. Let us define these notions precisely.[1]

Definition 1.1. *Let D be a stream of elements $p_1, \ldots, p_m$, where each stream element is a k-tuple $\mathbf{i} = (\mathbf{i}_1, \ldots, \mathbf{i}_k)$, where $\mathbf{i}_l \in [n]$. Here m is the size of stream D. A* frequency *of a tuple $\mathbf{i} \in [n]^k$ is defined as the number of times it appears in D: $f_{\mathbf{i}} = |\{j : p_j = \mathbf{i}\}|$. For $l \in [k]$, a l-th* margin frequency *of $t \in [n]$ is the number of times t appears as a l-th coordinate: $f_l(t) = \sum_{\mathbf{i} \in [n]^k, \mathbf{i}_l = t}^{n} f_{\mathbf{i}}$. A* joint distribution *is defined by a vector of probabilities $P_{joint}(\mathbf{i}) = \frac{f_{\mathbf{i}}}{m}, \mathbf{i} \in [n]^k$. A l-th* margin distribution *is defined by a vector of probabilities $P_l(t) = \frac{f_l(t)}{m}$, $t \in [n]$. A* product distribution *is defined as: $P_{product}(\mathbf{i}) = \prod_{l=1}^{k} P_l(\mathbf{i}_l), \mathbf{i} \in [n]^k$.*

Statistical distance is one of the most fundamental metrics for measuring the similarity of two distributions, and it has been a metric of choice in many papers that discuss distribution closeness (see e.g., [1, 2, 4, 6, 21, 28, 27]). Given two distributions over a discrete domain, the statistical distance is half of L_1 distance between the probability vectors.

Definition 1.2. *Consider two distributions over a finite domain Ω given by two random variables V, U.* Statistical distance *$\Delta(V, U)$ is defined as:*

$$\Delta(V, U) = \frac{1}{2} \sum_{x \in \Omega} |P(V = x) - P(U = x)|.$$

In particular, one of the most common methods of measuring independence is computing statistical distance between product and joint distributions (see e.g., [4, 21]). This is precisely the way we define our problem:

Definition 1.3. *An* Independence Problem *is the following: Given stream D of k-tuples, $(1 \pm \epsilon)$-approximate, with one pass over D, with small memory the statistical distance between joint and product distribution $\Delta(P_{joint}, P_{product})$.*

In the streaming model, Indyk and McGregor [21] recently gave exciting new results for measuring pairwise independence, i.e., for $k = 2$. To measure the independence, they consider two metrics: L_2 and L_1. Recall that the L_2 distance between two probability distributions is a L_2 distance of their probability vectors. In particular, the Independence Problem under the L_2 metric is defined as $\|P_{joint} - P_{product}\|_2$. For the L_2 metric and $k = 2$, Indyk and McGregor give an $(1 \pm \epsilon)$-approximation using polylogarithmic space. Recently Braverman, Chung, Liu, Mitzenmacher and Ostrovsky [8] generalized the L_2 results of [21] to any constant k.[2] However, it is well known that for probability distributions, statistical distance is a significantly more powerful metric then the L_2 metric. For instance, consider two distributions on $[2n]$, where the first distribution is uniform on $\{1, \ldots, n\}$ and the second is uniform on $\{n+1, \ldots, 2n\}$. In this case the statistical distance is 1 but the L_2 distance is $\sqrt{2/n} \to 0$. We refer a reader to a paper of Batu, Fortnow, Rubinfeld, Smith and White [5] for a further discussion.

For the statistical distance L_1 metric and $k = 2$, the Indyk and McGregor methods provide $\log n$-approximation with polylogarithmic memory. In addition to $\log n$-approximation, Indyk and McGregor give an $(1 \pm \epsilon)$-approximation that requires $\Omega(n)$ memory, and also give a method that requires two passes to solve a promise problem for a restricted range of parameters. Indyk and McGregor leave, as their main open question, the problem of improving their $\log n$-approximation for the statistical distance metric.

In this paper we solve the main open problem posed by Indyk and McGregor for the statistical distance for pairwise independence and extend this result to any constant k. In particular, we present an algorithm that computes an (ϵ, δ)-approximation of the statistical distance between the joint and product distributions defined by a stream of k-tuples. Our algorithm requires $O\left(\left(\frac{1}{\epsilon}\log(\frac{nm}{\delta})\right)^{(30+k)^k}\right)$ memory and a single pass over the data stream. Theorem 2.5 formally describes our main result. We stress that we omit from this version optimization of constants and polylog factors (including exponents) in our constructions and analysis, which we leave for the journal version of this paper.

1.2 Implicit Tensors

It is convenient to present an alternative, equivalent formulation of the *Independence Problem* as well. We can consider the problem of approximating the sum of absolute values of a *tensor M_{Ind}*.

Definition 1.4. *An s-dimensional* tensor *M is a s-dimensional array with indexes in the range $[n]$; that is, M has an entry for each $\mathbf{i} \in [n]^s$. We denote by $m_{\mathbf{i}}$ the $\mathbf{i}$-th entry of M for each $\mathbf{i} \in [n]^s$.*

Definition 1.5. *Let M be a s-dimensional tensor with entries $m_{\mathbf{i}}, \mathbf{i} \in [n]^s$. An L_1-norm of M is a $|M| = \sum_{\mathbf{i} \in [n]^s} |m_{\mathbf{i}}|$.*

For example, a 1-dimensional tensor is an n-dimensional vector, a 2-dimensional tensor is an $n \times n$-matrix and so forth.

Many streaming problems address *explicitly* defined vectors (or matrices) where entries are equal to frequencies of corresponding stream elements. The Independence problem diverges from this setting; e.g., for pairwise independence, a pair (i, j) affects all entries in i-th row and j-th column of the product probability matrix. To reflect this important difference we consider the case where the entries of a tensor are defined *implicitly* by a data stream.

Definition 1.6. *Let $\mathcal{D}$ be a collection of data streams of size m of elements from domain Ω. Let $\mathcal{F} : \mathcal{D} \times [n]^s \mapsto R$ be a fixed function. We say that s-dimensional tensor M with entries $m_{\mathbf{i}} = \mathcal{F}(D, \mathbf{i}), \mathbf{i} \in [n]^s$ is* implicity defined *by $\mathcal{F}$, given D. We denote an implicitly defined tensor as $\mathcal{F}(D)$.*

[1]Here and henceforth, we use lowercase Latin characters for indexes. We use an italic font for integers and a boldface font for multidimensional indexes, e.g., $i \in [n]$ and $\mathbf{i} \in [n]^k$. For a multidimensional index, we use subscript to indicate its coordinate, e.g., $\mathbf{i}_1$ indicates the first coordinate of $\mathbf{i}$.

[2]It is worth pointing out that our methods in [8] are completely different and do not seem to apply to the L_1 problem.

Definition 1.7. *Let $\mathcal{D}$ be a collection of data streams of size m of k-tuples from domain $[n]^k$. A k-wise* Independence Function *$\mathcal{F}_{Ind} : \mathcal{D} \times [n]^k \mapsto R$ is a function defined as $\mathcal{F}_{Ind}(D, \mathbf{i}) = m^k f_{\mathbf{i}} - \prod_{l=1}^{k} f_l(\mathbf{i}_l)$ for $\mathbf{i} \in [n]^k$. Here $f_{\mathbf{i}}$ is given by Definition 1.1.* Statistical distance tensor *M_{Ind} is a k-dimensional tensor implicitly defined by $\mathcal{F}_{Ind}$, i.e., $M_{Ind} = \mathcal{F}_{Ind}(D)$.*

The main objective of our paper is approximating $|M_{Ind}|$. In particular, this implies solving the Independence problem since $\Delta(P_{joint}, P_{product}) = \frac{1}{m^k}|M_{Ind}|$, and since $m = |D|$ can be computed precisely. We thus freely interchange the notions of the Independence Problem and computing $|M_{Ind}|$. In fact, our approach is applicable to any function $\mathcal{F}$ for which conditions of our main theorems are true.

1.3 Why Existing Methods for Estimating L_1 Do Not Work

Alon, Matias and Szegedy [3] initiated the study of computing norms of vectors defined by a data stream. In their setting vector entries are defined by frequencies of the corresponding elements in the stream. Their influential paper was followed by a sequence of results including, among many others, works by Bhuvanagiri, Ganguly, Kesh and Saha [7]; Charikar, Chen and Farach-Colton [14]; Cormode and Muthukrishnan [15, 16]; Feigenbaum, Kannan, Strauss and Viswanathan [17]; Ganguly and Cormode [18]; Indyk [20]; Indyk and Woodruff [22]; and Li [24] as well as work of authors [9, 10, 13].

There is an important difference between settings of [3] and the Independence problem. Indeed, while the entries of the independence tensor M_{Ind} are defined by frequencies of tuples, there is no linear dependence. As a result, the aforementioned algorithms are not directly applicable to the Independence problem.

To illustrate this point, consider the celebrated method of stable distributions by Indyk [20]. For L_1 norm, Indyk observed that a polylogarithmic (in terms of n and m) number of sketches of the form $\sum C_i v_i$ gives an $(1 \pm \epsilon)$-approximation of $|V|$, when C_i are independent random variables with Cauchy distribution. Let us discuss the applicability of this method to the problem of pairwise independence. A sketch $\sum_{\mathbf{i}} C_{\mathbf{i}} m_{\mathbf{i}}$, $\mathbf{i} \in [n]^2$, would solve this problem; unfortunately, it is not clear how to construct a sketch in this form. In particular, the probability matrix of the product distribution is given implicitly as two vectors of margin sketches. It is not hard to construct sketches for margin distributions; however, it is not at all clear how to obtain a sketch for product distribution without using a multiplication of margin sketches. On the other hand, if we do use a multiplication of margin sketches (this is the approach of Indyk and McGregor), the random variable that is associated with the tensor's elements is a product of independent Cauchy variables. Therefore, random variables for distinct entries are *not independent*, and thus typical arguments used for stable distribution methods do not work anymore. In fact, the main focus of the Indyk and McGregor analysis is to overcome this problem:

> *"Perhaps ironically, the biggest technical challenges that arise relate to ensuring that different components of our estimates are sufficiently independent."*

For pairwise independence, Indyk and McGregor use the product of two Cauchy variables, where one of them is "truncated." Using elegant observations, they show that such a sketch allows achieving $\log n$-approximation of the statistical distance. Unfortunately, it is not clear how the method of a Cauchy product can be improved at all, since the $\log n$ factor is a necessary component of their seemingly tight analysis.

1.4 A Description of Our Approach

As we discuss below, solving the Independence problem requires developing multiple new tools and using them jointly with known methods.

Dimension Reduction for Implicit Tensors. Our solution can be logically divided into three steps which are explained, informally, below.

First, we prove that given a *polylog*-approximation algorithm for k-dimensional tensors and an ϵ-approximation algorithm for a special type of $(k-1)$-dimensional tensors, it is possible to derive an ϵ-approximation algorithm on k-dimensional tensors, where the resulting algorithm increases memory bound by a factor $O((\frac{1}{\epsilon} \log \frac{nm}{\delta})^{O(1)})$. Thus, we can *trade* dimensionality and precision for memory. To illustrate this step, consider pairwise independence. There exist an ϵ-approximation algorithm on vectors [20] and a $\log n$-approximation algorithm on matrices [21]. We show that these algorithms can be used to obtain an ϵ-approximation algorithm on matrices. This informal idea is stated precisely as Dimension Reduction Theorem 2.1. This theorem is the main technical contribution of our paper; the majority of the paper is devoted to establishing its validity.

Second, given a *polylog*-approximation algorithm for k-dimensional tensors and an ϵ-approximation algorithm on vectors, we can derive an ϵ-approximation algorithm on k-dimensional tensors by applying the Dimension Reduction Theorem recursively k-times. The memory will be increased by a factor roughly $O((\frac{1}{\epsilon} \log \frac{nm}{\delta})^{(30+k)^k})$ which is $O((\frac{1}{\epsilon} \log \frac{nm}{\delta})^{O(1)})$ for constant k. This informal idea is stated precisely as Theorem 2.2.

Third, we show that the conditions for Theorem 2.2 hold for the Independence problem. These results are stated in Lemmas 2.4 and 2.3, and in fact are a generalization of results from [20, 21].

The rest of our discussion is devoted to a description of the main ideas behind the Dimension Reduction Theorem.

Hyperplanes and Absolute Vectors. Consider a matrix M; a very natural idea to approximate $|M|$ is by approximating a L_1 norm of a vector with entries equal to L_1 norms of rows of M. We generalize this idea to tensors by defining the following operators.

Definition 1.8. *For any $s, t \geq 0$, we denote by $(,)$ a mapping from $[n]^s \times [n]^t$ to $[n]^{s+t}$ obtained by concatenation of coordinates. For instance, $((1,2),3)$ is a an element from $[n]^3$ with coordinates $1, 2, 3$ respectfully.*

Definition 1.9. *Let M be a s-dimensional tensor with entries $m_{\mathbf{j}}, \mathbf{j} \in [n]^s$. For any $l \in [n]$, Hyperplane(M, l) is a $(s-1)$-dimensional tensor with entries $m_{(l,\mathbf{i})}$ for $\mathbf{i} \in [n]^{s-1}$.*

For example, when $k = 2$, the l-th hyperplane of a matrix M is its l-th row.

Definition 1.10. *An l-th hyperplane is α-*significant *if $|Hyperplane(M,l)| \geq \alpha|M|$.*

For example, when $k = 2$, the l-th row is α-significant[3] if the L_1-norm of the vector defined by the l-th row carries at least α-fraction of $|M|$.

Definition 1.11. *For a s-dimensional tensor M, an $AbsoluteVector(M)$ is a vector of dimensionality n with entries $|Hyperplane(M,l)|, l \in [n]$. In particular, $|AbsoluteVector(M)| = |M|$.*

Projected Dimensions. To prove Dimension Reduction Theorem 2.1 we need to map s-dimensional tensors to $(s-1)$-dimensional tensors with a small distortion of L_1. We come up with the following mapping.

Definition 1.12. *Let M be a s-dimensional tensor with entries $m_{\mathbf{l}}$, where $\mathbf{l} \in [n]^s$, and let $0 \leq t \leq s$. A* Suffix-Sum *tensor $T_t(M)$ is a $(s-t)$-dimensional tensor with entries (for each $\mathbf{i} \in [n]^{s-t}$):*

$$m'_{\mathbf{i}} = \sum_{\mathbf{j} \in [n]^t} m_{(\mathbf{j},\mathbf{i})}$$

Also, we define $T_0(M) = M$. In other words, the $\mathbf{i}$-th entry of $T_t(M)$ is obtained by summing all elements of M with the $(s-t)$-suffix equal to $\mathbf{i}$. In particular, $T_s(M)$ is a scalar that is equal to $\sum_{\mathbf{i} \in [n]^s} m_{\mathbf{i}}$.

For matrix M with entries $m_{i,j}$, the Suffix-Sum operator $T_1(M)$ defines a vector V with entries $v_j = \sum_i m_{i,j}$. In other words, all entries of M that belong to the same columns (i.e., have the same second coordinate, i.e., the same "suffix") are "summed-up" to generate a single entry of V. Note that the Suffix-Sum operator is different from the AbsoluteVector operator. In the latter case we sum up the absolute values that belong to the same hyperplane, i.e., have identical prefix; in the former case we sum up all elements (and not their absolute values) that have an identical suffix.

Clearly $|T_1(M)| \leq |M|$; however, it is possible in general that $|T_1(M)| \ll |M|$. The key observation is that in some cases $|T_1(M)| \sim |M|$ and thus we can use an approximation of $|T_1(M)|$ to approximate $|M|$. To illustrate this point, consider a matrix M with entries $m_{i,j}$ that contains a very "significant" row i (i.e., $\sum_j |m_{i,j}| \sim |M|$). The key observation is that in this case $|T_1(M)| \sim |M|$; thus, if there is a significant row, it can approximated using $|T_1(M)|$. The same idea is easily generalized: if a s-dimensional tensor M contains a $(1-\epsilon)$-significant hyperplane $Hyperplane(M,l)$, then $|T_1(M)|$ is an 2ϵ-approximation of $|Hyperplane(M,l)|$. We prove this statement in Fact 3.6.

Note that $T_1(M)$ is a $(s-1)$-dimensional tensor; if M is a matrix, then $T_1(M)$ is a vector for which we can apply methods from [20]. Thus, approximating $|T_1(M)|$ is potentially an easier problem.

[3]It is important to note that the notion of α-significant is *different* from the notion of α-major in our other paper in this STOC [12]. In particular, here, we compare an element with the entire sum, while in [12] we compare an element with the sum of all elements except of that element. Further, notice that α-significant ≤ 1, while α-major can be arbitrary large.

Certifying Tournaments. We have shown that $T_1(M)$ can be useful for approximating $|M|$. However, when can we rely on the value of $|T_1(M)|$? In particular, how can we distinguish between the cases when there is a heavy hyperplane (and thus $|T_1(M)|$ is a good approximation) and the case when there is no heavy hyperplane (and thus $|T_1(M)|$ does not contain reliable information)? The second key observation is that it can be done using "certifying tournaments." To illustrate this point, consider again the case $k = 2$, where M is a matrix. Split M into two random sub-matrices by sampling the rows w.p. 1/2. If there is a heavy row, then with probability close to 1, one sub-matrix will have a significantly larger norm then the other. Recall that the method of [21] gives us a $\log n$-approximation. Thus, for very heavy rows, the *ratio* between approximations of norms obtained by the method from [21] will be large. On the other hand, we show that if there are no heavy rows, then such behavior is quite unlikely to be observed many times. Thus, there exists a way to distinguish between the first and the second cases for $(1 - \frac{\epsilon}{\log^2 n})$-significant rows.[4]

The method of certifying tournaments can be generalized to any $s \leq k$ as follows. Let M be a s-dimensional tensor with entries $m_{\mathbf{i}}$ for $\mathbf{i} \in [n]^s$. We "split" M into two "sampled" s-dimensional tensors M^0 and M^1 by randomly sampling the first coordinate. That is, M^1 has entries $m_{\mathbf{i}} H(\mathbf{i}_1)$ and M^0 has entries $m_{\mathbf{i}}(\mathbf{1} - H(\mathbf{i}_1))$, where $H : [n] \mapsto \{0,1\}$ is pairwise independent and uniform. If there exists a β-approximation algorithm for sampled tensors, and there exists an ϵ-approximation algorithm for Suffix-Sum, $|T_1(M^0)|$ and $|T_1(M^0)|$, then we can approximate L_1 norm of significant hyperplanes. Indeed, if there exists a significant hyperplane M_l of M, then the ratio between β-approximations of $|M^0|$ and $|M^1|$ will be large. If this is the case, the approximation of $T(M^{H(l)})$ is also an ϵ-approximation of $|M_l|$.

To summarize, our main technical Theorem 4.3 proves that it is possible to output a number U such that U is either an approximation of some hyperplane or 0. Further, if there exists a $(1 - \frac{\epsilon}{\beta^2})$-significant hyperplane, then with high probability, U is its approximation. We call such an algorithm an α-ThresholdMax algorithm, for $\alpha = O(\frac{\epsilon}{\beta^2})$.

Indirect Sampling. Many streaming algorithms compute statistics on *sampled* streams, which are random subsets of D defined by some randomness $\mathcal{H}$. In many cases, a sampled stream directly corresponds to a collection of sampled entries of a frequency vector. In contrast, subsets of D do not correspond directly to entries M_{Ind}. Thus, our algorithms employ *indirect* sampling, where randomness defines sampled entries of M_{Ind} rather then the entries of a data stream D. We define a Prefix-Zero operator.

Definition 1.13. *Let M be a s-dimensional tensor with entries $m_{\mathbf{i}}, \mathbf{i} \in [n]^s$ and let $H_1, \ldots, H_t, t \leq s$ be hash functions $H_j : [n] \mapsto \{0,1\}$. A* Prefix-Zero *tensor $W(M, H_1, \ldots, H_t)$ a is a s-dimensional tensor with entries $m_{\mathbf{i}} \prod_{l=1}^{t} H_l(\mathbf{i}_l)$.*

Our algorithms work with tensors that are defined by com-

[4]It is worth noting that the idea of "split-and-compare" is not new. Group testing [16] exploits a similar approach. However, the methods from [16] require ϵ-approximation of L_1; in contrast, we use certifying tournaments to improve the approximation.

positions of $\mathcal{F}_{Ind}$, Prefix-Zero and Suffix-Sum. We thus extend the definition of implicitly defined tensors.

Definition 1.6. (Revised) *Let $\mathcal{D}$ be a collection of data streams of size m of elements from domain Ω and let $\mathfrak{H}$ be a collection of hash functions from $[n]$ to $\{0,1\}$. Let $\mathcal{F} : \mathcal{D} \times \mathfrak{H}^t \times [n]^s \mapsto R$ be a fixed function, for some $0 \leq t \leq s$. We say that a s-dimensional tensor M with entries $m_{\mathbf{i}} = \mathcal{F}(D, \mathcal{H}, \mathbf{i}), \mathbf{i} \in [n]^s$ is* implicity defined *by $\mathcal{F}$, given $D \in \mathcal{D}$ and $\mathcal{H} \in \mathfrak{H}^t$. We denote an implicitly defined tensor as $\mathcal{F}(D, \mathcal{H})$.*

EXAMPLE 1.14. *Consider $k = 2$. Then $\mathcal{F}'(D, H) = W(\mathcal{F}_{Ind}(D), H)$ defines a matrix that represents a collection of rows sampled by a hash function $H : [n] \mapsto \{0,1\}$.*

Generalizing the Method of Indyk and Woodruff [22]. The ThresholdMax algorithm solves the problem that resembles the well-known problem of finding an element with maximal frequency, see, e.g., [14] and [15]. The celebrated method of Indyk and Woodruff [22] uses maximal entries to estimate L_p norms on vectors defined by frequencies. We apply the ideas of [22] to approximate $|AbsoluteVector(M)| = |M|$.

Unfortunately, the method of Indyk and Woodruff [22] is not directly applicable since some basic tools available for frequency vectors (such as L_2 norm approximation) cannot be used. We propose a different algorithm which is still in the same spirit as [22]. We prove Lemmas 5.5 and 5.3 which state that an existence of an α-ThresholdMax algorithm for an implicitly defined vector V implies an existence of an (ϵ, δ)-approximation algorithm for $|V|$, with memory increased by an additional factor of $\frac{1}{\alpha} poly(\frac{1}{\epsilon} \log \frac{nm}{\delta})$.

Other Technical Issues. There are several other technical issues that need to be resolved. We need to prove that the methods of Indyk [20] and Indyk and McGregor [21] are applicable for k-dimensional tensors that are obtained from M_{Ind} by applying Prefix-Zero and Suffix-Sum operators. We prove these claims in Lemmas 2.4 and 2.3. To prove our main theorems, certain properties of the operations on tensors should be established. We prove these in Section 3.

1.5 Related Work

Measuring pairwise independence between two or more random variables is a fundamental problem that touches many areas of computer science. The problems of efficiently testing pairwise, or k-wise, independence were recently considered by Alon, Andoni, Kaufman, Matulef, Rubinfeld and Xie [1]; Alon, Goldreich and Mansour [2]; Batu, Fortnow, Fischer, Kumar, Rubinfeld and White [4]; and Batu, Kumar and Rubinfeld [6]. These works address the problem of minimizing the number of samples needed to obtain sufficient approximation, when the joint distribution is accessible through a sampling procedure. Unlike the work in [1, 2, 4, 6], in the streaming model, the joint distribution is given by a stream of tuples.

2. MAIN THEOREMS

The proof of our result is based on three main steps which are summarized by the following theorems. The remainder of this paper is devoted to establishing these theorems.

THEOREM 2.1. ***Dimension Reduction for Implicit Tensors.*** *Let $s \geq 1$ and let M be a s-dimensional tensor with $poly(n, m)$-bounded entries that is defined by a function $\mathcal{F}$, i.e., $M = \mathcal{F}(D, \mathcal{H})$ where D is a data stream and $\mathcal{H}$ is a fixed randomness. Let $H : [n] \mapsto \{0,1\}$ be an arbitrary fixed hash function. Assume that*

1. *There exists an algorithm $\mathfrak{A}(D, \mathcal{H}, H, \delta)$ that, given D and an access to $\mathcal{H}$ and H, in one pass obtains $(\log^k(n), \delta)$-approximation of $|W(M, H)|$;*
2. *There exists an algorithm $\mathfrak{B}(D, \mathcal{H}, H, \epsilon, \delta)$ that, given D and an access to $\mathcal{H}$ and H, in one pass obtains an (ϵ, δ)-approximation of $|T_1(W(M, H))|$;*
3. *Both algorithm require memory $\nu(n, m, \epsilon, \delta) \leq O((\frac{1}{\epsilon} \log \frac{nm}{\delta})^{(30+k)^s})$, beyond the memory required for H and $\mathcal{H}$.*

Then there exists an algorithm that, given an access to $\mathcal{H}$, in one pass obtains an (ϵ, δ)-approximation of $|M|$ using memory $(\frac{1}{\epsilon} \log \frac{nm}{\delta})^{(30+k)^{s+1}}$.

PROOF. Follows from Theorem 4.3, Lemma 5.5, Lemma 5.3 and elementary computations.

Indeed, the assumptions of Theorem 2 imply, by Theorem 4.3, an existence of a $\frac{\epsilon}{\log^{2k}(n)}$-ThresholdMax algorithm (see Definition 4.2) for restricted function $\mathcal{F}' = AbsoluteVector(\mathcal{F}(D, \mathcal{H}))$. The existence of a ThresholdMax algorithm implies, by Lemma 5.3, the existence of a Cover algorithm (see Definition 5.2) for $AbsoluteVector(\mathcal{F}(D, \mathcal{H}))$. The assumption that the entries of M are polynomially bounded and Fact 3.7 imply that the entries of $AbsoluteVector(\mathcal{F}(D, \mathcal{H}))$ are polynomially bounded as well. By Lemma 5.5, there exists an (ϵ, δ)-approximation algorithm for $|AbsoluteVector(\mathcal{F}(D, \mathcal{H}))|$ and $|AbsoluteVector(\mathcal{F}(D, \mathcal{H}))| = \sum_{i \in [n]} |Hyperplane(\mathcal{F}(D, \mathcal{H}), l)| = |M|$.

After substituting the parameters, the memory required is less than (for sufficiently large n) $\frac{1}{\epsilon^{30}} \log(\frac{1}{\delta}) \log^{2k+20}(nm) \nu(n, m, \frac{\epsilon^7}{\log^4(nm)}, \frac{\epsilon^{17}}{\log^{2k}(n) \log^8(mn)}) \leq (\frac{1}{\epsilon} \log \frac{nm}{\delta})^{(30+k)^{s+1}}$. $\square$

THEOREM 2.2. ***Approximation Theorem for Tensors*** *Let M be a k-dimensional tensor with entries bounded by $poly(n, m)$ and implicitly defined by a function $\mathcal{F}(D)$. Assume that*

1. *There exists an algorithm $\mathfrak{B}_s(D, H_1, \ldots, H_s)$ (for some $s < k$) that, given D and an access to fixed hash functions $H_1, \ldots, H_s$, in one pass obtains an (ϵ, δ)-approximation of $|T_s(W(M, H_1, \ldots, H_s))|$;*
2. *There exist algorithms $\mathfrak{A}_{s_1, s_2}(D, H_1, \ldots, H_{s_1})$ (for any $0 \leq s_2 \leq s_1 \leq s$) that, given D and an access to H_is, in one pass obtain a $(\log^k(n), \delta)$-approximation of $|T_{s_2}(W(M, H_1, \ldots, H_{s_1}))|$;*
3. *All algorithms use memory bounded by $O((\frac{1}{\epsilon} \log \frac{nm}{\delta})^{20})$, beyond the memory required for H_is.*

Then there exists an algorithm that in one pass obtains an (ϵ, δ)-approximation of $|M|$ using memory $O((\frac{1}{\epsilon} \log \frac{nm}{\delta})^{(30+k)^k})$.

PROOF. Define $g(x) = \left(\frac{1}{\epsilon}\log\frac{nm}{\delta}\right)^{(30+k)^{k-x}}$ First, we show that for any $s_1 \leq s$ there exists an algorithm $\mathfrak{B}_{s_1}(D, H_1, \ldots, H_{s_1})$ that gives an (ϵ, δ)-approximation of $|T_{s_1}(W(M, H_1, \ldots, H_{s_1}))|$ and uses memory at most $g(s_1)$.

We prove this fact by induction on s_1. For $s_1 = s$, the fact follows from the first assumption of Theorem 2.2 since $g(s) \geq \left(\frac{1}{\epsilon}\log\frac{nm}{\delta}\right)^{20}$. For $s_1 < s$, denote $\mathcal{F}'(D, H_1, \ldots, H_{s_1}) = T_{s_1}(W(\mathcal{F}(D), H_1, \ldots, H_{s_1})$. Denote $M' = \mathcal{F}'(D, H_1, \ldots, H_{s_1})$ and let H be an arbitrary hash function. By Corollary 3.4, $W(M', H) = W(\mathcal{F}'(D, H_1, \ldots, H_{s_1}), H) = W(T_{s_1}(W(\mathcal{F}(D), H_1, \ldots, H_{s_1}), H)$, i.e.,

$$W(M', H) = T_{s_1}(W(M, H_1, \ldots, H_{s_1}, H)). \qquad (1)$$

Thus, and by the second assumption of the theorem, there exists an algorithm $\mathfrak{A}_{s_1, s_1+1}$ that in one pass obtains a $(\log^k(n), \delta)$-approximation of $|W(M', H)|$ using memory less than or equal to $g(s_1 + 1)$.

Also, by Corollary 3.5 and by (1):

$$T_1(W(M', H)) = T_{s_1+1}(W(M, H_1, \ldots, H_{s_1}, H)). \qquad (2)$$

By induction, there exists an algorithm that gives an (ϵ, δ)-approximation of $|T_{s_1+1}(W(M, H_1, \ldots, H_{s'}, H))| = |T_1(W(M', H))|$ using memory $g(s_1 + 1)$.

M' is implicitly defined by a fixed function $\mathcal{F}'(D, H_1, \ldots, H_s)$. By Fact 3.7, M' entries are polynomially bounded. Thus, by (1) and (2), all assumptions of Theorem 2.1 are satisfied for M'. Therefore, there exists an algorithm that gives an ϵ-approximation of $|M'| = |T_{s_1}(W(M, H_1, \ldots, H_{s_1}))|$ using memory $g(s_1)$.

In particular, there exists an algorithm that for any H gives an ϵ-approximation of $|T_1(W(M, H))|$ using $g(1)$. Also, by the second assumption of the theorem, there exists an algorithm that gives a $\log^k(n)$-approximation of $|T_0(W(M, H))| = |W(M, H)|$. Thus, we can apply Theorem 2.1 for M and obtain an ϵ-approximation of $|M|$. The resulting memory usage will be $O(\left(\frac{1}{\epsilon}\log\frac{nm}{\delta}\right)^{(30+k)^k})$. $\square$

We use the following lemmas (the proofs can be found in the full version of our paper [11]).

LEMMA 2.3. *There exists an algorithm $\mathfrak{B}_{k-1}$ that, given a data stream D and an access to hash functions $H_1, \ldots, H_{k-1}$, in one pass obtains an ϵ-approximation of $|T_{k-1}(W(M_{Ind}, H_1, \ldots, H_{k-1}))|$ using memory $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\log\frac{nm}{\epsilon\delta})$.*

LEMMA 2.4. *There exists an algorithm $\mathfrak{A}_{s_1,s_2}$ (for any $0 \leq s_2 \leq s_1 \leq k$) that, given a data stream D and an access to hash functions $H_1, \ldots, H_{s_1}$, in one pass obtains a $\log^k n$-approximation of $|T_{s_2}(W(M_{Ind}, H_1, \ldots, H_{s_1}))|$ using memory $O(\log{(nm)}\log\frac{1}{\delta})$.*

THEOREM 2.5. ***Main Theorem*** *Let $k \geq 2$ be a constant, and let D be a stream of k-tuples from $[n]^k$. For any $0 < \epsilon < 1$, there exists an algorithm that makes a single pass over D and returns an (ϵ, δ)-approximation of the statistical distance between product and joint distribution (see Definition 1.1) using memory $O(\left(\frac{1}{\epsilon}\log(\frac{nm}{\delta})\right)^{(30+k)^k})$.*

PROOF. By Lemma 2.3 and Lemma 2.4, the algorithms required by Theorem 2.2 exist for M_{Ind}. Also, by Fact 3.7, the entries of M_{Ind} are polynomially bounded. Thus all assumptions of Theorem 2.2 are true for M_{Ind}. Applying Theorem 2.2 to M_{Ind}, we obtain the main result. $\square$

3. PROPERTIES OF TENSORS

We prove the following useful facts about Suffix-Sum and Prefix-Zero operations.

FACT 3.1. *Let M be a t-dimensional tensor and $0 \leq s \leq t$. Then*

$$W(T_s(M), H) = T_s(W(M, H_1 = \mathbf{1}, \ldots, H_s = \mathbf{1}, H)).$$

PROOF. Denote by $m_{\mathbf{w}}$ (for $\mathbf{w} \in [n]^t$) the $\mathbf{w}$-th entry of M. For any $\mathbf{i} \in [n]^{t-s}$, denote by $a_{\mathbf{i}}$ the entry of $T_s(M)$. By Definition 1.12:

$$a_i = \sum_{\mathbf{j}\in[n]^s} m_{(\mathbf{j},\mathbf{i})}.$$

Denote by $b_{\mathbf{i}}$ the entry of $W(T_s(M), H)$. By Definitions 1.12 and 1.13:

$$b_{\mathbf{i}} = H(\mathbf{i}_1)a_{\mathbf{i}} = \sum_{\mathbf{j}\in[n]^s} m_{(\mathbf{j},\mathbf{i})}H(\mathbf{i}_1).$$

Denote by $c_{\mathbf{i}}$ the $\mathbf{i}$-th entry of $T_s(W(M, H_1 = \mathbf{1}, \ldots, H_s = \mathbf{1}, H))$. By Definitions 1.12 and 1.13:

$$c_{\mathbf{i}} = \sum_{\mathbf{j}\in[n]^s} m_{(\mathbf{j},\mathbf{i})}H(\mathbf{i}_1).$$

Thus, for any $\mathbf{i}$, $b_{\mathbf{i}} = c_{\mathbf{i}}$ and the fact is correct. $\square$

FACT 3.2. *Let M be a t-dimensional tensor and let $0 \leq s < t$. Then $T_1(T_s(M)) = T_{s+1}(M)$.*

PROOF. Denote by $m_{\mathbf{w}}$ (for $\mathbf{w} \in [n]^t$) the $\mathbf{w}$-th entry of M. For $\mathbf{j} \in [n]^{t-s}$ denote $b_{\mathbf{j}}$ to be an entry of $T_s(M)$. By Definition 1.12:

$$b_{\mathbf{j}} = \sum_{\mathbf{u}\in[n]^s} m_{(\mathbf{u},\mathbf{j})}.$$

For every $\mathbf{i} \in [n]^{t-s-1}$, denote by $c_{\mathbf{i}}$ the entry of $T_1(T_s(M))$. By Definition 1.12:

$$c_{\mathbf{i}} = \sum_{l\in[n]} b_{(l,\mathbf{i})} = \sum_{l\in[n]}\sum_{\mathbf{u}\in[n]^s} m_{(\mathbf{u},(l,\mathbf{i}))} =$$

$$\sum_{l\in[n]}\sum_{\mathbf{u}\in[n]^s} m_{((\mathbf{u},l),\mathbf{i})} = \sum_{\mathbf{v}\in[n]^{s+1}} m_{(\mathbf{v},\mathbf{i})}.$$

For any $\mathbf{i} \in [n]^{t-s-1}$ denote by $a_{\mathbf{i}}$ the entry of $T_{s+1}(M)$. By Definition 1.12:

$$a_{\mathbf{i}} = \sum_{\mathbf{v}\in[n]^{s+1}} m_{(\mathbf{v},\mathbf{i})}.$$

Thus, for any $\mathbf{i}$, $a_{\mathbf{i}} = c_{\mathbf{i}}$ and the fact is correct. $\square$

FACT 3.3. *Let M be a t-dimensional tensor, let $s \leq t$ and let $H_1, \ldots, H_s$ and $G_1, \ldots, G_s$ be hash functions. Then*

$$W(M, H_1G_1, \ldots, H_sG_s) = W(W(M, H_1, \ldots, H_s), G_1, \ldots, G_s))$$

COROLLARY 3.4. *Let M be a t-dimensional tensor and let $0 \leq s < t$. Let $M' = T_s(W(M, H_1, \ldots, H_s))$. Then*

$$W(M', H) = T_s(W(M, H_1, \ldots, H_s, H)).$$

PROOF. Denote $M'' = W(M, H_1, \ldots, H_s)$. Then by Fact 3.1:

$$W(M', H) = W(T_s(M''), H) =$$

$$T_s(W(M'', G_1 = \mathbf{1}, \ldots, G_k = \mathbf{1}, H)).$$

Also by Fact 3.3:

$$W(M'', G_1, \ldots, G_s, H) =$$

$$W(W(M, H_1, \ldots, H_s, \mathbf{1}), G_1, \ldots, G_s, H) =$$

$$W(M, H_1, \ldots, H_s, H).$$

□

COROLLARY 3.5. *Let M be a t-dimensional tensor and let $0 \leq s < t$. Let $M' = T_s(W(M, H_1, \ldots, H_s))$. Then*

$$T_1(M', H)) = T_{s+1}(W(M, H_1, \ldots, H_s, H)).$$

PROOF. By Fact 3.2 and Corollary 3.4:

$$T_{s+1}(W(M, H_1, \ldots, H_s, H)) =$$

$$T_1(T_s(W(M, H_1, \ldots, H_s, H))) = T_1(W(M', H)).$$

□

FACT 3.6. *Let M be an arbitrary s-dimensional tensor, let M_l be $(1 - \epsilon/2)$-significant hyperplane of M, $M_l = Hyperplane(M, l)$, and let $M' = T_1(M)$. Then $|M'|$ is an ϵ-approximation of $|M_l|$.*

PROOF. We have

$$|M'| = \sum_{\mathbf{i} \in [n]^{s-1}} |\sum_{j \in [n]} m_{(j,\mathbf{i})}| \leq$$

$$\sum_{\mathbf{i} \in [n]^{s-1}} \sum_{j \in [n]} |m_{(j,\mathbf{i})}| = |M| \leq \frac{1}{1 - \epsilon/2}|M_l| \leq (1+\epsilon)|M_l|.$$

On the other hand,

$$|M'| = \sum_{\mathbf{i} \in [n]^{s-1}} |\sum_{j \in [n]} m_{(j,\mathbf{i})}| \geq$$

$$\sum_{\mathbf{i} \in [n]^{s-1}} (|m_{(l,\mathbf{i})}| - \sum_{j \in [n], j \neq l} |m_{(j,\mathbf{i})}|) = |M_l| - (|M| - |M_l|) =$$

$$\geq (2 - \frac{1}{1 - \epsilon/2})|M_l| \geq (1 - \epsilon)|M_l|.$$

□

FACT 3.7.

1. *Let M be a s-dimensional tensor with polynomially bounded (in n and m) entries for $s \leq k$. Let M' be a tensor obtained from M by an arbitrary composition of Prefix-Zero, AbsoluteVector, Hyperplane and Suffix-Sum operators. Then the entries of M' are polynomially bounded.*

2. *All entries of M_{Ind} are integers with absolute values bounded by $2m^k$ and thus claim 1 is true for M_{Ind}.*

PROOF. The first claim follows from the fact that the entries of M' are sums of disjoint subsets of M and that the number of entries in M is bounded by n^k. The second claim follows from Definition 1.7. □

4. CERTIFYING TOURNAMENTS

Recall that certifying tournaments allow us to distinguish the case when there exists an extremely significant hyperplane and where all hyper-planes are "light" in the implicit tensors. Furthermore, in case there is heavy hyperplane, certifying tournaments return a good approximation of the weight of the heavy hyperplane[5]. The algorithm is as follows:

ALGORITHM 4.1. *Tensor Tournament$(D, \mathcal{H}, H, \epsilon)$*

```
1. Repeat in parallel $O(\frac{\log \frac{1}{\delta}}{p})$ times where $p = 1 - \sqrt{1 - \epsilon/2}$.
    (a) Generate 2-wise independent random hash function $Z$ from $[n]$ to $\{0,1\}$ such that $Z(i) = 0$ w.p. 0.5. Denote $Z_1 = HZ$, $Z_0 = H(1 - Z)$.
    (b) Compute in a single pass over $D$ for $i = 0, 1$: $\mathfrak{t}_i = \mathfrak{A}(D, \mathcal{H}, Z_i, \epsilon, \delta')$, where $\delta' = \frac{p\epsilon}{4 \log \frac{1}{\delta}}$.
    (c) Simultaneously (in the same pass), compute $\mathfrak{l}_i = \mathfrak{B}(D, \mathcal{H}, Z_i, \delta')$.
    (d) Put $\mathfrak{u}_i = \max\{\frac{\mathfrak{l}_i}{\beta}, \mathfrak{t}_i, 0\}$,   $i = 0, 1$.
    (e) Define $\lambda' = (1 + \epsilon)\lambda$, where $\lambda$ is the constant from Fact 4.4, $\lambda = (1 + \frac{2(1-\epsilon)^{1/4}}{1-(1-\epsilon)^{1/4}})$.
    (f) Compute
        $$U' = \begin{cases} \mathfrak{u}_1, & \text{if } \mathfrak{u}_1 \geq \lambda'\beta^2\mathfrak{u}_0, \\ \mathfrak{u}_0, & \text{if } \mathfrak{u}_0 \geq \lambda'\beta^2\mathfrak{u}_1, \\ 0, & \text{otherwise.} \end{cases}$$
2. Output $U$ to be the minimum of all $U'$s.
```

DEFINITION 4.2. *Let $\mathcal{F}$ be a fixed function that defines implicit vectors, given a data stream and a fixed randomness and denote $V = \mathcal{F}(D, \mathcal{H})$ as a vector with entries v_i. For $\alpha > 0.5$, an α-*ThresholdMax* algorithm for restricted $\mathcal{F}$ is an algorithm that receives as an input a data stream D and an access to a randomness $\mathcal{H}$ and a random function $H : [n] \mapsto \{0,1\}$, and in one pass over D returns $U \geq 0$ such that w.p. at least $1 - \delta$:*

1. *If $U > 0$ then U is an ϵ-approximation of $|v_i|$ for some i with $H(i) = 1$.*

2. *If [6] $|VH| > 0$ and there exists i such that $H(i) = 1$ and $|v_i| \geq (1 - \alpha)|VH|$ then U is an ϵ-approximation of $|v_i|$.*

THEOREM 4.3. *Let H be a fixed hash function defined as above and let $\epsilon \leq 0.1$. Let M be a s-dimensional tensor implicitly defined by a fixed function $\mathcal{F}$, stream D and randomness $\mathcal{H}$, $M = \mathcal{F}(D, \mathcal{H})$. Assume that there exist two algorithms:*

[5]It is important to note that despite superficial similarities, this is different from *Hybrid-Major* algorithm in [12] since here we reduce $(\log n)$-approximation of L_1 norm to ϵ-approximation, while in *Hybrid-Major* we approximate all increasing functions with high accuracy using L_2 norm.

[6]Here and henceforth we denote by VH a vector with entries $v_iH(i)$, $i \in [n]$.

- *An algorithm $\mathfrak{A}(D, \mathcal{H}, H, \delta)$ that in one pass obtains (β, δ)-approximation of $|W(M, H)|$ using memory $\mu_1(n, m, \epsilon, \delta)$;*

- *An algorithm $\mathfrak{B}(D, \mathcal{H}, H, \epsilon, \delta)$ that in one pass over D obtains an (ϵ, δ)-approximation of $|T_1(W(M, H))|$ using memory $\mu_2(n, m, \epsilon, \delta)$;*

and let $\alpha = \frac{\epsilon}{64\beta^2}$. Then, the Algorithm $TensorTournament(D, \mathcal{H}, H, \epsilon)$ is an α-ThresholdMax algorithm for restricted $\mathcal{F}'$ (see Definition 4.2), where $\mathcal{F}'(D, \mathcal{H}) = AbsoluteVector(\mathcal{F}(D, \mathcal{H}))$. The algorithm makes a single pass over D and uses memory $O(\frac{1}{\epsilon} \log \frac{1}{\delta}(\mu_1(n, m, \epsilon/3, \delta\epsilon/\log(1/\delta)) + \mu_2(n, m, \epsilon/3, \delta\epsilon/\log(1/\delta)) + \log nm)$.

Proof.
Denote $M^t = W(M, Z_t)$ for $t = 0, 1$. Let $M_i = Hyperplane(M, i)$ for $i \in [n]$ and let V' to be a vector with elements $|M_i|$. By Definition 1.11, $V' = \mathcal{F}'(D, \mathcal{H})$. Further, let V be a vector with entries $v_i = |M_i|H(i)$. We prove that the algorithm satisfies two conditions of Definition 4.2 for the ThresholdMax algorithm for V and H.

Proof of the first condition of Definition 4.2
We prove the following stronger statements which imply the first condition of Definition 4.2:

I. If there is no $(1 - \epsilon)$-significant entry v_l then, w.p. at least $1 - \frac{\delta}{3}$, $U = 0$.

II. If $|V| > 0$ and there is a $(1 - \epsilon)$-significant entry v_l then, w.p. at least $1 - \frac{\delta}{3}$, either $U = 0$ or U is a 3ϵ-approximation of $|v_l|$.

Proof of statement I
By definitions of $\mathfrak{B}, \mathfrak{A}$, we have w.p. at least $1 - 8\delta'$ for $t = 0, 1$: $\mathfrak{u}_t \geq \frac{\mathfrak{l}_t}{\beta} \geq \frac{|M^t|}{\beta^2}$; and $\mathfrak{t}_t \leq (1 + \epsilon)|T_1(M^t, H)| \leq (1 + \epsilon)|M^t|$; and $\frac{\mathfrak{l}_t}{\beta} \leq |M^t|$. Thus,

$$\frac{|M^t|}{\beta^2} \leq \mathfrak{u}_t \leq (1 + \epsilon)|M^t|. \tag{3}$$

Following the terminology of Fact 4.4, we define $X = |M^1|$ and $Y = |M^0|$. We have the following relations:

$$|V| = \sum_i v_i = \sum_i H(i)|M_i| =$$

$$\sum_{i \in [n]} H(i) \sum_{\mathbf{j}' \in [n]^{s-1}} |m_{(i, \mathbf{j}')}| = |W(M, H)|,$$

$$X = |M^1| = \sum_{\mathbf{j} \in [n]^s} Z(\mathbf{j}_1)H(\mathbf{j}_1)|m_{\mathbf{j}}| =$$

$$\sum_{i \in [n]} Z(i)H(i) \sum_{\mathbf{j}' \in [n]^{s-1}} |m_{(i, \mathbf{j}')}| =$$

$$\sum_i Z(i)H(i)|M_i| = \sum_i Z(i)v_i,$$

and similarly

$$\begin{aligned} Y = |M^0| = \sum_i (1 - Z(i))H(i)|M_i| = \\ = |V| - X = |V| - |M^1|. \end{aligned} \tag{4}$$

By statement **I**, for all i, $v_i < (1 - \epsilon)|V|$. Thus we can apply Fact 4.4. We have:

$$P((|M^0| \geq \lambda|M^1|) \cup (|M^1| \geq \lambda|M^0|)) =$$

$$P((X \geq \lambda Y) \cup (Y \geq \lambda X)) \leq \sqrt{1 - \epsilon}.$$

Let Υ be the event $(\mathfrak{u}_0 \geq \lambda'\beta^2\mathfrak{u}_1) \cup (\mathfrak{u}_1 \geq \lambda'\beta^2\mathfrak{u}_0)$. Let Φ be the event that $\frac{|M^t|}{\beta^2} \leq \mathfrak{u}_t \leq (1 + \epsilon)|M^t|$ for both values of t. We have $P(\Upsilon) \leq P(\Upsilon, \Phi) + P(\bar{\Phi})$. By (3), we have $P(\bar{\Phi}) \leq 8\delta'$. Also, events $\mathfrak{u}_0 \geq \lambda'\beta^2\mathfrak{u}_1$ and Φ imply that $|M^0| \geq \lambda|M^1|$; indeed:

$$|M^0| \geq \frac{\mathfrak{u}_0}{(1 + \epsilon)} \geq \frac{\lambda'}{1 + \epsilon}\beta^2\mathfrak{u}_1 \geq \lambda|M^1|.$$

Thus we have

$$P(\Upsilon, \Phi) \leq P((|M^0| \geq \lambda|M^1|) \cup (|M^1| \geq \lambda|M^0|)) \leq \sqrt{1 - \epsilon}.$$

We summarize that if no $(1 - \epsilon)$-significant v_i exists, then

$$P(U' \neq 0) \leq P(\Upsilon) \leq \sqrt{1 - \epsilon} + O(\delta') \leq \sqrt{1 - \epsilon/2}.$$

Recall that the number of repetitions is $O(\frac{1}{p} \log 1/\delta)$, where $p = 1 - \sqrt{1 - \epsilon/2}$. Thus $P(U \neq 0) \leq (1 - p)^{\frac{1}{p} \log \frac{3}{\delta}} \leq \frac{\delta}{3}$.

Proof of statement II
Let v_l be a $(1 - \epsilon)$-significant entry of V. Assume, w.l.o.g., that for one execution of the main cycle of the $Tournament$ algorithm, $Z(l) = 0$. Statement **II** implies $|V| > 0$ which implies $v_l = |M_l|H(l) > 0$ which implies $(1 - Z(l))H(l) = 1$. Thus, $|Hyperplane(M^0, l)| = |Hyperplane(W(M, (\mathbf{1} - Z)H), l)| = |M_l| = v_l$. Therefore by (4), $|Hyperplane(M^0, l)| = v_l \geq (1 - \epsilon)|V| \geq (1 - \epsilon)|M^0|$, i.e., the l-th hyperplane of M^0 is $(1 - \epsilon)$-significant. By Fact 3.6, $|T(M^0)|$ is an 2ϵ-approximation of $|M_l|$. By the assumptions of the theorem, $\mathfrak{B}$ returns an ϵ-approximation of $|T(M^0)|$. Thus, $\mathfrak{t}_0$ is a 3ϵ-approximation of $|M_l|$, w.p. at least $1 - \delta'$, in which case

$$\mathfrak{u}_0 \geq \mathfrak{t}_0 \geq (1 - 3\epsilon)|M_l|.$$

Also, by the assumption of Theorem 4.3, w.p. at least $1 - \delta'$, we have $\frac{\mathfrak{l}_0}{\beta} \leq |M^0|$. Thus

$$\mathfrak{u}_0 = \max\{\frac{\mathfrak{l}_0}{\beta}, \mathfrak{t}_0, 0\} \leq \max\{|M^0|, (1 + 3\epsilon)|M_l|\} \leq (1 + 3\epsilon)|M_l|.$$

On the other hand, w.p. at least $1 - 2\delta'$

$$\mathfrak{u}_1 = \max\{\frac{\mathfrak{l}_1}{\beta}, \mathfrak{t}_1, 0\} \leq \max\{|M^1|, (1 + \epsilon)|M^1|\} = (1 + \epsilon)|M^1|.$$

But since $Z_s(l) = 0$ we have by (4):

$$|M^1| \leq |V| - |Hyperplane(M^0, l)| = |V| - v_l \leq \frac{\epsilon}{1 - \epsilon}|M_l|.$$

Combining all of the above computations, we conclude that w.p. at least $1 - 4\delta'$ (for sufficiently small ϵ, e.g., $\epsilon \leq 0.1$):

$$\mathfrak{u}_1 \leq (1 + \epsilon)|M_1| \leq \frac{\epsilon(1 + \epsilon)}{1 - \epsilon}|M_l| \leq \frac{\epsilon(1 + \epsilon)}{(1 - \epsilon)(1 - 3\epsilon)}\mathfrak{u}_0 < \lambda'\mathfrak{u}_0.$$

Thus, U' is equal to either 0 or $\mathfrak{u}_0$ w.p. at least $1-4\delta'$. Recall simultaneously $\mathfrak{u}_0$ is a 3ϵ-approximation of $|M_l| = v_l$. The same inequality is true if $Z(l) = 1$. By union bound, w.p. at least $1 - \Omega(\frac{\log \frac{1}{\delta}}{p}\delta') = 1 - \Omega(\delta)$, U is either 0 or a 3ϵ-approximation of v_l.

Proof of the second condition of Definition 4.2

Finally, consider the case when v_l is a $(1-\alpha)$-significant entry of V. Consider the case when $Z(l) = 0$. Repeating the arguments from the proof of statement **II**, we have, w.p. at least $1-4\delta'$, $\mathfrak{u}_0$ is a 3ϵ-approximation of v_l and

$$\mathfrak{u}_1 \leq (1+\epsilon)|M^1| \leq (1+\epsilon)\frac{\alpha}{(1-\alpha)}v_l \leq 4\alpha v_l.$$

Therefore,

$$\mathfrak{u}_0 \geq (1-3\epsilon)v_l \geq \frac{(1-3\epsilon)}{4\alpha}\mathfrak{u}_1 \geq \lambda'\beta^2\mathfrak{u}_1.$$

Thus, w.p. $1-4\delta'$, $U' = \mathfrak{u}_0 = (1 \pm 3\epsilon)v_l$. The same is true when $Z(l) = 1$. Thus, U is a 3ϵ-approximation of v_l w.p. at least $1 - \Omega(\delta)$.

Conclusion and memory analysis

Since both conditions of Definition 4.2 are met (substituting ϵ with $\epsilon/3$), we conclude that *TensorTournament* is an α-ThresholdMax algorithm for restricted $\mathcal{F}'$. Let us count the memory needed for a single iteration of the main cycle of the algorithm. To generate pairwise independent Z, we need $O(\log n)$ bits. In addition, we need $\mu_1 + \mu_2$ for the algorithms $\mathfrak{B}$ and $\mathfrak{A}$ and $O(\log nm)$ bits to keep the auxiliary variables. Thus, in total we need memory $O(\frac{1}{\epsilon}\log\frac{1}{\delta}(\mu_1(n,m,\epsilon/3,\delta\epsilon/\log(1/\delta)) + \mu_2(n,m,\epsilon/3,\delta\epsilon/\log(1/\delta)) + \log nm)$. Recall that we do not count memory required to store $\mathcal{H}$ and H. $\square$

FACT 4.4. *Let V be a n-dimensional vector with non-negative entries $v_i \geq 0, i \in [n]$. Let Z be 2-wise independent random hash functions from $[n]$ to $\{0,1\}$, such that $P(Z(i) = 1) = 0.5$. Let $X = \sum_i v_i Z(i)$, and $Y = L_1(V) - X$. If there exists $\epsilon > 0$ such that for all i $v_i < (1-\epsilon)L_1(V)$, then for $\lambda = \lambda(\epsilon) \geq 1 + \frac{2(1-\epsilon)^{1/4}}{1-(1-\epsilon)^{1/4}}$ we have*

$$P((X \geq \lambda Y) \cup (Y \geq \lambda X)) \leq \sqrt{1-\epsilon}.$$

5. APPROXIMATING L_1 NORMS OF IMPLICIT VECTORS

DEFINITION 5.1. *Let V with $v_i \geq 0$ be a vector from R^n. A set $\mathcal{U}$ of positive numbers is an* ϵ-cover *of V if:*

1. *All elements of $\mathcal{U}$ are ϵ-approximations of distinct and positive coordinates from V (i.e., there is a one-to-one mapping ρ from the set $\mathcal{U}$ to a subset $S' \subseteq [n]$ such that for all $U \in \mathcal{U}$, U is an ϵ-approximation of $v_{\rho(U)}$.)*
2. *$\mathcal{U}$ contains ϵ-approximations of all ϵ-significant elements of V (i.e., for all v_i such that $v_i \geq \epsilon|V|$, it is true that $i \in S'$.)*

The size *of the cover is $|\mathcal{U}|$.*

DEFINITION 5.2. *Let $\mathcal{F}$ be a fixed function that implicitly defines vectors, given a data stream D and a fixed randomness $\mathcal{H}$. Denote $V = \mathcal{F}(D,\mathcal{H})$. A* Cover algorithm *for restricted $\mathcal{F}$ is an algorithm that receives as an input a data stream D, an access to a randomness $\mathcal{H}$ and a random function $H : [n] \mapsto \{0,1\}$ and an ϵ and δ. The algorithm makes a single pass over D and w.p. at least $1-\delta$, returns an ϵ-cover of vector with entries $v_i H(i)$.*

LEMMA 5.3. *Let $\mathcal{F}$ be a fixed function that implicitly defines vectors, given a data stream D and a fixed randomness $\mathcal{H}$. An existence of α-ThresholdMax algorithm for restricted $\mathcal{F}$ that uses memory $\mu(n,m,\epsilon,\delta)$ implies an existence of a Cover algorithm for restricted $\mathcal{F}$ for any ϵ. The Cover algorithm uses memory $O(\frac{1}{\epsilon^2\delta\alpha}(\mu(n,m,\epsilon,\delta^2\epsilon^2\alpha) + \log nm))$.*

PROOF. Denote by $\mathfrak{L}_\alpha(D,\mathcal{H},H,\epsilon,\delta)$ the existing α-ThresholdMax algorithm for restricted $\mathcal{F}$.

Using $\mathfrak{L}_\alpha$ we construct the following algorithm. Let $\epsilon' = \epsilon^2\delta/3$ and $\varrho = \lceil\frac{1}{\epsilon'\alpha}\rceil$. Let G be a pairwise independent random hash function from $[n]$ to $[\varrho]$ that is independent of $\mathcal{H}$ and H. For $s \in [\varrho]$, define function F_s as $F_s(i) = \mathbf{1}_{G(i)=s}$ and execute, in parallel for all s, $\mathfrak{L}_\alpha(D,\mathcal{H},HF_s,\epsilon,\delta/\varrho)$. Let U_s be the output of s-th ran of $\mathfrak{L}_\alpha$. The output of our new algorithm is a set of all strictly positive U_s. We show below that the output is indeed ϵ-cover of V with probability at least $1-\delta$.

Let $V = \mathcal{F}(D,\mathcal{H})$ be a vector with entries v_i and let V_s be a vector with entries $v_{s,i} = v(i)F_s(i)$. By the union bounds and by the definition of α-ThresholdMax algorithm, w.p. at least $1-\delta$, every positive U_s is an ϵ approximation of $|v_{i_s}|$ for some i_s with $H(i_s)F_s(i_s) = 1$. But this implies that U_s is an approximation of $|v_i|$ with $H(v_i) = 1$. Since G splits $[n]$ into disjoint subsets, the output of our algorithm corresponds to ϵ-approximations of absolute values of a set of distinct entries of V. I.e., the first condition of ϵ-cover is correct.

To show that the second condition is true as well, let S_ϵ be set of all is such that $|v_iH(i)| \geq \epsilon|VH| > 0$. Consider a fixed $i \in S_\epsilon$. Let $X_i = |VHF_{G(i)}| - |v_i| = \sum_{j\neq i}|v_j|H(j)F_{G(i)}(j) \geq 0$. By pairwise independence of G: $E(X_i) = \sum_{j\neq i}|v_j|H(j)P(G(j) = G(i)) \leq \frac{|VH|}{\varrho}$.

Let Ψ_i be the event that $X_i > \frac{\epsilon}{\varrho\epsilon'}|VH|$; by Markov inequality $P(\Psi_l) \leq \frac{\epsilon'}{\epsilon}$. Note that if Ψ_i does not happen, then $|VHF_{G(i)}| - |v_i| \leq \frac{\epsilon}{\varrho\epsilon'}|VH| \leq \frac{1}{\varrho\epsilon'}|v_i| \leq \alpha|v_i|$, in which case $|v_i| \geq (1-\alpha)|VHF_{G(i)}|$. Let Γ_l be the event that $U_{G(i)}$ is not an ϵ-approximation of $|v_i|$. By the properties of algorithm $\mathfrak{L}_\alpha$, $P(\Gamma_i|\bar{\Psi}_i) \leq \frac{\delta}{\varrho}$. Thus

$$P(\Gamma_i) \leq P(\Gamma_i|\bar{\Psi}_i) + P(\Psi_i) \leq \frac{\delta}{\varrho} + \frac{\epsilon'}{\epsilon}.$$

Finally, let $\Phi_{i,j}$ be the event where there is a collision between i and j. By pairwise independence of G, $P(\Phi_{i,j}) = \frac{1}{\varrho}$, and thus the probability of collisions for ϵ-significant entries is bounded by $\frac{1}{\epsilon^2\varrho}$. Thus, the probability that the output of the algorithm does not meet the second condition of ϵ-cover is bounded by $P((\cup_{i\in S_\epsilon}\Gamma_i)\cup(\cup_{i,j\in S_\epsilon}\Phi_{i,j})) \leq \frac{\delta}{\varrho\epsilon} + \frac{\epsilon'}{\epsilon^2} + \frac{1}{\varrho\epsilon^2} \leq \delta$. $\square$

DEFINITION 5.4. *Let $\mathcal{F}$ be a fixed function that defines an implicit vector $V = \mathcal{F}(D,\mathcal{H})$, given D and a randomness $\mathcal{H}$, as in Definition 1.6. An algorithm that receives as an input a data stream D and an access to a randomness $\mathcal{H}$ and in one pass over D returns an (ϵ,δ)-approximation of $|\mathcal{F}(D,\mathcal{H})|$ is called an* (ϵ,δ)-approximation algorithm *for $L_1(\mathcal{F})$.*

Finally, we are ready to state our main lemma, which is a strict generalization of Indyk and Woodruff [22] (for the proof see our full version [11]):

LEMMA 5.5. *Let $\mathcal{F}$ be a fixed function that defines an implicit vector $V = \mathcal{F}(D, \mathcal{H})$, given D and a randomness $\mathcal{H}$. Assume that V has non-negative entries bounded by poly(n, m). Then the existence of Cover algorithm $\mathfrak{Q}(D, \mathcal{H}, H, \epsilon, \delta)$ for restricted $\mathcal{F}$ (see Definition 5.2) that uses memory $\mu(n, m, \epsilon, \delta)$ implies an existence of an $(\epsilon, 2/3)$-approximation algorithm for $L_1(\mathcal{F})$ (Definition 5.4) that uses memory*

$$O\left(\frac{1}{\epsilon}\log(n)\mu(n,m,\frac{\epsilon^7}{\log^3(nm)},\frac{\epsilon}{\log(nm)})+\frac{1}{\epsilon^2}\log^2(nm)\right).$$

Acknowledgments

We thank Piotr Indyk for a helpful discussion.

6. REFERENCES

[1] N. Alon, A. Andoni, T. Kaufman, K. Matulef, R. Rubinfeld, N. Xie, "Testing k-wise and almost k-wise independence," *Proceedings of the ACM symposium on Theory of computing*, 2007, pp. 496-Ű505.

[2] N. Alon, O. Goldreich, Y. Mansour, "Almost k-wise independence versus k-wise independence," *Inform. Process. Lett.*, 88:107Ű110, 2003.

[3] N. Alon, Y. Matias, M.Szegedy, "The space complexity of approximating the frequency moments". *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp.20-29, 1996.

[4] T. Batu, L. Fortnow, E. Fischer, R. Kumar, R. Rubinfeld, P. White, "Testing random variables for independence and identity," *FOCS*, 2001, pp. 442Ű-451.

[5] T. Batu , L. Fortnow , R. Rubinfeld , W. D. Smith , P. White, "Testing that distributions are close," *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, p.259, 2000.

[6] T. Batu, R. Kumar, R. Rubinfeld, "Sublinear algorithms for testing monotone and unimodal distributions," *In Proc. 36th Annual ACM Symposium on the Theory of Computing*, pp. 381-Ű390, 2004.

[7] L. Bhuvanagiri, S. Ganguly, D. Kesh, C. Saha, "Simpler algorithm for estimating frequency moments of data streams," *in ACM-SIAM Symposium on Discrete Algorithms*, 2006, pp. 708Ű-713.

[8] V. Braverman, K.-M. Chung, Z. Liu, M. Mitzenmacher, R. Ostrovsky, "AMS Without **4**-Wise Independence on Product Domains," STACS 2010.

[9] V. Braverman, R. Ostrovsky, "Effective Computations on Sliding Windows," SIAM J. Comput. Volume 39, Issue 6, pp. 2113-2131 (2010).

[10] V. Braverman, R. Ostrovsky, "Smooth Histograms for Sliding Windows," In Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (October 21 - 23, 2007).

[11] V. Braverman, R. Ostrovsky, "Measuring Independence of Datasets," http://arxiv.org/abs/0903.0034.

[12] V.Braverman, R.Ostrovsky, "Zero-One Frequency Laws," STOC 2010.

[13] V. Braverman, R. Ostrovsky, C. Zaniolo, "Optimal sampling from sliding windows," In Proceedings of the Twenty-Eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (Providence, Rhode Island, USA, June 29 - July 01, 2009).

[14] M. Charikar, K. Chen, M. Farach-Colton, "Finding frequent items in data streams," *Theoretical Computer Science*, v.312 n.1, p.3-15, 2004.

[15] G. Cormode and S. Muthukrishnan. "An Improved Data Stream Summary: The Count-Min Sketch and its Applications". J. Algorithms, 55(1):58Ű75, April 2005.

[16] G. Cormode, S. Muthukrishnan, "What's New: Finding Significant Differences in Network Data Streams," INFOCOM 2004.

[17] J. Feigenbaum, S. Kannan, M. Strauss, M. Viswanathan, "An Approximate L1-Difference Algorithm for Massive Data Streams," *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, p.501, 1999.

[18] S. Ganguly, G. Cormode, "On Estimating Frequency Moments of Data Streams," APPROX-RANDOM 2007.

[19] I. F. Ilyas, V. Markl , P. Haas, P. Brown, A. Aboulnaga, "CORDS: automatic discovery of correlations and soft functional dependencies," *SIGMOD 2004*.

[20] P. Indyk, "Stable distributions, pseudorandom generators, embeddings, and data stream computation," *J. ACM*, 53 (2006), pp. 307-Ű323.

[21] P. Indyk, A. McGregor, "Declaring Independence via the Sketching of Sketches," *ACM-SIAM Symposium on Discrete Algorithms*, 2008.

[22] P. Indyk, D. P. Woodruff, "Optimal approximations of the frequency moments of data streams," *in ACMSymposium on Theory of Computing*, 2005, pp. 202Ű-208.

[23] R. Kimball, Joe Caserta, "The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleanin," John Wiley & Sons, 2004.

[24] P. Li, "Estimators and tail bounds for dimension reduction in $l_\alpha, (0 \leq \alpha \leq 2)$ using stable random projections," SODA 2008.

[25] S. Muthukrishnan, "Data Streams: Algorithms And Applications," *Foundations and Trends in Theoretical Computer Science*, Volume 1, Issue 2.

[26] V. Poosala, Y. E. Ioannidis, "Selectivity Estimation Without the Attribute Value Independence Assumption," *Proceedings of the 23rd International Conference on Very Large Data Bases*, pp.486-495, 1997.

[27] R. Rubinfeld, R. A. Servedio, "Testing monotone high-dimensional distributions," *In Proc. 37th Annual ACM Symposium on the Theory of Computing*, pp. 147-156, 2005.

[28] A. Sahai, S. Vadhan, "Manipulating statistical difference," DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 43., pp. 251-270.