Summer 7-15-2019

New Algorithms for Large Datasets and Distributions

Sutanu Gayen
University of Nebraska - Lincoln, sutanugayen@gmail.com

NEW ALGORITHMS FOR LARGE DATASETS AND DISTRIBUTIONS

by

Sutanu Gayen

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Doctor of Philosophy

Major: Computer Science

Under the Supervision of Professor Vinodchandran N. Variyam

Lincoln, Nebraska

May, 2019

NEW ALGORITHMS FOR LARGE DATASETS AND DISTRIBUTIONS

Sutanu Gayen, Ph.D.

University of Nebraska, 2019

Adviser: Vinodchandran N. Variyam

In this dissertation, we make progress on certain algorithmic problems broadly over two computational models: the streaming model for large datasets and the distribution testing model for large probability distributions.

First we consider the streaming model, where a large sequence of data items arrives one by one. The computer needs to make one pass over this sequence, processing every item quickly, in a limited space. In Chapter 2 motivated by a bioinformatics application, we consider the problem of estimating the number of low-frequency items in a stream, which has received only a limited theoretical work so far. We give an efficient streaming algorithm for this problem and show its complexity is almost optimal.

In Chapter 3 we consider a distributed variation of the streaming model, where each item of the data sequence arrives arbitrarily to one among a set of computers, who together need to compute certain functions over the entire stream. In such scenarios combining the data at a computer is infeasible due to large communication overhead. We give the first algorithm for k-median clustering in this model. Moreover, we give new algorithms for frequency moments and clustering functions in the distributed sliding window model, where the computation is limited to the most recent W items, as the items arrive in the stream.

In Chapter 5, in our identity testing problem, given two distributions P (unknown, only samples are obtained) and Q (known) over a common sample space of exponential

size, we need to distinguish $P = Q$ (output 'yes') versus P is far from Q (output 'no'). This problem requires an exponential number of samples. To circumvent this lower bound, this problem was recently studied with certain structural assumptions. In particular, optimally efficient testers were given assuming P and Q are product distributions. For such product distributions, we give the first tolerant testers, which not only output yes when $P = Q$ but also when P is close to Q, in Chapter 5. Likewise, we study the tolerant closeness testing problem for such product distributions, where Q too is accessed only by samples.

ACKNOWLEDGMENTS

"...in mathematics you don't understand things, you just get used to them."

(John von Neumann)

It has been a matter of great pleasure and fortune to be advised by Dr. Variyam. He gave me the freedom to choose problems which made it easier to engage in research. I received tremendous support from him through both favorable and testing conditions and have tried to imbibe his qualities and viewpoints to the fullest. All this have made me better for sure.

I thank my supervisory committee which includes Dr. Stephen Scott, Dr. Lisong Xu and Dr. Jamie Radcliffe for their valuable comments. I thank Dr. Pavankumar Aduri for meetings and discussions on multiple occasions and for hosting me. I thank the staffs of UNL who were helpful during times of need. I thank the colleagues in the theory and machine learning lab, in CSE Department, the cool people of UNL and the city of Lincoln, who maintained an environment conducive for work and life. My friends in Lincoln including Suraj, Pinaki, Abhishek, Sandrayee, Debalin, Bhaskar, Kunal, Kaushik, Sairam, kept me going through the stressful times and I wish them all the best. I thank Dr. Arnab Bhattacharyya for hosting me for a research visit to IISc.

I gratefully recall Dr. Surender Baswana for making Computer Science interesting to me for the first time, Dr. Anil Seth for his cool courses, and Dr. Raghunath Tewari for his guidance. I offer respectful salutes to the great teachers of RKMV Purulia and Narendrapur schools, including Bhargab maharaj and Ranjit da in particular. I offer my salutes to Debashish Mitra (acharyadeb) for teaching me basics of maths from childhood. Dr. Dipak Das has guided me like an elder brother since school days.

I sincerely believe if there is any positivity and good in me, it is all due to the hard works, sacrifices, love and blessings of my mother Shilla Manna Gayen and my father Tapan Kumar Gayen and. I also recall my dida, dadu, masi, boromasi, meso, bone, ranga dida, sanu bhai, who had been great supports to us since childhood. Finally, I sincerely believe if any works have been done, it was done by thakur's power and will and I humbly offer them to his lotus feet.

"Lead us from the unreal to the real. Lead us from darkness unto light. Lead us from death to immortality. Peace, peace, peace."

(Yajurveda)

PREFACE

- Some of the results presented in Chapter 2 has been published in [10].
- Some of the results presented in Chapter 3 has been published in [36] and [37].
- The results presented in Chapter 5 are to be included in a manuscript [11] that is currently in preparation for publication.

Table of Contents

Chapter 1

Introduction to Streaming Algorithms

Organizations have taken a quantum leap in their ability to generate data due to advances in computing facilities such as storage, processor, and communication channels. It is estimated that 1.7 MB of data will be created per person on earth per second by 2020 [34]. We refer the reader to an interesting study conducted by domo.com [34] about the gigantic scale of data that was generated per minute in 2018 by a few major commercial organizations including Google, Netflix, Skype, and Twitter. There are also massive static datasets coming out of crucial scientific studies [20, 56]. All these data are stored under the assumption that they can be 'mined' or analyzed for the benefit of the stakeholder.

This phenomenal ability of the above organizations to generate massive amounts of data have posed a challenge to traditional ways of looking at the computation involved. Traditional models of computing assumes data resides in one place in the physical memory of the system and any desired location of it can be accessed efficiently. Traditionally, a computation is considered efficient if the time and space (physical memory) taken by it are small polynomial functions in the size of the input. These assumptions are no longer valid for modern datasets for reasons such as:

- The data may be distributed across several locations which could not be easily

combined in a single place

- Any desired location of the data might not be accessed efficiently since it may be residing in a secondary storage, where access time is orders of magnitudes higher than the seek time

- Data might be coming through a communication channel, and it must be processed on the fly

- Data may be so large that even linear space and linear time computations are too costly, and any saving in them would enable us to analyze a larger amount of data in proportion.

These challenges posed by large datasets and their analysis is referred to by the umbrella term: big data. In an attempt to address the above concerns, researchers have proposed new models of computation for big data. Among them, the streaming model has received considerable attention.

1.1 Streaming algorithms and their history

In the first part of this thesis, we study certain algorithmic problems over the streaming model of computation. In this model we can make only a single pass over the dataset: we cannot revisit an item unless it is stored in the memory. The time to process an item and the space used by the algorithm need to be as small as possible. This model is schematically represented in Figure 1.1.

In a classic study in 1980, Munro and Patterson [55] studied the searching and sorting in data stored on a one-way read-only tape with constrained memory. In 1985, Flajolet and Martin [35] gave a fast and small space randomized algorithm for counting the distinct items in a dataset, by making a single pass over it. These

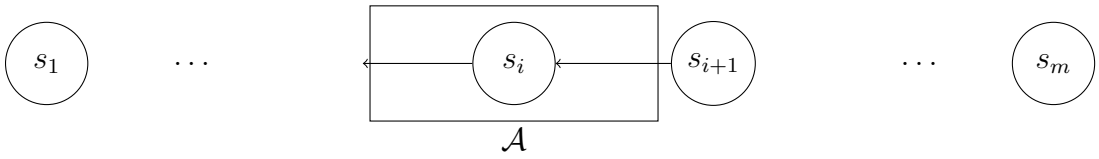

Figure 1.1: Streaming algorithm $\mathcal{A}$ processing the stream $D = \langle s_1, \ldots, s_i, s_{i+1}, \ldots, s_m \rangle$. $\mathcal{A}$ sees each item once in the above sequence and must process it fast within a small space.

eventually led to the formalization of the streaming model by the popular work of Alon, Matias, and Szegedy [2]. Since then, a flurry of important research works appeared in this model from theory, database, machine learning, and networking researchers. We refer the reader to the textbooks [49, 12] and the references therein for a comprehensive coverage of streaming algorithms and big data algorithms in general.

1.2 Summary of our results on streaming algorithms

In Chapter 2 we consider a problem motivated by bioinformatics. In this problem, we are interested to estimate the number of low frequency items in a stream. We give an efficient streaming algorithm for this problem and also show its space complexity is almost optimal.

In Chapter 3, we consider a variation of the streaming model, where the dataset is distributed across several locations. In this variation, simply combining the data at a single place would cost too much communication overhead. The primary goal in this case is to use as little communication as possible. We give the first algorithm for approximate k-median clustering in this model. We also look at the sliding window variation of the distributed streaming model, where the computation is limited to the set of most recent W items, which is continuously updated at the arrival of every new

item. We give the first algorithms for frequency moments, k-median clustering and k-center clustering in this model.

Chapter 2

Estimating Low Frequency Items in a Stream and Its Application to Bioinformatics

2.1 Introduction

In this chapter, we present our algorithm for estimating the number of low frequency items in a data stream. We start by presenting the problem.

2.1.1 Algorithmic problem

In the streaming model, the input is a sequence of data items: $D = \langle s_1, s_2, \ldots, s_m \rangle$, where each s_i is an item from the universe $\{1, 2, \ldots, n\}$. We need to make a single pass over D, processing each s_i fast using a limited amount of memory, and need to estimate the following statistics about D:

- n_i, the number of distinct items which appear exactly i times in D,

where $1 \leq i \leq L$ for some constant L such as 64. We denote by F_0, the number of items from the universe $\{1, 2, \ldots, n\}$ which appear at least once in the stream. We next present how this abstraction could model the problem of estimating k-mer abundance histogram in massive genetic datasets.

A genetic dataset essentially consists of a large sequence of symbols from the alphabet $\{A, C, T, G\}$. A k-mer is a subsequence of exactly k consecutive symbols for a parameter k such as 31. Our goal is to approximately compute the following:

- total number of k-mer patterns which appear exactly i times in the sequence

for small values of i such as $1 \leq i \leq 64$. We use the following example to illustrate this modeling.

Let $ACCTAGAGTAATTTGACAT$ be our dataset and let $k = 2$. We treat the dataset as the following stream: $D = \langle AC, CC, CT, TA, AG, GA, AG, GT, TA, AA, AT, TT, TT, TG, GA, AC, CA, AT\rangle$, where each data item is from the universe $\{AA, AC, AG, AT, CA, CC, CG, CT, GA, GC, GG, GT, TA, TC, TG, TT\}$ whose vector of frequencies in the D would be $\langle 1, 2, 2, 2, 1, 1, 0, 1, 2, 0, 0, 1, 2, 0, 1, 2\rangle$. Hence $n_1 = 6$ and $n_2 = 6$, where n_i is the number of k-mer patterns of length 2 which occur exactly i times in the dataset.

2.1.2 Previous work

Computing important functions of the frequencies of a stream of items has been studied extensively. In particular, Flajolet and Martin [35] gave the first algorithm for approximating F_0, the number of distinct items in a stream, up to a factor of 3. Subsequently in [7], three algorithms for this problem were given with varying space and time complexities. Finally, an optimal algorithm for estimating F_0 was given in [46]. The optimal lower bound was given earlier in [44, 66]. Another well-studied problem is the identification and frequency-computation of 'heavy hitters': the set of items which appear in the stream with *very high frequency* [13].

In contrast, the problem of counting *the number* of *very low frequency* items in the stream has received only limited attention from theoretical perspective. Melsted

Table 2.1: Results on estimating n_i for a constant i, for any error parameter $0 < \epsilon < 0.5$. Results marked with [$*$] are our contribution. Results with [†] use a perfectly random hash function whose space complexity is excluded.

Approximation	Space	Time
additive ϵF_0 [52, 60, 50] [†]	$O(\log n/\epsilon^2)$	$O(1)$
additive ϵF_0 [10] [$*$]	$O(\log n + \frac{1}{\epsilon^2}(\log \frac{1}{\epsilon} + \log\log n))$	$O(1)$
additive ϵF_0 [10] [$*$]	$\Omega(\log n + \frac{1}{\epsilon^2})$	-
multiplicative constant [10] [$*$]	$\Omega(n)$	-

and Halldorsson [52] gave the first algorithm for computing n_1. Later Sivadasan et al. [60] extended the algorithm of [52] for computing n_i for a general i. Lipovsky et al. [50] proposed a fix for a 'bias' in the algorithm of [60], which they verified experimentally. We note that all the above papers are based on random hashing and the analyses given assume perfectly random hash functions and hence the optimal space complexity is not analyzed. Mohamadi et al. [54] gave a heuristic algorithm based on statistical modeling.

2.1.3 Our results

We gave an efficient algorithm for estimating low frequency items in a data stream and show its optimality of space usage. Our algorithm gives state-of-the-art empirical performance on large genetic datasets [10]. We summarize our key results in Table 2.1 and in the following. Our algorithm is based on an observation that the third among the three algorithms given in [7] (henceforth referred to as BJKST) can be used to keep a pairwise random sample of distinct items from the set of distinct items, which in turn can be used to estimate n_i. Our algorithm uses a pairwise random hash function. Our lower bound uses known techniques from communication complexity [66].

In particular, we give the following upper bound for estimating n_i upto an additive error.

Theorem 2.1. *(Upper bound) For any constant $i \geq 1$ and $0 < \epsilon < 1$ there is a streaming algorithm that on a data stream over a universe of size n outputs an estimate $\hat{n}_i$ for n_i such that $|\hat{n}_i - n_i| \leq \epsilon F_0$ with probability at least $\frac{2}{3}$. This algorithm has space complexity $O(\log n + \frac{1}{\epsilon^2}(\log \frac{1}{\epsilon} + \log\log n))$ bits and has amortized update time $O(1)$.*

We next show that in general it takes prohibitive amounts of space for multiplicatively estimate n_i when the universe size is large.

Theorem 2.2. *(Hardness of multiplicative approximation) For any $i \geq 1$ and for any constant $c \geq 1$, any randomized streaming algorithm that outputs $\hat{n}_i$ on any stream, such that $\frac{n_i}{c} \leq \hat{n}_i \leq cn_i$, needs $\Omega(n)$ bits of space, where n is the universe size.*

Finally, we show that our space complexity from Theorem 2.1 for additively estimating n_i is optimal upto small factors.

Theorem 2.3. *(Hardness of additive approximation) For any $\epsilon < 0.5$ and $i \geq 1$ any streaming algorithm for estimating n_i of a stream of items over a universe of size n, up to at most ϵF_0 absolute error, with probability at least $\frac{2}{3}$, requires $\Omega(\log n + \frac{1}{\epsilon^2})$ space.*

We also show the following optimal theoretical bound for n_i estimation for a constant i and ϵ.

Theorem 2.4. *(Optimal theoretical bound for constant ϵ) For any cosntant $i \geq 1$ and $0 < \epsilon < 1$ there is a streaming algorithm that on a data stream over a universe of size n outputs an estimate $\hat{n}_i$ for n_i such that $|\hat{n}_i - n_i| \leq \epsilon F_0$ with probability at least $\frac{2}{3}$. This algorithm takes $l \geq i$ as a parameter and have space complexity $O(l \log l(\log n + (\frac{C_{l,i}}{\epsilon})^{\frac{2(l+1)}{l+1-i}}))$ bits and have amortized update time $O(l \log l)$. Here, $C_{l,i}$ is a constant that depends on i and l.*

2.1.4 Organization of the rest of the chapter

In Section 2.2 we present our algorithm and its analysis. In Section 2.2.1 we analyze its space complexity and establish Theorem 2.1. In Section 2.2.2 we prove the lower bounds, Theorem 2.2 and Theorem 2.3. In Section 2.3 we present an alternate algorithm for estimating n_i based on a reduction to F_0 and prove Theorem 2.4, which could be of independent theoretical interest.

2.2 Our algorithm and its optimality

In this section, we formally present and analyze Algorithm 1 for estimating n_i for a general i.

2.2.1 Upper bound

We show the main upper bound in this section. We denote the set of all distinct items in a data stream by $\mathcal{S}$ and the number of distinct items by F_0. Then $F_0 = |\mathcal{S}|$. We use Markov's and Chebyshev's inequalities.

Fact 2.5 (Markov's inequality)**.** *Let X be a non-negative random variable with finite expectation $E(X)$. Then for any constant $t > 0$,* $\Pr[X \geq t] \leq \frac{E[X]}{t}$.

Fact 2.6 (Chebyshev's inequality)**.** *Let X be a random variable with finite expectation $E(X)$ and variance* $\mathrm{Var}(X)$. *Then for any constant $t > 0$, $Pr[|X - E[X]| > t] \leq \frac{\mathrm{Var}(X)}{t^2}$.*

Algorithm 1 is a general algorithm which is based on the BJKST algorithm and maintains a set of random samples (more precisely a sketch) chosen from $\mathcal{S}$, that are uniform and pairwise independent, along with their frequencies in the stream. To get

Algorithm 1: Algorithm for estimating n_i. Constants d and L are fixed from tha analysis in 2.2.1

$h \leftarrow$ a 2-wise independent uniformly random hash function mapping $[n] \rightarrow [n]$;
$g \leftarrow$ a 2-wise independent uniformly random hash function mapping $[n] \rightarrow [\Theta(\frac{10d^2}{\epsilon^4} \log^2 n)]$;
$L \leftarrow$ a parameter fixed in the analysis;
$B_0 \leftarrow \phi$;
$s \leftarrow 0$;
for *arrival of data item* $j \in \{1, 2, \cdots, n\}$ *in the stream* **do**
    if $zeros(h(j)) \geq s$ **then**
        if $key = (g(j), zeros(h(j)) \in B_s$ **then**
            $key.count = key.count + 1$;
        else
            $Insert(key, B_s)$;
            $key.count = 1$;
        end
    end
    while $|B_s| \geq L$ **do**
        $B_{s+1} \leftarrow$ Remove all keys (α, β) with $\beta = s$ from B_s ;
        $s \leftarrow s + 1$;
    end
end
/* At the end of stream */
$s_f \leftarrow s$; // final value of s
$k_i \leftarrow$ the number of samples in B_{s_f} with count value exactly i;
Return $\hat{n}_i = k_i \cdot 2^{s_f}$;

an estimate of n_i we just count the number of items in the sample that has frequency exactly i, and scale this count appropriately.

The analysis is similar to that of BJKST. We introduce necessary notation. The function $zeros(x)$ gives the number of trailing zeros of a bit-string x. We fix $L = \frac{10c}{\epsilon^2}$ in the algorithm for a constant c which we fix later. Let s_f denote the random variable that gives final value of s in Algorithm 1. For each $j \in \mathcal{S}$ and each integer

$0 \leq s \leq \lceil \log n \rceil$, define the following indicator random variables:

$$X_{j,s} = 1 \quad \text{if } zeros(h(j)) \geq s$$
$$= 0 \quad \text{otherwise}$$

Then, $\Pr(X_{j,s} = 1) = \frac{1}{2^s}$. Let $X_s = \sum_{j \in \mathcal{S}} X_{j,s}$. Then, $E(X_s) = \frac{F_0}{2^s}$.

The following key lemma guarantees the approximation of our algorithm.

Lemma 2.7. *Let k_i be the number of items with count exactly i in B_{s_f} in Algorithm 1. Let $\hat{n}_i = k_i \cdot 2^{s_f}$. Then $|\hat{n}_i - n_i| \leq \epsilon F_0$ with probability at least $1 - \frac{(10+c)}{5c}$.*

Proof. For the analysis we will assume $F_0 \geq \frac{c}{2\epsilon^2}$ (for smaller F_0, the algorithm will give exact count as we describe later). Let $s_* \geq 0$ be the unique integer such that $\frac{c}{2\epsilon^2} \leq \frac{F_0}{2^{s_*}} < \frac{c}{\epsilon^2}$.

Claim 2.8. *For $F_0 \geq \frac{c}{2\epsilon^2}$, $\Pr(s_f > s_*) \leq \frac{1}{10}$.*

Proof. (of claim.) Note that for s_f to be $< s_*$, X_{s_*} should be $\geq L$. Hence $Pr(s_f < s_*) \leq Pr(X_{s_*} \geq L) = Pr(X_{s_*} \geq \frac{10c}{\epsilon^2}) \leq \frac{1}{10}$. The last inequality because $E(X_{s_*}) = \frac{F_0}{2^{s_*}} \leq \frac{c}{\epsilon^2}$ and hence by Markov's inequality $\Pr(X_{s_*} \geq \frac{10c}{\epsilon^2}) \leq \frac{1}{10}$. □

Let $N_i \subseteq \mathcal{S}$ denote the set of distinct items in the stream having frequency exactly i. Thus $n_i = |N_i|$. Fix any integer $1 \leq s \leq s_*$. Recall the random variable $X_{j,s}$ for each $j \in \mathcal{S}$: $X_{j,s} = 1$ if $zeros(h(j)) \geq s$ and 0 otherwise. (Thus $\Pr(X_{j,s} = 1) = \frac{1}{2^s}$). Let $X_s = \sum_{j \in N_i} X_{j,s}$ (with slight abuse of notation). Then, $E(X_s) = n_i/2^s$ and from the pairwise independence of h,

$$\text{Var}(X_s) = \sum_{j \in N_i} \text{Var}(X_{j,s}) \leq n_i/2^s \leq F_0/2^s.$$

Using Chebyshev's inequality,

$$\Pr(|X_s - n_i/2^s| > t) \leq \frac{\mathrm{Var}(X_s)}{t^2} \leq \frac{F_0}{2^s t^2}.$$

We choose $t = \epsilon F_0/2^s$, to get that $\Pr(|2^s X_s - n_i| > \epsilon F_0) \leq \frac{2^s}{\epsilon^2 F_0} \leq 2/c$. The last inequality because $s \leq s_* \Rightarrow 2^s/F_0 \leq 2^{s_*}/F_0 \leq 2\epsilon^2/c$.

Notice, the final bucket B_{s_f} includes each item $j \in N_i$ satisfying $zeros(h(j)) \geq s_f$. Moreover, assuming g is one-to-one (no collision), the count value for each such j will be exactly i, implying $X_{s_f} = k_i$. So $2^{s_f} X_{s_f} = k_i \cdot 2^{s_f} = \hat{n}_i$. If $F_0 < \frac{c}{2\epsilon^2} < L$, $s_f = 0$ and $\hat{n}_i = k_i = n_i$. Since this holds for an arbitrary $s : 1 \leq s \leq s_*$, the statement holds for s_f in particular, provided $s_f \leq s_*$.

The probability that $s_f > s_*$ is $\leq 1/10$. Also probability that g is not one-to-one is $\leq 1/10$ (see Lemma below). Hence $\Pr(|\hat{n}_i - n_i| > \epsilon F_0) \leq (10 + c)/5c$. □

The following lemma is standard and shows that with high probability g does not have any collision on the set of samples. Notice, for the value of L fixed earlier, $\sum_{s \leq s_f} |B_s| \leq \frac{10c(1+\log n)}{\epsilon^2} \leq \frac{d \log n}{\epsilon^2}$ for some constant d. Then by Lemma 2.9, g is one-to-one.

Lemma 2.9. *Let d be any constant. On any fixed set $D \subseteq [n]$ of at most $\frac{d}{\epsilon^2} \log n$ items, $g : [n] \to [\frac{10d^2 \log^2 n}{\epsilon^4}]$ is one-to-one except for probability at most $\frac{1}{10}$.*

Proof. Probability that any two fixed items a, b maps to the same element by g, $\Pr(g(a) = g(b)) = \frac{\epsilon^4}{10d^2 \log^2 n}$. Restrict g to a domain D of size $\frac{d}{\epsilon^2} \log n$. Taking union bound over choices of $a, b \in D$; we get that the probability that g collides is at most $\frac{1}{10}$. □

At this point, we present the correctness and efficiency of Algorithm 1.

Theorem 2.1. *(Upper bound) For any constant $i \geq 1$ and $0 < \epsilon < 1$ there is a streaming algorithm that on a data stream over a universe of size n outputs an estimate $\hat{n}_i$ for n_i such that $|\hat{n}_i - n_i| \leq \epsilon F_0$ with probability at least $\frac{2}{3}$. This algorithm has space complexity $O(\log n + \frac{1}{\epsilon^2}(\log \frac{1}{\epsilon} + \log\log n))$ bits and has amortized update time $O(1)$.*

Proof. We run Algorithm 1 on the given stream. At any point, buckets B_s for at most two s values need to be kept. Storing each key $(g(j), zeros(h(j)))$ takes $O(\log \frac{1}{\epsilon} + \log\log n)$ bits of space. Count for each item could be large in general but for our purpose, count up to $(i+1)$ suffices. This needs $O(\log i)$ bits per key in addition. The hash functions require $O(\log n)$ bits. The buckets B_s can be implemented using a balanced binary search tree, requiring $O(\log \frac{1}{\epsilon})$ time except when B_s needs to be updated. Note that, if the length of the stream, m is at most L, every element from the stream will be stored and the running time will be $O(m)$. Otherwise, $m > L$, hence, $O(\log \frac{1}{\epsilon}) = O(\log m)$. In the RAM model it is standard in the literature to treat any $O(\log m)$ bit operation as $O(1)$ for time complexity. Updating B_s requires $O(\frac{1}{\epsilon^2}\log n)$ time over the entire course of the algorithm. From Lemma 2.7, $\hat{n}_i$ is the required output of the algorithm. Choosing the constant $c = 20$, we have $\Pr(|\hat{n}_i - n_i| > \epsilon F_0) \leq 1/3$. □

2.2.2 Lower bounds

For hardness results, we use communication complexity techniques. In two party communication complexity, there are two players Alice and Bob who together has to compute a function of interest on the union of their input data. The main resource of interest is the communication between Alice and Bob. Communication complexity has been a very effective technique in establishing lower bound results in streaming

algorithms. We establish our lower bounds by giving reductions from the communication problems INDEX, EQUALITY and GAP-HAMMING DISTANCE which we define next, to our frequency estimation problem. In this subsection we use definitions and notation that are standard in communication complexity literature. Please see the text book by Nisan and Kushilevitz [48] for fundamentals of communications complexity.

Definition 2.10 (INDEX(n)). *Alice has an n-bit long private string X and Bob has an integer $1 \leq j \leq n$. Alice may send a single message to Bob after which Bob needs to output $X[j]$.*

Definition 2.11 (EQ(n)). *Alice and Bob both have a n-bit long private string X and Y respectively, and they need to decide whether $X = Y$.*

Definition 2.12 (GAP-HAM(n)). *Alice and Bob both have a n-bit long private string X and Y respectively and they need to decide whether $\text{HAM}(X, Y) \geq \frac{n}{2} + \sqrt{n}$ or $\text{HAM}(X, Y) \leq \frac{n}{2} - \sqrt{n}$, given the promise that one of these two conditions holds. Here HAM denotes the hamming distance function.*

We will use the following known communication complexity lower bounds.

Fact 2.13 ([48]). *The randomized (private) one-way communication complexity of EQ(n) is $\Omega(\log n)$. The problem is hard for inputs coming from the following set: $C \subset \{0,1\}^n, |support(c)| = \frac{n}{100}, |c \cap c'| \leq \frac{n}{2000} \forall c \neq c' \in C, |C| = 2^{\Omega(n)}$.*

Theorem 2.14 ([66, 45, 21]). *The randomized one-way communication complexity of GAP-HAM(n) is $\Omega(n)$.*

Theorem 2.15 ([47]). *The randomized one-way communication complexity of INDEX(n) is $\Omega(n)$.*

First we show that any multiplicative approximation of n_i is hard in general.

Theorem 2.2. *(Hardness of multiplicative approximation) For any $i \geq 1$ and for any constant $c \geq 1$, any randomized streaming algorithm that outputs $\hat{n}_i$ on any stream, such that $\frac{n_i}{c} \leq \hat{n}_i \leq cn_i$, needs $\Omega(n)$ bits of space, where n is the universe size.*

Proof. Given an instance (x, j) where is an n bit vector and $1 \leq j \leq n$ of the INDEX problem, Alice generates a stream which consists of a single occurrence for each non-zero coordinates of x. Let us call this stream σ_A. Bob generates a second stream σ_B which consists of $(i + 1)$ occurrences for each coordinate except for the j^{th} one. He also adds $(i - 1)$ occurrences of the j^{th} coordinate. Then in the combined stream σ_A followed by σ_B, n_i is 1 if $x[j]$ is 1. Otherwise, it is 0.

Consider any streaming algorithm $\mathcal{A}$ for computing a factor c approximation of n_i of n items that uses $S(n)$ bits of space. Alice runs $\mathcal{A}$ on σ_A and forwards the memory content to Bob using $S(n)$ bits of communication. Bob completes running of $\mathcal{A}$ on σ_B and obtains $\hat{n}_i$ of the combined stream. By previous discussion, Bob can use whether $\hat{n}_i$ is zero or not to decide the hard problem INDEX(n). Hence, from Theorem 2.15, $S(n) = \Omega(n)$. □

We next give space lower bound for additive approximations. The proof of this lower bound closely follows that of the lower bound for F_0 estimation [66].

Theorem 2.3. *(Hardness of additive approximation) For any $\epsilon < 0.5$ and $i \geq 1$ any streaming algorithm for estimating n_i of a stream of items over a universe of size n, up to at most ϵF_0 absolute error, with probability at least $\frac{2}{3}$, requires $\Omega(\log n + \frac{1}{\epsilon^2})$ space.*

Proof. We first prove the result for n_1. Consider an instance (X, Y) of the problem GAP-HAM$(\frac{1}{\epsilon^2})$. We will treat the inputs X and Y as subsets of $\{1, 2, \cdots, \frac{1}{\epsilon^2}\}$. Con-

sider a stream produce by the elements of X followed by the elements of Y. Alice starts to run a streaming algorithm for estimating n_1 on X. Upon completion, she sends the memory contents of the algorithm of size S to Bob, who then completes the algorithm on Y. Notice, the true value of n_1 is exactly $\textsc{Ham}(X, Y)$ and also, $F_0 \leq \frac{1}{\epsilon^2}$. Thus an ϵf_0 additive error translates to an absolute error of at most $\frac{1}{\epsilon}$ for the hamming distance. Hence by checking whether the estimate for $n_1 < 1/2\epsilon^2$ or not, we can decide $\textsc{Gap-Ham}(\frac{1}{\epsilon^2})$. Hence, from Theorem 2.14, $S \geq \Omega(\frac{1}{\epsilon^2})$ bits. The result can be generalized to larger stream size by padding a single fixed item enough number of times to either the stream of Alice or Bob. This increases F_0 by 1 and keeps n_1 unchanged.

For the second term, we start from the hard instance of $\mathrm{EQ}(n)$ from Fact 2.13. As before consider a stream produced by the elements of X followed by the elements of Y (where $X, Y \subseteq \{1, \cdots, n\}$). Alice starts to run a streaming algorithm for estimating n_1 on X. Upon completion, she sends the memory contents of the algorithm of size S to Bob, who then completes the algorithm on Y. If the two sets are equal, $n_1 = 0$ and this estimate will be at most $\epsilon F_0 \leq \epsilon \frac{n}{100}$. Otherwise, since two different sets of the distribution have a low overlap, the estimate will be at least $(1 - \epsilon)\frac{n}{50} - \frac{n}{1000}$. For $\epsilon < 0.5$, this gap can exploited to decide the hard problem $\mathrm{EQ}(n)$, which requires $\Omega(\log n)$ bits.

In general, for larger values of i, the two parties could repeat each item exactly i times while creating the stream and the rest of the arguments remain the same. $\square$

2.3 Alternate optimal algorithm for small i and ϵ based on a reduction to F_0 computation

In this section, we present an alternate algorithm that works efficiently when i and ϵ are small constants. This algorithm is based on a completely different approach of F_0 computation on a sub-sampled stream, which could be of independent theoretical interest. Computation of statistics over a sub-sampled stream has been investigated before [51].

We recall the notations and also introduce new ones. The stream σ consists of m data elements each coming from a universal item set of size n. For an item a, the frequency of a is the number of occurrences of a in σ. For an integer i, n_i denote the number of items in the stream with frequency i. Let F_0 denote the the number of elements having non-zero frequency in σ (also known as 0^{th} frequency moment of σ). Our goal is to approximate n_i for a given i.

Consider the following process: For a probability p, obtain a substream σ'_p by keeping each item of σ with probability p. Let $F_0^p = F_0(\sigma'_p)$ be a renaming of this random variable. The following lemma gives the expected value of F_0^p for this process.

Lemma 2.16. $E(F_0^p) = pn_1 + (1-(1-p)^2)n_2 + \cdots + (1-(1-p)^m)n_m$

Proof. Consider an item a with frequency i in σ. The probability that none of these i occurances of a appear in σ' is exactly $(1-p)^i$. With the remaining probability it appears and contributes exactly 1 to F_0^p. The expected contribution to F_0^p for element a will be $(1-(1-p)^i)$. Using linearity of expectation and the fact that there are exactly n_i items with frequency i, the result follows. □

It can be observed, $pF_0 < E(F_0^p) < F_0$. A form of Chernoff's bound could be used to show concentration of F_0^p around $E(F_0^p)$.

Theorem 2.17 (Chernoff's bound)**.** *Let* $X = \sum_{i=1}^{n} X_i$ *such that each* X_i *is an independent indicator random variable. Then,* $\Pr(|X - E(X)| \leq \epsilon E(X)) \geq (1 - 2e^{-\frac{\epsilon^2 E(X)}{3}})$ *for any* $0 < \epsilon < 1$.

For an item a let X_a to be the indicator random variable for event a appearing at least once in σ'. Since the selection of each element into the substream is independent, X_a are all independent. The above theorem could be applied to get the following claim.

Corollary 2.18. *For* $p > \frac{1}{2}$ *and for any* $a > 0$ *such that* $F_0 > \frac{6a}{\epsilon^2}$, $\Pr(|F_0^p - E(F_0^p)| \leq \epsilon E(F_0^p)) > (1 - 2/e^a)$.

Proof. $\frac{\epsilon^2 E(F_0^p)}{3} \geq \frac{\epsilon^2 p F_0}{3} > \frac{\epsilon^2 F_0}{6} > a$. From Theorem 2.17, the required probability $\geq (1 - 2e^{-\frac{\epsilon^2 E(F_0^p)}{3}}) > (1 - \frac{2}{e^a})$. □

The previous process could be repeated a number of times for a given p so that their median lies in the desired range with high probability. A key lemma follows next. This shows how to obtain the values n_i for a small set of values of i given values of F_0^p for enough number of p's. The intuition is that, in the substream, elements which have smaller frequencies will have lesser chance of contributing to F_0^p. On the other hand, higher frequency elements will have a higher chance.

Lemma 2.19. *For any* l: $1 \leq l \leq m$ *and any set of distinct positive values* $0 < p_i < 1$ *where* $1 \leq i \leq l$*, the following holds:*

$$\begin{pmatrix} n_1 \\ \vdots \\ n_l \end{pmatrix} = \begin{bmatrix} (1-p_1) & \cdots & (1-p_1)^l \\ \vdots & \vdots & \vdots \\ (1-p_l) & \cdots & (1-p_l)^l \end{bmatrix}^{-1} \begin{pmatrix} F_0 - E(F_0^{p_1}) - e_1 \\ \vdots \\ F_0 - E(F_0^{p_l}) - e_l \end{pmatrix}$$

where $e_j = \sum_{t=l+1}^{m} (1-p_j)^t n_t$.

Proof. By definition $F_0 = n_1 + n_2 + \cdots + n_m$. Then, from Lemma 2.16,

$$\begin{aligned}(F_0 - E(F_0^p)) &= (1-p)n_1 + (1-p)^2 n_2 + \cdots + (1-p)^m n_m \\ &= (1-p)n_1 + \cdots + (1-p)^l n_l + e\end{aligned}$$

where $e = \sum_{t=l+1}^{m} (1-p)^t n_t$. For l such distinct values of p,

$$\begin{pmatrix} F_0 - E(F_0^{p_1})) \\ \vdots \\ F_0 - E(F_0^{p_l})) \end{pmatrix} = \begin{bmatrix} (1-p_1) & \cdots & (1-p_1)^l \\ \vdots & \vdots & \vdots \\ (1-p_l) & \cdots & (1-p_l)^l \end{bmatrix} \begin{pmatrix} n_1 \\ \vdots \\ n_l \end{pmatrix} + \begin{pmatrix} e_1 \\ \vdots \\ e_l \end{pmatrix}$$

and the result follows. □

2.3.1 A simple algorithm for estimating n_1

The above Lemma could be used to design an algorithm for computing the number of low frequency in a stream. An issue that arises is that the values of $E(F_0^{p_i})$'s and F_0 can only be approximated up to a relative error ξ. This introduces some error for the n_i's. The later error seems to depend on the entries of the inverse matrix. Larger value of these entries would result in a larger error. As a warm-up, let us obtain n_1 approximately.

Lemma 2.20. *Let $p_1 = (1 - \sqrt{2\xi})$ for any $0 < \xi < 1$. Let $\tilde{F}_0$ and $\tilde{F}_0^{p_1}$ are $(1 \pm \xi)$ approximations for F_0 and $E(F_0^{p_1})$ respectively. Then, one can obtain $\tilde{n}_1$, such that, $|\tilde{n}_1 - n_1| \leq 2\sqrt{2\xi}F_0$.*

Proof. From Lemma 2.19, with $l = 1$, $n_1 = \frac{1}{1-p_1}(F_0 - E(F_0^{p_1}) - e_1)$ for all $0 < p_1 < 1$, p_1 will be fixed later. Let $\tilde{n}_1 = \frac{1}{1-p_1}(\tilde{F}_0 - \tilde{F}_0^{p_1})$. Then, $|\tilde{n}_1 - n_1| \leq \frac{1}{1-p_1}(\xi(F_0 +$

$E(F_0^{p_1})) + e_1)$. Using the facts, $E(F_0^{p_1}) \leq F_0$ and $e_1 = \sum_{t=2}^{m}(1-p_1)^t n_t \leq (1-p_1)^2 F_0$, the error $\leq (\frac{2\xi}{1-p_1} + (1-p_1))F_0$. Finally, choose $p_1 = (1-\sqrt{2\xi})$ to get the result. □

An algorithm for n_1 is given below. For the F_0 computations, the following result from [46] will be used.

Theorem 2.21 (Kane *et al.* [46] restated)**.** *There is a streaming algorithm for computing F_0 of n items up to approximation ratio $(1 \pm \epsilon)$ with probability at least $2/3$ that uses $O(\log n + \frac{1}{\epsilon^2})$ bits of space. This algorithm has update time $O(1)$ per item and reporting time $O(1)$.*

Algorithm 2: Algorithm for estimation of n_1 up to at most ϵF_0 absolute error. Works for $F_0 > \frac{24^3}{\epsilon^4}$ and succeeds with probability at least $\frac{2}{3}$

$l = 1$;
Choose $\epsilon < 1$;
$p_1 = (1 - \frac{\epsilon}{2})$;
Denote by $F_0(\epsilon_1, \delta_1, k)$, the k^{th} instance of the F_0 algorithm with relative error at most ϵ_1 and failure probability at most δ_1;
for *arrival of data item d in the stream* **do**
    `/* Process current item */`
    if *Toss*(p_1) = *True* **then**
        `/* Obtain substream implicitly */`
        Update $F_0(\frac{\epsilon^2}{24}, \frac{1}{12}, 1)$ with item d to get $\tilde{F}_0^{p_1}$;
    end
    Update $F_0(\frac{\epsilon^2}{8}, \frac{1}{6}, 2)$ with item d to get $\tilde{F}_0$;
end
`/* At the end of stream */`
$F = \tilde{F}_0 - \tilde{F}_0^{p_1}$;
$\tilde{n}_1 = \frac{2}{\epsilon}F$;
Return $\tilde{n}_1$;

Lemma 2.22. *Algorithm 2 computes n_1, the number of items with frequency 1, in a data stream up to absolute error at most ϵF_0 with probability at least $\frac{2}{3}$ for any*

$0 < \epsilon < 1$. *It uses* $O(\log n + \frac{1}{\epsilon^4})$ *bits of space,* $O(1)$ *update time and requires* $O(1)$ *time to report the output.*

Proof. A pseudocode for the algorithm (with constant error probability) is given in Algorithm 2. From line number 7, $\tilde{F}_0^{p_1}$ is an $(1 \pm \frac{\epsilon^2}{24})$ approximation of $F_0^{p_1}$, with a failure probability at most $\frac{1}{12}$. Notice that for $\epsilon < 1$, $p_1 > \frac{1}{2}$. From Corollary 2.18 with $a = 4$ and $F_0 > \frac{24^3}{\epsilon^4}$, $F_0^{p_1}$ is an $(1 \pm \frac{\epsilon^2}{24})$ approximation of $E(F_0^{p_1})$, with a failure probability at most $\frac{2}{e^4} < \frac{1}{12}$. Together, $\tilde{F}_0^{p_1}$ is an $(1 \pm \frac{\epsilon^2}{24})^2 < (1 \pm \frac{\epsilon^2}{8})$ approximation for $E(F_0^{p_1})$ with a failure probability at most $\frac{1}{6}$. Similarly From line number 9, $\tilde{F}_0$ is an $(1 \pm \frac{\epsilon^2}{8})$ approximation for F_0 with a failure probability at most $\frac{1}{6}$. It follows from Lemma 2.20 with $\xi = \frac{\epsilon^2}{8}$, $\tilde{n}_1$ is an approximation of n_1 with at most ϵF_0 absolute error with a failure probability at most $\frac{1}{3}$. From Theorem 2.21, in total it takes $O(\log n + \frac{1}{\epsilon^4})$ space, $O(1)$ reporting and update times. □

2.3.2 Algorithm for n_i for $i > 1$ and improving the dependency on the error term

In general, to obtain n_i, one needs to analyze the entries of the inverse matrix in Lemma 2.19. In the process, the absolute error for n_1 will be improved as well. The following known fact about the inverse of Vandermonde matrices, will be useful.

Fact 2.23. *Let* $P = \begin{bmatrix} a_1 & a_1^2 & \cdots & a_1^l \\ a_2 & a_2^2 & \cdots & a_2^l \\ \vdots & \vdots & \vdots & \vdots \\ a_l & a_l^2 & \cdots & a_l^l \end{bmatrix}$ *and* $Q = P^{-1}$.

Then, $Q_{ij} = \frac{(-1)^{i-1}\Pi_{\bar{j}}^{l-i}}{a_j(\Pi_{\substack{1 \le m \le l \\ m \neq j}}(a_j - a_m))}$ *where* $\Pi_{\bar{j}}^{l-i}$ *is the symmetric polynomial of degree* $(l-i)$ *on the variables* $\{a_1, a_2, \cdots, a_l\} \setminus \{a_j\}$.

The values of p_i's need to be fixed in order to derive an upper bound for the error. This requires to first fix a value for l, which in turn depend on the desired errors (e_i's). If e_i is too small, some of the entries of P become very small, and hence some entries of Q become very large. But setting $e_i \leq \epsilon F_0$ is good enough.

Lemma 2.24. *Let e_i and p_i be as in Lemma 2.19. For any $0 < \epsilon < 1$, let $p_i \geq (1 - \epsilon^{\frac{1}{l+1}})$. Then, $e_i \leq \epsilon F_0$.*

Proof.

$$\begin{aligned} e_i &= (1-p_i)^{l+1} n_{l+1} + (1-p_i)^{l+2} n_{l+2} + \cdots + (1-p_i)^m n_m \\ &\leq (1-p_i)^{l+1} (n_{l+1} + n_{l+2} + \cdots + n_{m-1} + n_m) \\ &\leq (1-p_i)^{l+1} F_0 \end{aligned}$$

For $p_i \geq (1 - \epsilon^{\frac{1}{l+1}})$, the last term is at most ϵF_0. □

The next lemma states an upper bound for the entries Q_{ij}, for suitable choices of p_i's.

Lemma 2.25. *Let $a_j = 1 - (p + (j-1)\delta)$ for all $j = 1$ to l in Fact 2.23 for some $0 < p < 1$ such that $a_l > 0$. Then, $|Q_{ij}| < \frac{\binom{l-1}{l-i}(1-p)^{l-i}}{(1-(p+(l-1)\delta))\delta^{l-1}(\lfloor \frac{l}{2} \rfloor)!(\lfloor \frac{l-1}{2} \rfloor)!}$ for all i, j.*

Proof. By definition $a_i \geq 1-(p+(l-1)\delta), \forall j$. Also, $\Pi_{m \neq j} |a_j - a_m|$ is minimum when j is $\lfloor \frac{l}{2} \rfloor$ and this minimum value is $\delta^{l-1}(\lfloor \frac{l}{2} \rfloor)!(\lfloor \frac{l-1}{2} \rfloor)!$. Also, $\Pi_{\tilde{j}}^{l-i} < \binom{l-1}{l-i}(1-p)^{l-i}$. The result follows from Fact 2.23. □

The following lemma gives an upper-bound for the error for estimation of the n_i's for any i in general.

Lemma 2.26. *Let $p_i = (p + (i-1)\delta)$, for an integer $1 \leq i \leq l$, for some p, δ; such that $0 < p \leq p + (l-1)\delta < 1$. For each $j = 1$ to l, let $\tilde{F}_0$ and $\tilde{F}_0^{p_j}$ are $(1 \pm \xi)$*

approximations for F_0 *and* $E(F_0^{p_j})$ *respectively for some* $\xi < \min\{\frac{l+1}{2i} - \frac{1}{2}, 1\}$. *Then, for this particular* i, *one can obtain* $\tilde{n}_i$, *such that,* $|\tilde{n}_i - n_i| \leq (C_{l,i}\xi^{1-\frac{i}{l+1}})F_0$ *for any* $0 < \xi < 1$ *and* $C_{l,i}$ *is a constant depending on* l *and* i.

Proof. Recall, by Lemma 2.19, $n_i = \sum_{j=1}^{l} Q_{ij}(F_0 - E(F_0^{p_j}) - e_j)$, where Q is the inverse matrix of the RHS. Let us set $\tilde{n}_i = \sum_{j=1}^{l} Q_{ij}(\tilde{F}_0 - \tilde{F}_0^{p_j}))$. Then for all i, the absolute error,

$$
\begin{aligned}
|\tilde{n}_i - n_i| &\leq \sum_{j=1}^{l} |Q_{ij}|.(\xi F_0 + \xi E(F_0^{p_j}) + e_j) \\
&\leq \sum_{j=1}^{l} |Q_{ij}|.(\xi F_0 + \xi F_0 + e_j) && \text{(Since } E(F_0^{p_j}) < F_0\text{)} \\
&\leq \sum_{j=1}^{l} |Q_{ij}|.(\xi F_0 + \xi F_0 + \xi_i F_0) \\
& && \text{(From Lemma 2.24 choosing } p_j = (1 - \xi_i^{\frac{1}{l+1}}) + (j-1)\delta\text{)} \\
&\leq l(\max_j |Q_{ij}|)(2\xi + \xi_i)F_0 \\
&\leq \frac{\binom{l-1}{l-i} l}{(\lfloor \frac{l}{2} \rfloor)!(\lfloor \frac{l-1}{2} \rfloor)!} \frac{\xi_i^{\frac{l-i}{l+1}}}{\delta^{l-1}(\xi_i^{\frac{1}{l+1}} - (l-1)\delta)}(2\xi + \xi_i)F_0 \\
& && \text{(From Lemma 2.25, using value of } p_j \text{ set before)} \\
&= \frac{\binom{l-1}{l-i} l^{l+1}}{(\lfloor \frac{l}{2} \rfloor)!(\lfloor \frac{l-1}{2} \rfloor)!} \frac{(2\xi + \xi_i)}{\xi_i^{\frac{i}{l+1}}} F_0 \quad \text{(By minimizing with respect to } \delta\text{, } \delta = \tfrac{1}{l}\xi_i^{\frac{1}{l+1}}\text{)} \\
&= \frac{2\binom{l-1}{l-i} l^{l+1}(l+1)}{(\lfloor \frac{l}{2} \rfloor)!(\lfloor \frac{l-1}{2} \rfloor)!(2i)^{\frac{i}{l+1}}(l+1-i)^{1-\frac{i}{l+1}}} \xi^{1-\frac{i}{l+1}} F_0 \\
& && \text{(By minimizing with respect to } \xi_i\text{, } \xi_i = \tfrac{2i\xi}{l+1-i} \text{ for } \xi < \tfrac{l+1-i}{2i}\text{)} \\
&= C_{l,i}\xi^{1-\frac{i}{l+1}} F_0
\end{aligned}
$$

The required probability values could be $p_j = (1 - (\frac{2i\xi}{l+1-i})^{\frac{1}{l+1}}(1 - \frac{j-1}{l}))$ for $j = 1$ to l. □

Table 2.2: $\lceil C_{l,i} \rceil$ values for first 10 values of i and l. i indexes column and l indexes row.

3	0	0	0	0	0	0	0	0	0
24	20	0	0	0	0	0	0	0	0
240	459	170	0	0	0	0	0	0	0
1471	4564	3973	971	0	0	0	0	0	0
10922	46875	66292	37205	6881	0	0	0	0	0
63682	348093	686243	621547	258632	38815	0	0	0	0
428068	2835468	7178441	9058511	6036325	2004979	254531	0	0	0
2446389	18998896	58720256	95301107	88236885	46606299	12926754	1426892	0	0
15635485	139071650	507196952	1007149378	1198520991	876774458	384383412	91753071	8980228	0
88424563	885244788	3718064400	8732195663	12717716720	11941057341	7228103601	2713220512	569506698	50150545

Lemma 2.26 gives a method to estimate the n_i values one at a time up to some absolute error. The dependency of the error term on ξ is largest for n_1 and gets smaller for $n_2, n_3, \cdots, n_l$. For a fixed i, this dependency is larger for larger values of l. Note, Lemma 2.20 can be derived as a special case. This gives the following algorithm (Algorithm 3), which uses the F_0 result of Theorem 2.21. A few values of $C_{l,i}$ can be found in Table 2.2.

Theorem 2.4. *(Optimal theoretical bound for constant ϵ) For any cosntant $i \geq 1$ and $0 < \epsilon < 1$ there is a streaming algorithm that on a data stream over a universe of size n outputs an estimate $\hat{n}_i$ for n_i such that $|\hat{n}_i - n_i| \leq \epsilon F_0$ with probability at least $\frac{2}{3}$. This algorithm takes $l \geq i$ as a parameter and have space complexity $O(l \log l(\log n + (\frac{C_{l,i}}{\epsilon})^{\frac{2(l+1)}{l+1-i}}))$ bits and have amortized update time $O(l \log l)$. Here, $C_{l,i}$ is a constant that depends on i and l.*

Proof. From Lemma 2.26, to obtain an absolute error of at most ϵF_0, $\xi = (\frac{\epsilon}{C_{l,i}})^{\frac{l+1}{l+1-i}} < \min\{\frac{l+1}{2i} - \frac{1}{2}, 1\}$ and $p_j = (1 - (\frac{2i}{l+1-i})^{\frac{1}{l+1}} (\frac{\epsilon}{C_{l,i}})^{\frac{1}{l+1-i}} (1 - \frac{j-1}{l})) > \frac{1}{2}$. $\epsilon < 1 < C_{l,i} \min\{(\frac{l+1}{2i} - \frac{1}{2})^{\frac{l+1-i}{l+1}}, 1\}$ works. ξ is the upper bound of relative error required from F_0 and $E(F_0^p)$.

We start with Algorithm 3. Line number 9 computes $\tilde{F}_0^{p_j}$, a $(1 \pm \frac{\xi}{3})$ approximation of $F_0^{p_j}$ except for failure probability at most $\frac{1}{6(l+1)}$. Assume $F_0 > (\frac{54 \log 12(l+1)}{\xi^2}) = \Omega(\log l(\frac{C_{l,i}}{\epsilon})^{\frac{2(l+1)}{l+1-i}})$. Otherwise, it is sufficient to maintain a frequency table. From Corollary 2.18 with $a = \log 12(l+1)$, $F_0^{p_j}$ is at most $(1 \pm \frac{\xi}{3})$ approximation of $E(F_0^{p_j})$,

Algorithm 3: Algorithm for estimation of n_i for a particular i, up to at most ϵF_0 absolute error. Works for $F_0 > (54 \log 12(l+1))(\frac{C_{l,i}}{\epsilon})^{\frac{2(l+1)}{l+1-i}}$ and succeeds with probability at least $\frac{2}{3}$.

1 Choose $l \geq i$;
2 Choose $\epsilon < 1$;
3 $P = \{p_j = 1 - (\frac{2i}{l+1-i})^{\frac{1}{l+1}}(\frac{\epsilon}{C_{l,i}})^{\frac{1}{l+1-i}}(1 - \frac{j-1}{l})\}_{j=1}^{l}$;
4 $\xi = (\frac{\epsilon}{C_{l,i}})^{\frac{l+1}{l+1-i}}$;
5 Denote by $F_0(\epsilon_1, \delta_1, k)$, the k^{th} instance of the F_0 algorithm with relative error at most ϵ_1 and failure probability at most δ_1;
6 **for** *arrival of data item d in the stream* **do**
  `/* Process current item */`
7 **for** $j = 1$ *to* l **do**
8 **if** *Toss*(p_j) = *True* **then**
  `/* Obtain substream implicitly */`
9 Update $F_0(\frac{\xi}{3}, \frac{1}{6(l+1)}, j)$ with item d to get $\tilde{F}_0^{p_j}$;
10 **end**
11 **end**
12 Update $F_0(\xi, \frac{1}{3(l+1)}, l+1)$ with item d to get $\tilde{F}_0$;
13 **end**
  `/* At the end of stream */`
14 **for** $j = 1$ *to* l **do**
15 $F^j = \tilde{F}_0 - \tilde{F}_0^{p_j}$;
16 **end**
17 $N = \begin{vmatrix} (1-p_1) & \cdots & (1-p_1)^l \\ \vdots & \vdots & \vdots \\ (1-p_l) & \cdots & (1-p_l)^l \end{vmatrix}^{-1} * \begin{pmatrix} F^1 \\ \vdots \\ F^l \end{pmatrix}$;
18 $\tilde{n}_i = i^{\text{th}}$ value in N;
19 Return $\tilde{n}_i$;

except for failure probability at most $\frac{1}{6(l+1)}$. Hence, for each $j = 1$ to l, $\tilde{F}_0^{p_j}$ is at most $(1 \pm \frac{\xi}{3})^2 \leq (1 \pm \xi)$ approximation of $E(F_0^{p_j})$ except for failure probability at most $\frac{1}{3(l+1)}$. Line number 12 computes $\tilde{F}_0$, a $(1 \pm \xi)$ approximation of F_0 except for failure probability at most $\frac{1}{3(l+1)}$. Apply Lemma 2.26 at this point to obtain $\tilde{n}_i$ except for failure probability at most $\frac{1}{3}$. From Theorem 2.21, for each of the $(l+1)$ F_0 computations, it takes $O(\log l(\log n + \frac{1}{\xi^2}))$ space, $O(\log l)$ update time. □

It may be observed, for constant l and for $i = 1$, the space complexity of Theorem 2.4 is $O(\log n + (\frac{1}{\epsilon})^{2(1+\frac{1}{l})})$, which is close to the lower bound of Theorem 2.3, for a large l. Also, for a constant ϵ and i, its space complexity is $O(\log n)$, which matches the lower bound of Theorem 2.3. This theorem could be applied repeatedly to obtain an algorithm for estimation of a set of n_i's.

Chapter 3

Algorithms in the Distributed Streaming Model

3.1 Introduction and Preliminaries

In this chapter, we focus on designing algorithms in a distributed variant of the basic streaming model and in its sliding window counterpart. To motivate the importance of distributed streaming model, we present the following two scenarios.

- A network operator needs to continuously monitor for the existence of global icebergs [68, 3] or elephant flows [42] that are coming across multiple servers in the network to prevent distributed denial of service attacks. These are enormous traffic flows intended to take down the network, which come from a single source and distributed itself to different destination serves in order to evade detection.

- A number of sensors are deployed in a vast field for sensing certain variables. A base station needs to continuously compute certain important statistics for past 24 hours based on the readings of the variables.

The following common properties emerge while studying the patterns of the above computations. Firstly, due to the distributed nature of computing devices, the computation could not be performed by bringing the data locally, as that would cost an impracticable communication overhead. Secondly, the statistics must be updated

quickly, with time or arrival of new items, since the computations are limited to a well-defined set of recent items. The distributed streaming model, and its sliding window counterpart were proposed as an abstraction to model computations in such scenarios. In this chapter, we study certain important computational problems over these models. We start by presenting these models.

3.1.1 The models

The distributed streaming model. In the distributed streaming model there are $(K+1)$ computational nodes: $\{N_1, N_2, \ldots, N_K, C\}$ where N_is are called *distributed* nodes and C is called the *coordinator* node. These nodes have to collectively compute a function f over a global stream of data items: $\{d_1, d_2, \ldots, d_t, \ldots, d_m\}$ which are distributed over N_is in an arbitrary manner. More precisely, at time t, the item d_t will be sent to the node N_j for some $1 \leq j \leq K$. At all times t, the coordinator should maintain an approximation of f over the set of items $\{d_1, d_2, \ldots, d_t\}$ seen so far from the global stream. In order to achieve this, each N_j can communicate with C through a bi-directional channel. An algorithm in this model must work for any ordering of the global input stream. The resources of interest are the total communication, total space usage over all nodes, and time to process each data item. The term *local stream* will be used to refer to the substream seen only at a particular node N_j. In this chapter we call this model, the *distributed streaming* model. Sometimes a subtle distinction is made between the cases when the computation must produce outputs continuously versus on demand or one-shot. The former variation is usually referred to as the distributed infinite window model, whereas the later variation is called the distributed streaming model.

The distributed sliding window model. In the sliding window variation of the

distributed infinite window model, there are the global stream and the set of $(K+1)$ nodes as before. But at any time t, the coordinator needs to maintain an approximation of the function f over the set of *most recent* W data items: $\{d_{t-W+1}, \ldots, d_t\}$. This set of items is known as the *active window* and W is known as the *window size*. As in most of the prior literature, we assume each data item comes with a unique (or unique modulo W) increasing time-stamp. As an example, the current UTC time could be the time-stamp. This model is referred to as the time-based distributed sliding window model. There is another variation called sequence-based distributed sliding window model, where no time-stamps are available. This model is harder to design algorithms on. Unless otherwise mentioned, by distributed sliding window we'll mean the time-based model. Schematic diagrams of these models appear in Figures 3.1 and 3.2.

3.1.2 Notations

Throughout this chapter, we use the abbreviations diw and dsw to mean distributed infinite window and distributed sliding window respectively. In certain contexts, we denote by $(t_1, t_2]$, the substream from (t_1+1) th through t_2 th element. We use: K to denote the number of distributed nodes, m the length of stream, W the length of window and k the number of medians/centers for clustering. $\langle c, s, t\rangle$ denotes the costs of our diw/dsw algorithm, where c is the communication complexity over any window of size W or over the length of the stream m as appropriate, s is the space complexity and t is the update time (possibly amortized). We assume storage of each data item, as well as a $O(\log m)$ bit counter, takes unit space.

For a function f, by a c-factor approximation we mean $\frac{f}{c} \leq \tilde{f} \leq c \cdot f$ and by $(1 \pm \epsilon)$ approximation we mean $(1-\epsilon)f \leq \tilde{f} \leq (1+\epsilon)f$. $\tilde{O}$ notation subsumes poly$(\log(\cdot))$ factors in the parameters of the problem such as $W, m, \tilde{f}, p, 1/\epsilon$. For a

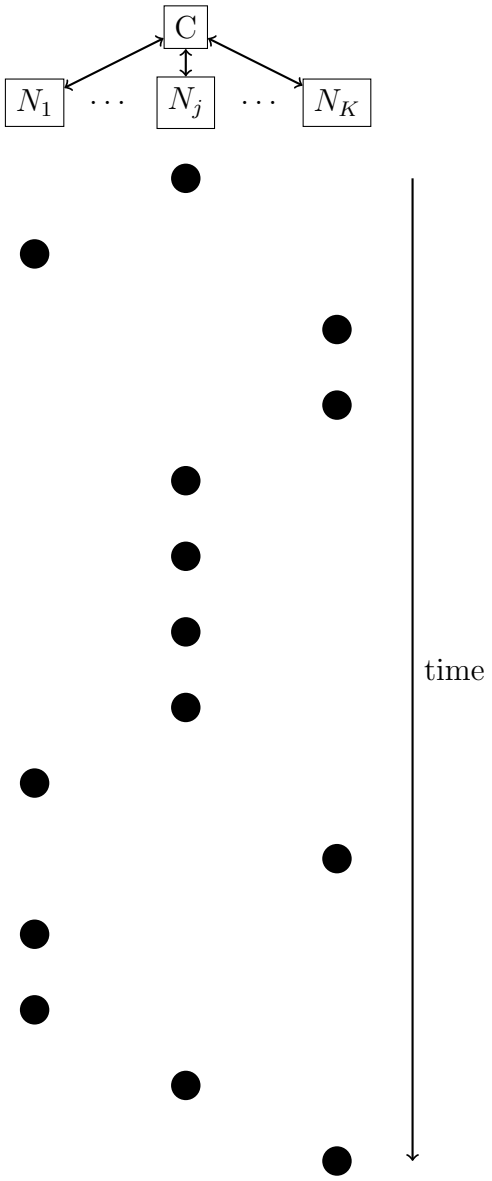

Figure 3.1: distributed streaming

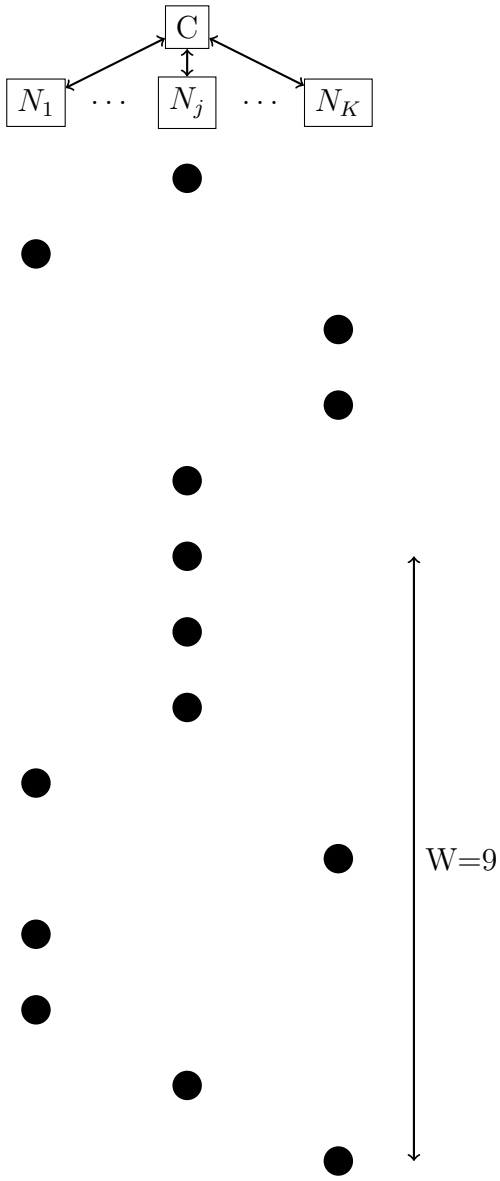

Figure 3.2: distributed sliding window

diw algorithm, with high probability means with probability at least $(1 - 1/\text{poly}(m))$ per m items. For a dsw algorithm, with high probability means with probability at least $(1 - 1/\text{poly}(W))$ per W items.

3.1.3 Previous work

Gibbons and Tirthapura [38] introduced the distributed streaming model in 2001. A formal study of computation over distributed streams was performed by Cormode, Muthukrishnan and Yi [27], who considered the problem of computing frequency moments in this model. Subsequently, efficient randomized algorithms for computing sum of bits, frequencies and ranks, were given in this model [43]. Woodruff and Zhang gave near-optimal bounds for computing frequency moments in [67]. Sampling algorithms over a distributed stream were given in [28, 62]. An efficient algorithm for computing k-center clustering, was given in [29]. Algorithms for computing entropy in this model were given in [4, 24]. We refer the reader to the survey of Cormode [26] on this topic.

In contrast, the sliding window variation of distributed streams have received only limited attention. Cormode et al. [28] gave the first efficient algorithm in dsw model, for the problem of maintaining a random sample. Later in [30] algorithms for counting the number of bits, quantiles, and heavy hitters were given. In [59], the authors gave an algorithm for computing approximate frequencies over a dsw.

We note that computing frequency moments and clustering, have received considerable attention in the *single stream sliding window* model [6, 14, 15, 17, 25].

3.1.4 Problem definitions for this chapter

Definition 3.1 (p-th Frequency moment)**.** *Given a set of items $\{1, 2, \ldots, n\}$ such that their frequencies are $\{f_1, f_2, \ldots, f_n\}$ respectively, their p-th frequency moment is defined as $F_p = \sum_{i=1}^{n} f_i^p$.*

Definition 3.2 (Metric k-median clustering problem)**.** *Given a set of points P from a metric space χ, output a set of k optimal medians $C^* = \arg\min_{C \subseteq \chi, |C| \leq k} \sum_{p \in P} \min_{c \in C}$*

$d(p, c)$ and the optimal clustering cost $\mathrm{OPT}_k = \sum_{p \in P} \min_{c \in C^*} d(p, c)$ *where d is the distance function of χ.*

Definition 3.3 (Metric k-center clustering problem)**.** *Given a set of points P from a metric space χ, output a set of k optimal centers* $C^* = \arg\min_{C \subseteq \chi, |C| \leq k} \max_{p \in P} \min_{c \in C}$ *$d(p, c)$ and the optimal clustering cost* $\mathrm{OPT}_k = \max_{p \in P} \min_{c \in C^*} d(p, c)$ *where d is the distance function of χ.*

For the above two clustering problems, an algorithm with approximation ratio of $r > 1$ means the clustering cost of the algorithm is in the range $[\mathrm{OPT}_k, r.\mathrm{OPT}_k]$. When the context is clear, we sometimes use OPT in place of OPT_k. Sometimes we write $\mathrm{OPT}(S)$ to emphasize the input set of points S.

3.1.5 Our results

We give algorithms for certain functions on dsw and diw model. We note that if every data item is communicated to the coordinator, the diw (dsw) model reduces to the non-distributed streaming (sliding window) model, with the cost $c = \Theta(m)$ (c=$\Theta(W)$). The challenge is to obtain better communication cost. Our strategy is to use space to reduce the communication cost, without impacting the (amortized) time by much. At a high level, our dsw algorithms are obtained using a reduction based approach. We give a framework that takes an efficient black-box diw algorithm $\mathcal{A}$, and gives an efficient dsw algorithm $\mathcal{B}$ that uses $\mathcal{A}$.

Such a framework was given in [17] for computing certain functions over a non-distributed stream. Our work gives an analogous framework for computing these functions over a distributed stream. We summarize our results on the dsw model, arising out of applying this framework for important functions such as frequency moments, k-median clustering, and k-center clustering in Table 3.1. We note that

these are the first algorithms given for the above functions in the dsw model. We also give the first algorithm for metric k-median clustering in the diw model. It remains open to either improve the cost of the above algorithms or to prove non-trivial lower bounds.

Table 3.1: Results on dsw and diw. All results, except the two on k-center, are randomized. All results are up to $\tilde{O}(1)$ factor. Results with [$*$] are our contribution.

Problem	Approx.	Result
F_2, diw	$(1 \pm \epsilon)$	$\langle \frac{K^2}{\epsilon^2} + \frac{K^{1.5}}{\epsilon^4}, \frac{K}{\epsilon^3}, \frac{1}{\epsilon^3} \rangle$ [27]
F_2, dsw [$*$]	$(1 \pm \epsilon)$	$\langle W^x(\frac{K^2}{\epsilon^4} + \frac{K^{1.5}}{\epsilon^8}), W^y + \frac{K}{\epsilon^6}, \frac{1}{\epsilon^6} \rangle$ [37]
F_p, diw, $p \geq 2$	$(1 \pm \epsilon)$	$\langle \frac{K^{p-1}}{\epsilon^{\Theta(p)}}, \frac{nK}{\epsilon}, \frac{1}{\epsilon^2} \rangle$ [67]
F_p, dsw, $p \geq 2$ [$*$]	$(1 \pm \epsilon)$	$\langle W^x \frac{K^{p-1}}{\epsilon^{\Theta(p^2)}}, W^y + \frac{nK}{\epsilon^{\Theta(p)}}, \frac{1}{\epsilon^{\Theta(p)}} \rangle$. [37]
k-center, diw	$O(1)$	$\langle kK, kK, k \rangle$ [29]
k-median, diw [$*$]	$O(1)$	$\langle kK, kK, k \rangle$ [36]
k-center, dsw [$*$]	$O(1)$	$\langle k^2K, k^2K + W, k^2 \rangle$ [37]
k-median, dsw [$*$]	$O(1)$	$\langle k^2K, k^2K + W, k^2 \rangle$ [37]
F_2, sequence based dsw [$*$]	$(1 \pm \epsilon)$	$\langle k\sqrt{n}/\epsilon^2, k + \sqrt{n}/\epsilon^2, 1 \rangle$ Section 3.5

3.1.6 Organization of the rest of the chapter

In Section 3.2 we give a distributed streaming algorithm for k-median clustering. In Section 3.3 we give a generic framework that takes a diw algorithm for certain functions and gives a dsw algorithm for it. In Section 3.4 we apply this framework to give dsw algorithms for clustering functions. We conclude this chapter with an algorithm for computing F_2 over a sequence based dsw model in Section 3.5.

3.2 New distributed streaming algorithm for k-median clustering

3.2.1 Brief recap of streaming algorithms for k-median clustering

Our starting point is the algorithm of Charikar et al. [23] for k-median clustering on insertion-only streams rooted on Meyerson's algorithm [53] for computing online facility location problem. The later problem is closely related to k-median clustering. Given a set of points P, the online facility problem asks to find a set $F \subseteq P$ of facilities to open such that $(f.|F| + Cost(P, F))$ is as small as possible. Here, $Cost(P, F)$ is the sum total of distances for each point in P to its closest facility from F and f is the cost of opening a single facility. The first part of the sum is known as the facility cost and the second part of the sum is known as the service cost.

Let us first go over Meyerson's algorithm briefly. It keeps in memory the set of already opened facilities F. As the new point p arrives, it computes the distance d from p to the closest facility from F. We open facility at p with probability $\min\{1, \frac{d}{f}\}$, where f is the cost of opening a single facility. It can be shown if we set f to be $\frac{L}{k(1+\log m)}$, where m is the number of points, and $0 < L \leq \text{OPT}$ is a parameter, the expected service cost becomes at most $(L + 4\text{OPT})$, where OPT is the optimal k-median clustering cost. The expected number of opened facilities becomes $k(1 + \log m)(1 + \frac{4\text{OPT}}{L})$. Moreover, it was shown by Braverman et al. [16] that these values are close to their expectations with high probability. For metric space, we restate this result in the following lemma.

Lemma 3.4 (Theorem 3.1 of [16])**.** *If we run the online facility location algorithm of [53] with* $f = \frac{L}{k(1+\log m)}$ *for some* $0 < L \leq \text{OPT}$*, on a set of* m *points, the service cost is at most* 6.2OPT *and the number of opened facilities is at most* $7k(1+\log m)\frac{\text{OPT}}{L}$

with probability at least $1 - \frac{1}{m^k}$. *Here* OPT *is the optimum* k*-median clustering cost.*

Given these facts, we can do a binary search to guess $L = \Theta(\text{OPT})$ so that we get a set of $O(k \log m)$ weighted points which represents the original points with additional k-media clustering cost of $O(\text{OPT})$. This idea was used in both the algorithms of [16] and [23]. We start with a small value of L and check that the clustering cost is at most αL and the number of opened facilities is at most $\beta k(1 + \log m)$ for some constants α, β. As soon as this condition is violated we increase the current value of L by a factor of γ for some $\gamma > 1$. With this new value of L, we start a new round and continue seeing all the elements of the stream again. But now the already seen elements would be replaced by the current set of at most $\beta k(1 + \log m)$ weighted centers produced. In this way, after all elements of the stream are seen, we can run an offline k-median algorithm on these weighted centers to get the final k clusters.

3.2.2 Distributed streaming algorithm for k-median clustering

Theorem 3.5. *There is a distributed streaming algorithm for computing a* $O(1)$*-approximate metric* k*-median clustering with high probability, with cost* $\langle O(kK \log^3 m), O(kK \log^2 m), O(k \log^2 m) \rangle$*, assuming* $\text{OPT}_k = \text{poly}(m)$.

Proof. Our approach is to adapt the streaming algorithm discussed in Section 3.2.1 into the distributed setting. We need to know the stream length m to set the facility cost $f = L/k(1 + \log m)$. If this is unknown, we use a set of $O(\log m)$ guesses for m: $1, 2, \ldots$, up to an appropriate large number. For each guess we run a separate algorithm. We also count the stream length as the elements arrive. At the end of stream, we output from the algorithm with the guess $\hat{m} \in [m, 2m)$, and discard other algorithms. Henceforth, we assume the knowledge of such a value $\hat{m} \in [m, 2m)$.

We run the Meyerson's algorithm in a distributed manner. Upon arrival of a point at a node, it opens a median at the arrived point with probability $\min\{1, d/f\}$, where d is its distance to the nearest median and f is the current facility cost, and maps it to the nearest median with the remaining probability $1 - \min\{1, d/f\}$. We keep track of the set of opened medians C, at every node and the coordinator. This can be achieved by the nodes broadcasting, whenever a median is opened in its local stream. The coordinator keeps track of the approximate service cost. Each node tracks the service cost due to its local stream and updates the coordinator whenever it crosses a multiple of 2. Thus the coordinator always have a value $\hat{s} \in (s/2, s]$, where s is the true service cost.

Our algorithm works in rounds. In the first round, $L = 1$, and $f = 1/k(1+\log \hat{m})$. A round is ended just before either $|C| > 14k(1 + \log \hat{m})$ or $\hat{s} > 13L$. During the change of round, the value of L is doubled and we need to replay the seen part of the stream with the doubled value of f. The set of currently opened medians, along with the weights of points mapped to each median, serves as a proxy to this seen part with a clustering cost at most $2\hat{s} < 26L$. The coordinator performs this replay with the updated value of f. At the end of replay, the coordinator broadcasts the set of medians opened from the replay to the nodes, and the next round is started.

We assume the success of the event stated in Lemma 3.4, at every point in the stream. When the guess of L reaches a value $(\mathrm{OPT}/2, \mathrm{OPT}]$, $|C| \leq 14k(1 + \log m) < 14k(1 + \log \hat{m})$, and $\hat{s} \leq s \leq 6.2\mathrm{OPT} \leq 12.4L$. Thus, the algorithm is guaranteed to terminate in $O(\log \mathrm{OPT}) = O(\log m)$ rounds. The failure probability is at most $1/\mathrm{poly}(m)$ from Lemma 3.4. When the algorithm terminates, the total clustering cost is at most $26(1 + 2 + 2^2 + \cdots + \mathrm{OPT}) = O(\mathrm{OPT})$. The final medians are obtained by running a $O(1)$-approximate offline k-median clustering algorithm on these $O(k \log m)$ weighted medians, with an additional clustering cost of $O(\mathrm{OPT})$. For the details, we

refer the reader to Section 2.3 of [23].

In each round, the communication for opening a median costs $O(K)$ and there are $O(k \log m)$ such medians per round. During the change of round, communicating the proxy stream costs $O(k \log m)$ and communicating the set of opened median from the replay, costs $O(kK \log m)$. Thus, over all the rounds, and over all the $O(\log m)$ guesses for the stream length, the total communication cost is $O(kK \log^3 m)$. The space complexity is determined by that for keeping the current set of $O(k \log m)$ medians at each node. The (amortized) update time is mainly spent to find the nearest median of the arrived point. □

3.3 Dsw algorithms for smooth functions

3.3.1 Smooth functions and smooth histograms

The notion of smooth functions was introduced by Braverman and Ostrovsky in [17]. The main property that a smooth function f should satisfy is the following continuity property: Consider f computed on a stream starting at two time points (or indices) i and j $(j > i)$. If f computed starting at i and f computed starting at j are within a constant factor of each other at a given point in time, then it will remain within a constant factor in the future.

Definition 3.6 ((α, β)-smooth function [17])**.** *A function f defined on a set χ of elements is called (α, β)-smooth, for some $0 < \beta < \alpha < 1$ if (1) f is non-decreasing and non-negative, (2) $f(A)$ is at most poly$(|A|)$, (3) $(1-\beta)f(A \cup B) \leq f(B) \implies (1-\alpha)f(A \cup B \cup C) \leq f(B \cup C)$ for any set $A, B, C \subseteq \chi$.*

Braverman and Ostrovsky show that for such smooth functions, it suffices to run an online algorithm to compute the function starting at a logarithmic number of care-

fully chosen indices to get constant approximation at any given window. These indices correspond to a constant factor decrease in the value of the function. The corresponding data structure is referred as smooth histogram. This resulted in a construction that translates any single stream infinite window algorithm to a single stream sliding window algorithm for smooth functions with comparable space complexity (up to log factors) as that of the infinite window algorithm.

Definition 3.7 (Approximate smooth histogram [17])**.** *Let f be an (α, β) smooth function. A smooth histogram for f is a data structure that consists of an increasing set of indices $[I_1, I_2, \ldots, I_L]$ over a sliding window of size W with the following properties.*

1. *For each $i = 1$ to L, there is an instance of a $(1 \pm \epsilon)$-approximating algorithm A, running for approximating $f(I_i, N)$, the value of f for $\epsilon \leq \beta/4$ over the set $\{d_{I_i}, d_{I_i+1}, \ldots, d_N\}$, where d_j is the data element at location j and d_N is the most recently arrived element.*

2. *I_1 is expired and I_2 is active or $I_1 = 0$.*

3. *For $i = 1$ to $t-2$, either 1) $(1-\alpha)f(I_i, N) \leq f(I_{i+1}, N)$ and $(1-\beta/2)f(I_i, N) > f(I_{i+2}, N)$ or 2) $(1 - \beta/2)f(I_i, N) > f(I_{i+1}, N)$ and $I_{i+1} = I_i + 1$.*

For a smooth function, the third point above ensures that consecutive indices are farthest apart but staying within at least $(1 - \alpha)$ factor (or immediate in stream and drops by $> (1 - \beta/2)$ factor). Note that $f(I_i, N)$ where I_i is the least index from the histogram that is contained in the current window, approximates the value of the function on the current window.

3.3.2 A transfer theorem for smooth functions

Our dsw algorithms take a reduction-based approach: using a diw algorithm for a smooth function f, we give a dsw algorithm for it. Informally, given a $\langle c, s, t\rangle$ cost diw algorithm for f, we give a dsw algorithm with cost $\langle \tilde{O}(\sqrt{W}c), \tilde{O}(\sqrt{W}+s), \tilde{O}(t)\rangle$ for it. More generally, we give a trade-off result, that takes roughly W^x times more communication, and W^{1-x} additional space. This result is presented below in full technical details, involving the smoothness and approximation parameters, and other natural parameters of the model and the problem.

Theorem 3.8. *Let f be an (α, β)-smooth function f for some $0 < \beta < \alpha < 1$. Let $0 < \epsilon < 1$ be such that $b = \frac{(1+\epsilon)^2}{(1-\epsilon)^2}(1-\beta) < 1$. Fix any $0 \le x, y \le 1, x+y = 1$. Suppose there is a diw algorithm $\mathcal{B}$ computes f over stream size at most m, up to approximation ratio $(1 \pm \epsilon)$, using cost $\langle c_{\mathcal{B}}(m,\epsilon), s_{\mathcal{B}}(m,\epsilon), t_{\mathcal{B}}(m,\epsilon)\rangle$. Then there is another algorithm that computes f over a dsw of size W up to approximation ratio $(1 \pm (\alpha+\epsilon))$ using cost $\langle L \cdot W^x \cdot (1+\log W)c_{\mathcal{B}}(W,\epsilon), (4L \cdot s_{\mathcal{B}}(W,\epsilon) + W^y), L(4+\log W)t_{\mathcal{B}}(W,\epsilon)\rangle$, where $L = ((\log \frac{f_{\max}}{f_{\min}(1-\epsilon)} / \log \frac{1}{b}) + 2)$, $f_{\max}$ = the maximum value of f over any window of size W, and $f_{\min}$ = smallest non-zero value of f. We assume storing each data element takes unit space.*

Note that the construction allows trade-offs between communication and space. In particular setting $x = y = 1/2$, we get that for any function f with $f_{\max}/f_{\min} = \text{poly}(W)$, a diw algorithm for f with cost $\langle c, s, t\rangle$ can be transformed to get a dsw algorithm with cost $\langle \tilde{O}(\sqrt{W}c), \tilde{O}(\sqrt{W}+s), \tilde{O}(t)\rangle$, where $\tilde{O}$ hides polylog factors.

We first give the algorithm and its proof for the case when $x = 0$ and $y = 1$, and then point out how to modify this algorithm to get the general algorithm. This algorithm has communication and time costs close to (up to polylog(W)) that of the diw algorithm, but uses $\Theta(W)$ space.

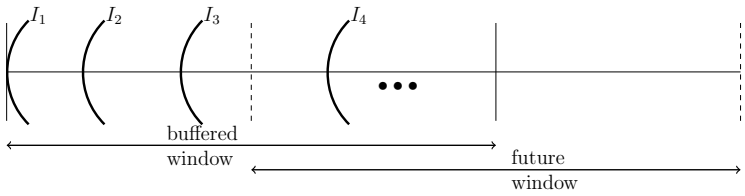

Figure 3.3: Online algorithms run from each index I_j (position indicated by '('). The one from I_3 can be used to approximate f over the future window. Thus, indices from the buffered window serves for next W elements.

Theorem 3.9. *Let f be an (α, β) smooth function for some $0 < \beta < \alpha < 1$. Let $0 < \epsilon < 1$ be any number such that $b = \frac{(1+\epsilon)^2}{(1-\epsilon)^2}(1-\beta) < 1$. Further assume f has a $(1 \pm \epsilon)$-approximate diw algorithm $\mathcal{B}$ over stream size at most m, with cost $\langle c_{\mathcal{B}}(m,\epsilon), s_{\mathcal{B}}(m,\epsilon), t_{\mathcal{B}}(m,\epsilon)\rangle$. Then, there is a dsw algorithm for computing f up to approximation ratio $(1 \pm (\alpha+\epsilon))$ with cost $\langle L(\log W + 1)c_{\mathcal{B}}(W,\epsilon), L \cdot s_{\mathcal{B}}(W,\epsilon) + W, L(\log W + 1)t_{\mathcal{B}}(W,\epsilon)\rangle$, where $L \leq (\log \frac{f_{\max}}{f_{\min}(1-\epsilon)} / \log \frac{1}{b}) + 2$, $f_{\max}$ = the maximum value of f over any window of size W, and $f_{\min}$ = smallest non-zero value of f. We assume storing each data element takes unit space.*

High level idea of the algorithm: Our general approach is to adapt the smooth histogram technique for sliding windows due to Braverman and Ostrovsky [17] to the distributed setting. Braverman and Ostrovsky showed that for smooth functions, if streaming algorithms are maintained from a small set of carefully chosen indices of the active window, one of these algorithms would approximate the value of the function over all active windows in near future. These indices correspond to a drop of the value of the function by some constant factor. Their algorithm has three main steps: when a new data item d arrives (1) start a new instance of the streaming algorithm $\mathcal{A}$ from this new item (2) update all running instances of $\mathcal{A}$ with d and (3) remove redundant indices and the corresponding instances of $\mathcal{A}$.

There are technical challenges to translate the smooth histogram technique to the

distributed setting. The main obstacle is the following: in the single stream case, for each newly arrived element, an instance of $\mathcal{A}$ is started from its index. Most of them are removed at some later point in the future so that at all times only a logarithmic number of indices are kept. In distributed setting, if one has to start instances of $\mathcal{A}$ each time a new element arrives, it will cost $\Omega(W)$ bits of communication (from the distributed node where the item arrives to the coordinator).

In order to reduce the communication cost we use the following approach. Instead of continuously building the histogram, we observe that it is enough to create it once per W items. Once built, this histogram will continue to work till the arrival of next W^{th} item due to smoothness of the function. In other words, steps (1) and (3) in the previous discussion could be dropped except once per W items. This keeps the asymptotic time and communication cost small. We split the entire stream into static windows of size W: $(0, W], (W, 2W], \ldots, (aW, aW + W], \ldots$ and we build the smooth histogram only when the current window coincides with one of the static windows. This is done by buffering the entire expiring window using $O(W)$ space (see Figure 3.3 for illustration). The indices, which correspond to a drop of some constant factor for f, are obtained by performing binary searches on the buffered static window. This will introduce further communication cost and time for about $O(\log^2 W)$ many instances of the online algorithm.

Proof. (*Of Theorem 3.9*). A high-level pseudocode of the algorithms is given in Algorithm 4. We split the entire stream into static windows of size W: $(0, W], (W, 2W], \ldots, (aW, aW + W], \ldots$ and maintain a smooth histogram over exactly one of them. If the current time t satisfies $aW - W < t < aW$, the smooth histogram from the static window $(aW - 2W, aW - W]$ can be used to approximate $f([t - W + 1, t])$. We recall, the smooth histogram maintains online algorithms from a set of appropriately

chosen indices guarantees that any two consecutive indices are either consecutive or are $(1-\alpha)$-factor close in f. If $I_1 \leq (t-W+1) < I_2$ are the two unique consecutive pair of indices enclosing $(t-W+1)$, the value of the online algorithm from the index I_1 can be used to approximate $f([t-W+1,t])$ up to $(1 \pm (\alpha+\epsilon))$-factor. We buffer the next static window $(aW, aW+W]$ locally (at the distributed nodes as they arrive) and build the smooth histogram from it at time $(aW+W)$. We describe how to build this in detail in the following claim.

Algorithm 4: High level dsw algorithm for smooth functions

Input: A stream of data: $\langle d_1, d_2, \ldots, d_N \rangle$
Output: Approximate value of $f((N-W, N])$

```
while Not the end of stream do
    Let d_N be the newly arrived item;
    if N = aW + W then
        Delete the current smooth histogram H((aW − W, aW]);
        H((aW, aW + W]) ← Build a new smooth histogram for (aW, aW + W];
        Output the value of online algorithm over (aW, aW + W];
    else
        if aW < N < aW + W then
            Update the online algorithm from each index of H((aW − W, aW])
            with d_N;
            Buffer d_N at the node where it arrived;
            I_j, I_{j+1} be the immediate indices such that I_j ≤ N − W + 1 < I_{j+1};
            Output the value of the online algorithm started from index I_j;
        end
    end
end
```

Claim 3.10. *Suppose we have stored a static window* $[L, R] = (aW, aW + W]$ *locally. A smooth histogram* $\mathcal{H}$ *for* $[L, R]$ *can be built using cost* $\langle L \log W \cdot c_{\mathcal{B}}(W, \epsilon), (s_{\mathcal{B}}(W, \epsilon) + W), L \log W \cdot t_{\mathcal{B}}(W, \epsilon)\rangle$*, where the number of indices in* $\mathcal{H}$ *is* $L \leq (\log \frac{f_{\max}}{f_{\min}(1-\epsilon)} / \log \frac{1}{b}) + 2$.

Proof. We find the indices from the buffered window $[L, R]$ by binary search and by running the online algorithm $\mathcal{B}$ on the buffered set of items[1]. The first index is always L. The last index is always the index R. We denote by $\tilde{f}$, a $(1 \pm \epsilon)$ factor approximation for f, found using $\mathcal{B}$. Then, the second index is created at a time-stamp t, such that,

$$\tilde{f}([t, R]) \geq (1 - \beta)\frac{(1 + \epsilon)}{(1 - \epsilon)}\tilde{f}([L, R]). \tag{3.1}$$

This ensures, $f([t, R]) \geq (1 - \beta)f([L, R])$, as desired for smooth histogram. In fact, we try to find such a t as far as possible in the window to minimize the number of indices. We tag a time-stamp 'yes' if it satisfies Equation (3.1) and 'no' otherwise. We set two variables $l = L$ and $r = R$ and maintain the invariant that l has 'yes' tag and r has 'no' tag. We next check whether $mid = (l + r)/2$ has 'yes' tag or 'no' tag and update l or r appropriately to maintain the invariant. Then, in $\log W$ steps, we will be able to get hold of a t, such that, t has 'yes' tag but $(t + 1)$ has 'no' tag. This is our next index. We find subsequent indices in a similar manner.

[1] There are some details for running $\mathcal{B}$ on the buffered items. For a general function, assume, all the nodes have some global knowledge of time. Then, a fixed time interval of sufficient length can be allotted for processing each data item. Thus, for example, it may be agreed upon that the k^{th} data item d in the current window will be processed during time interval $(t, t + \Delta \cdot k)$ where Δ is at least as large as the update time of the algorithm and t is the time of arrival of the oldest element of the current window. During this interval, whichever node has received d, will process it. For a permutation-invariant function such as F_p and clustering, the nodes can take turns and run the online algorithm over the desired sub-window of the local stream, to compute the function over any sub-window of the global stream.

Notice that, at the $(t+1)$-st item, the value of $\tilde{f}$ drops at least by factor $(1-\beta)\frac{(1+\epsilon)}{(1-\epsilon)}$ (Recall, if the indices are consecutive, drop could be larger). This implies, f drops at least by factor $b = (1-\beta)\frac{(1+\epsilon)^2}{(1-\epsilon)^2} < 1$. Suppose there are L indices in total. Then, After crossing the $(L-1)$-st index, the value of $\tilde{f}$ is at most $f_{\max} b^{(L-2)}$, where $f_{\max}$ is the maximum value of f over any window. Moreover, assuming the least non-zero value of f is $f_{\min}$, $f_{\max} b^{(L-2)} \geq f_{\min}(1-\epsilon)$. Hence $L \leq (\log \frac{f_{\max}}{f_{\min}(1-\epsilon)} / \log \frac{1}{b}) + 2$. We denote by I this set of indices. This concludes the updating of the smooth histogram. While finding the indices, at most $L \log W$ instance of $\mathcal{B}$ are run for at most W units of time. This can be achieved using $L \log W \cdot c_{\mathcal{B}}(W, \epsilon)$ total communication and $LW \log W \cdot t_{\mathcal{B}}(W, \epsilon)$ total time (i.e. amortized update time $L \log W \cdot t_{\mathcal{B}}(W, \epsilon)$ per item). During the binary search, we need space for running at most a single instance of $\mathcal{B}$ at any point in time, and the space is reused. We also need space for buffering the current window. So, the space complexity for this part is $(s_{\mathcal{B}}(W, \epsilon) + W)$. □

Afterwards, we maintain $\mathcal{B}$ from each of the indices and update them with newly arrived items. For any time till the arrival of next $(W-1)$ items, let $f_1 > f_2$ be the value of f at the two enclosing indices of the current window. Let $f_{current}$ be the value of f over the current window. From the properties of smooth histogram, either $f_{current} = f_1$, or $(1-\alpha)f_1 < f_2 \leq f_{current} \leq f_1$. Moreover, the value of online algorithm at the former index holds a value $\tilde{f}_1$, such that, $(1-\epsilon)f_1 \leq \tilde{f}_1 \leq (1+\epsilon)f_1$. Hence, $(1-\epsilon)f_{current} \leq \tilde{f}_1 \leq \frac{(1+\epsilon)}{(1-\alpha)} f_{current}$. This is close to $(1 \pm (\alpha+\epsilon))$-approximation for small α and ϵ. We also need to continue running $\mathcal{B}$ from each index for W units of time. This costs at most $L \cdot c_{\mathcal{B}}(W, \epsilon)$ communication, $L \cdot t_{\mathcal{B}}(W, \epsilon)$ update time per item and $L \cdot s_{\mathcal{B}}(W, \epsilon)$ space in total. □

Proof. (*of Theorem 3.8*). The proof follows closely from that of Theorem 3.9. In this case, we break the current window of size W into W^x blocks, each of size W^y,

such that $W = W^x \cdot W^y$, for some $0 < x, y < 1, x + y = 1$. We rebuild the smooth histogram per arrival of W^x elements in the combined stream. Then, the nodes need to store at most W^x items. As before, total number of indices within each block, $L_1 \leq L = ((\log \frac{f_{\max}}{f_{\min}(1-\epsilon)} / \log \frac{1}{b}) + 2)$. As in the algorithm of Theorem 3.9, finding these indices is done by binary search, using at most $L_1 \log W$ calls to $\mathcal{B}$. In total, this can be done with $L_1 \log W c_{\mathcal{B}}(W, \epsilon)$ total communication per block (i.e $L_1 \cdot W^x \cdot \log W c_{\mathcal{B}}(W, \epsilon)$ in total), $(s_{\mathcal{B}}(W, \epsilon) + W^y)$ space (since space is reused during binary-search) and $L_1 \log W W^y t_{\mathcal{B}}(W, \epsilon)$ total time per block (i.e. amortized $L_1 \log W t_{\mathcal{B}}(W, \epsilon)$ time per item).

Subsequently, we need to maintain $L_1 \cdot W^x$ online algorithms from each of the indices within the current window of size W. But we can do better by removing unnecessary ones while introducing indices from a new block. We arrange the combined indices in decreasing order of arrival. Then, for each index i, we look for the subsequent index j where the current value of the online algorithm drops by factor b for the first time. We remove all indices strictly between i and $(j-1)$ if there are any. We repeat this removal procedure until no more indices can be removed in such a manner. Then, starting from each index, at the next to next index, value of online algorithm drops at least by factor b. After merging, there are $L_2 \leq (2(\log \frac{f_{\max}}{f_{\min}(1-\epsilon)} / \log \frac{1}{b}) + 2) \leq 2L$ such indices at any point in time. Moreover, by previous discussion, every next index is either the subsequent item, or within a factor b from the previous index. This entire removal takes time linear in the set of the indices to be merged, i.e. at most $10(\log \frac{f_{\max}}{f_{\min}(1-\epsilon)} / \log \frac{1}{b})$ time per block. So, we ignore this while computing the update time per item. Note that, during this removal, no further communication or space is required. This concludes the indices removal procedure. Then, online algorithms from each of the indices run for $\geq W^y$ and $\leq W$ time. So the total communication cost is at most $L_1 \cdot W^x c_{\mathcal{B}}(W, \epsilon)$. The

space complexity for running the online algorithms is at most $2L_2 \cdot s_{\mathcal{B}}(W, \epsilon)$. Using the indices removal procedure, we improve the update time to $2L_2 \cdot t_{\mathcal{B}}(W, \epsilon)$ per arrival, for updating each of the current instances of $\mathcal{B}$. □

3.3.3 Better and simpler algorithm for symmetric smooth functions

For symmetric smooth functions we get a simpler algorithm with slightly better cost. In particular, the cost of the algorithm will be $\langle (L+1) \cdot c_{\mathcal{B}}(W, \epsilon) \cdot W^x, 4L \cdot s_{\mathcal{B}}(W, \epsilon) + W^y, 4(L+1)t_{\mathcal{B}}(W, \epsilon) \rangle$.

We call a function symmetric if its value is invariant to the permutation of its arguments. For symmetric functions, we create the indices of the smooth histogram by making a single backward pass (i.e. from the most recent to the least recent item in the window) of the distributed online algorithm on the buffered window $[aW + 1, aW + W]$. Let $\tilde{f}(A)$ be the value of this algorithm, which is within $(1 \pm \epsilon)$-factor of $f(A)$. We create the last index of the smooth histogram at $(aW + W)$. Recursively assume the previous index we created was at t_1. During the backward pass, suppose at the time-stamp $(t_2 - 1) \leq t_1$, the value $\tilde{f}([t_2 - 1, aW + W])$ is at least $\frac{1}{b} \cdot \tilde{f}([t_1, aW + W]) = \frac{1}{(1-\beta)} \frac{(1-\epsilon)}{(1+\epsilon)} \cdot \tilde{f}([t_1, aW + W])$ for the first time. If this happens at $t_2 - 1 = t_1 - 1$, we create an index at $(t_1 - 1)$. Otherwise, we create at t_2, which satisfies $\tilde{f}([t_2, aW + W]) < \frac{1}{b} \tilde{f}([t_1, aW + W])$. This implies, $f([t_1, aW + W]) \geq (1 - \beta) f([t_2, aW + W])$, ensuring the smoothness condition between the consecutive indices t_1 and t_2. We find all the L indices in similar manner, where $L = ((\log \frac{f_{\max}}{f_{\min}(1-\epsilon)} / \log \frac{1}{b}) + 2)$, $f_{\max}$ = the maximum value of f over any window of size W, and $f_{\min}$ = smallest non-zero value of f. This improves the cost of Theorem 3.9 to $\langle (L+1)c_{\mathcal{B}}(W, \epsilon), (L+1) \cdot s_{\mathcal{B}}(W, \epsilon) + W, (L+1)t_{\mathcal{B}}(W, \epsilon) \rangle$. We can shave off a $\log W$ factor from the costs of Theorem 3.8 using a similar simpler algorithm. Note that, we crucially use the symmetric nature of the function in the use of the backward online

algorithm. The F_p and geometric mean are symmetric smooth functions, whereas, the function 'length of longest increasing subsequence' is asymmetric smooth [17].

3.3.4 Dsw algorithms for frequency moments

In this section we apply the transfer theorem for computing F_p over a dsw. We first recall a result that shows F_p is smooth.

Theorem 3.11 (Lemma 5 of [17]). *Fix any $0 < \epsilon < 1$. For $p \leq 1, F_p$ is (ϵ, ϵ)-smooth function. For $p > 1, F_p$ is $(\epsilon, \frac{\epsilon^p}{p^p})$-smooth function.*

We'll be using the following diw algorithm for F_p, due to Woodruff and Zhang. We'll be using the simpler transfer framework discussed in Section 3.3.3 to get the dsw algorithm. Note that, this framework requires $\frac{1}{(1-\beta)}\frac{(1-\epsilon)}{(1+\epsilon)} > 1$, where (α, β) is the smoothness parameter of F_p and $(1 \pm \epsilon)$ is the approximation ratio of the diw algorithm. Thus, to achieve $\alpha = \epsilon$ we need $\beta = \epsilon^p/p^p$, requiring the approximation ratio of the diw algorithm to be something like $\epsilon^p/3p^p$.

Theorem 3.12 (Follows from Theorem 8 of [67]). *For any $0 < \epsilon < 1$, there is an algorithm that continuously computes F_p for any constant $p \geq 1$, over universe $[n]$ over a distributed stream of length at most W up to approximation $(1 \pm \epsilon)$ with high probability using cost $\langle \tilde{O}(\frac{K^{p-1}}{\epsilon^{\Theta(p)}}), \tilde{O}(\frac{nK}{\epsilon}), \tilde{O}(\frac{1}{\epsilon^2})\rangle$.*

Corollary 3.13. *Fix any $0 \leq x, y \leq 1, x + y = 1$. For any constant p, there is an algorithm that continuously computes F_p over a time based dsw of width W up to approximation ratio $(1 \pm \epsilon)$ with high probability using cost $\langle \tilde{O}(W^x \frac{K^{p-1}}{\epsilon^{\Theta(p^2)}}), \tilde{O}(W^y + \frac{nK}{\epsilon^{\Theta(p)}}), \tilde{O}(\frac{1}{\epsilon^{\Theta(p)}})\rangle$.*

In particular for F_2, we chose to work with the diw algorithm of [27], whose communication cost has a smaller dependence on ϵ.

Theorem 3.14 (Follows from Theorem 6.1 of [27]). *For any $0 < \epsilon < 1$, there is an algorithm that continuously computes F_2 over universe n over a distributed stream of length at most W up to approximation $(1 \pm \epsilon)$ with high probability using cost $\langle \tilde{O}(\frac{K^2}{\epsilon^2} + \frac{K^{1.5}}{\epsilon^4}), \tilde{O}(\frac{K}{\epsilon^3}), \tilde{O}(\frac{1}{\epsilon^3}) \rangle$.*

Corollary 3.15. *Fix any $0 \leq x, y \leq 1, x + y = 1$. There is an algorithm that continuously computes F_2 over a time based dsw of width W up to approximation ratio $(1 \pm \epsilon)$ with high probability using cost $\langle \tilde{O}(W^x(\frac{K^2}{\epsilon^4} + \frac{K^{1.5}}{\epsilon^8})), \tilde{O}(W^y + \frac{K}{\epsilon^6}), \tilde{O}(\frac{1}{\epsilon^6}) \rangle$*

3.4 Dsw algorithms for clustering

In this section, we give dsw algorithms for metric k-median and k-center clustering problems. These clustering costs are not exactly smooth functions. So, we could not use the transfer framework of Section 3.3. Nevertheless we show how to maintain a smooth histogram for these two problems, using additional observations. Our dsw algorithms take $O(W)$ space, and we could not give a trade-off result as in the case of smooth functions. The non-smoothness of these clustering costs was also observed in [15]. We give a slightly general proof.

Lemma 3.16. *k-center and k-median clustering cost functions are not smooth.*

Proof. Consider two subsets of points A, B such that $\text{OPT}(A \cup B) \leq \beta \text{OPT}(B)$ for some β. We set up a subset C such that $\text{OPT}(A \cup B \cup C) > \alpha \text{OPT}(B \cup C)$ for any α. The idea is to enforce at least one center from the set A by choosing a tight group of points and also placing these points far from B and C. This ensures that both in $\text{OPT}(A \cup B)$ and $\text{OPT}(A \cup B \cup C)$, one is forced to use at most $(k-1)$ centers from B part and $(B \cup C)$ part respectively. Now it remains to create a configuration

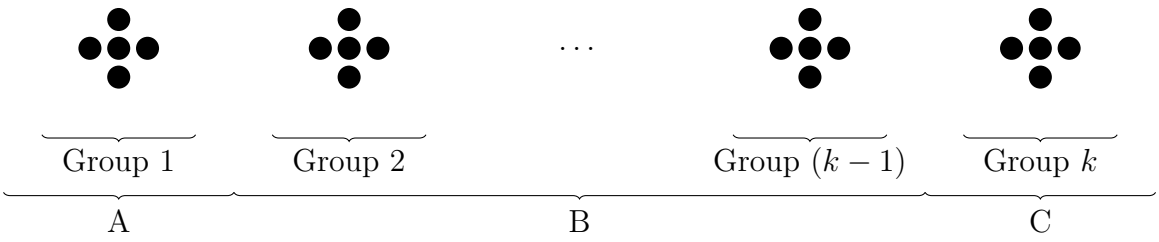

Figure 3.4: k-median and k-center costs are not smooth for A, B, C as above

of B and C such that $\mathrm{OPT}_{k-1}(B)$ and $\mathrm{OPT}_k(B)$ are close but $\mathrm{OPT}_{k-1}(B \cup C)$ and $\mathrm{OPT}_k(B \cup C)$ are not. We show such a counterexample in Figure 3.4. □

For convenience, only for the current Section 3.4, we slightly change the notation of Definition 3.6 for a smooth function as follows.

Definition 3.17 ((α, β)-smooth function)**.** *A function f defined on a set χ of elements is called (α, β)-smooth, for some $1 < \beta < \alpha$ if (1) f is non-decreasing and non-negative, (2) $f(A)$ is at most poly$(|A|)$, (3) $\frac{1}{\beta} f(A \cup B) \leq f(B) \implies \frac{1}{\alpha} f(A \cup B \cup C) \leq f(B \cup C)$ for any set $A, B, C \subseteq \chi$.*

We denote by $Cost(P, O)$ the clustering cost for a set of points P, when O is the set of k medians/centers.

3.4.1 Dsw algorithm for k-median clustering

We start with the following crucial Lemma from [15], which shows the k-median cost behaves like a smooth function, if an additional condition is satisfied.

Lemma 3.18 (Lemma 3.1 of [15] restated)**.** *For any distinct sets of points $A, B, C \subseteq \chi$ from some metric space χ, $\mathrm{OPT}(A \cup B) \leq \gamma \mathrm{OPT}(B) \implies \mathrm{OPT}(A \cup B \cup C) \leq (2 + r\gamma)\mathrm{OPT}(B \cup C)$ for any $r, \gamma \geq 1$, provided the following property holds for the sets A, B:*

- *There exists a k-median clustering $t : (A \cup B) \to F$ up to approximation ratio r such that $|t^{-1}(f) \cap A| \leq |t^{-1}(f) \cap B|$ for each median $f \in F$.*

We use the diw algorithm of Theorem 3.5 for k-median clustering, given in Section 3.2.

Theorem 3.19. *There is a dsw algorithm for $O(1)$-approximate metric k-median with success probability $(1 - \frac{1}{\text{poly}(W)})$ per W items, and with cost $\langle O(k^2 K \log^5 W), O(k^2 K \log^4 W + W), O(k^2 \log^4 W)\rangle$ assuming $\text{OPT} = \text{poly}(W)$.*

Proof. We split the stream into static windows of the form: $[aW + 1, aW + W]$ and store this window locally. At time $(aW + W)$, we need to rebuild a smooth histogram. We use the idea of binary-search described in Claim 3.10 to get a set of indices from the stored window such that crossing each index drops the clustering cost by a γ factor, where γ is a constant satisfying $\gamma > \lambda$, and λ is the approximation factor of the distributed streaming algorithm $\mathcal{A}$ from Theorem 3.5. These indices are referred to as the 'outer indices'. There are $O(\log W)$ of them. For each outer index, we also record the set of k medians produces by $\mathcal{A}$. Let the indices be $I = \{I_1 < I_2 < \cdots < I_L\}$ and the corresponding sets of k medians be $C = \{C_1, C_2, \ldots, C_L\}$, where each $C_i = \{c_{i1}, c_{i2}, \ldots, c_{ik}\}$. We communicate I and C to each node. We claim that for any $I_i \leq t < I_{i+1}$, C_i is a $\gamma\lambda$-approximate set of k-medians for $[t, aW + W]$. This is because, $Cost([t, aW + W], C_i) \leq Cost([I_i, aW + W], C_i) \leq \lambda \cdot \text{OPT}([I_i, aW + W]) \leq \gamma\lambda \cdot \text{OPT}([I_{i+1}, aW + W]) \leq \gamma\lambda \cdot \text{OPT}([t, aW + W])$ (Using monotonicity of OPT and smoothness).

We also need to ensure the additional property from Lemma 3.18. We ensure this by keeping a set of 'inner' indices between each pair of outer indices. We describe below how to find the inner indices between I_1 and I_2. Other inner indices can be found accordingly. For any $i = 1$ to L, and for any set $S = [t, aW+W] \subseteq [I_i, aW+W]$,

let n_{ij}^S denote the number of points from S which map to the median c_{ij}, in the clustering $[I_i, aW+W] \to C_i$. Let $\tilde{n}_{ij}^S$ denote a $(1 \pm \frac{1}{10})$ approximation of n_{ij}^S. We first assume the coordinator can compute $\tilde{n}_{ij}^S$ for any $S = [t, aW+W] \subseteq [I_i, aW+W]$. We defer the description of how to compute this in the following paragraph. Let J_1 be the first (earliest in window) inner index between I_1 and I_2. We will create the index J_1 at the farthest time-stamp $I_1 < t \leq I_2$, such that

$$\forall j = 1 : k, \tilde{n}_{1j}^{[I_1, aW+W]} \leq \frac{18}{11} \tilde{n}_{1j}^{[t, aW+W]} \tag{3.2}$$

The latter condition ensures, for each median $c_{1j} \in C_1$, $n_{1j}^{[I_1, aW+W]} \leq 2 \cdot n_{1j}^{[t, aW+W]}$, equivalently $n_{1j}^{[I_1, t-1]} \leq n_{1j}^{[t, aW+W]}$, as demanded in the additional property from Lemma 3.18. Such a farthest t satisfying Equation 3.2 is found by using binary search. Notice that $t = l = I_1$ is always satisfied. We first guess $t = r = I_2$. If this t satisfies Equation 3.2, we already have an index at I_2 and we don't need to keep any inner index. If not, we next guess $t = \lceil \frac{l+r}{2} \rceil$ and if this t satisfies, we change $l = \lceil \frac{l+r}{2} \rceil$, otherwise, we change $r = \lceil \frac{l+r}{2} \rceil$. In this way, we preserve the invariant that l satisfies Equation 3.2 but r does not. In $O(\log W)$ steps, we will get a t^*, such that t^* satisfies but (t^*+1) does not. We set $J_1 = t^*$. We find the next inner index J_2 similarly using the same clustering C_1 and at the farthest time-stamp t, such that, $\forall j = 1 : k, \tilde{n}_{1j}^{[J_1, aW+W]} \leq \frac{18}{11} \tilde{n}_{1j}^{[t, aW+W]}$ holds. Note that, the set C_1 is a $\gamma\lambda$ approximate median for any $I_1 \leq t < I_2$, from previous discussion. So, the additional property from Lemma 3.18 holds at indices J_1 and J_2, with respect to the clustering C_1 and $r = \gamma\lambda$. Let C_1' and C_2' be the λ-approximate clusterings for J_1 and J_2 respectively. Hence from Lemma 3.18 at any later time $(t' + W - 1)$, such that $J_1 \leq t' < J_2$, $Cost([t, t+W-1], C_1') \leq Cost([J_1, t+W-1], C_1') \leq \lambda \cdot \mathrm{OPT}([J_1, t+W-1]) \leq \lambda(2+\gamma^2\lambda) \cdot \mathrm{OPT}([I_{i+1}, t+W-1]) \leq \lambda(2+\gamma^2\lambda) \cdot \mathrm{OPT}([t, t+W-1])$ (Using mono-

tonicity of OPT and smoothness). So, the final approximation is $\lambda(2+\gamma^2\lambda)$. We find subsequent inner indices in similar manner. Note that, after crossing each inner index, $\tilde{n}_{ij}$ for some j reduces by a factor $\frac{18}{11}$. Since there are at most k medians and W items, the total number of inner indices between I_1 and I_2 is at most $O(k \log W)$. Also note that, for checking Equation 3.2, the coordinator needs $\tilde{n}_{ij}^{[t,aW+W]}$ values, for various values of t, and j, which we obtain as follows.

Each node makes a backward pass over its local data. During this pass, for each i, it maps each point $p \in [I_i, aW+W]$ to c_{ij^*}, where $j^* = \arg\min_j d(p, c_{ij})$, i.e. c_{ij^*} is the closest of the medians from C_i. It also keeps a counter n_{ij} for each c_{ij}, which increments for each new point mapping to c_{ij}. We then record the time-points where n_{ij} increases by $(1+\frac{1}{10})$-factor, i.e crosses $\{\frac{11}{10}, \frac{11^2}{10^2}, \ldots, W\}$ for the first time. We call this set of time-points as H_{ij}. Each node z sends such H_{ij}^z, for each i, j to the coordinator. Note that, $n_{ij}^{[t,aW+W]} = \sum_z n_{ij}^{[t,aW+W]_z}$, where $[t, aW+W]_z$ denotes items from $[t, aW+W]$ that appear at node z. Using H_{ij}^z, coordinator can approximate $n_{ij}^{[t,aW+W]_z}$ up to $(1 \pm \frac{1}{10})$ factor, for any z. Taking sum over all z, it can approximate $n_{ij}^{[t,aW+W]}$ for any t, up to $(1 \pm \frac{1}{10})$ factor, as required.

The total number of indices are $O(k \log^2 W)$, from each of which a distributed streaming algorithm is run with cost $\langle O(kK \log^3 W), O(kK \log^2 W), O(k \log^2 W)\rangle$. The total cost due to this is $\langle O(k^2 K \log^5 W), O(k^2 K \log^4 W), O(k^2 \log^4 W)\rangle$, which dominates the total cost of the dsw algorithm. We also need $O(W)$ space while rebuilding the histogram once per W points. Since we run at most $O(k \log^2 W)$ online algorithms, success probability per W items is still $(1 - \frac{1}{\text{poly}(W)})$. □

3.4.2 Dsw algorithm for k-center clustering

We start with a Lemma analogous to Lemma 3.18 for smoothness of k-center clustering. The additional property is more relaxed in the case of k-center clustering. To

the best of our knowledge, this result was not known before.

Lemma 3.20. *For any distinct sets of points $A, B, C \subseteq \chi$ from some metric space χ, $\text{OPT}(A \cup B) \leq \gamma \text{OPT}(B) \implies \text{OPT}(A \cup B \cup C) \leq (1 + 2r\gamma)\text{OPT}(B \cup C)$ for any $r, \gamma \geq 1$, provided the following property holds for the sets A, B:*

- There is a k-center clustering $t : (A \cup B) \to F$, up to approximation ratio r, such that, for each center $f \in F, |t^{-1}(f) \cap A| > 0 \implies |t^{-1}(f) \cap B| > 0$.

Proof. Let O be the optimal set of centers for $B \cup C$. We will map each element $a \in A$ to some point in O. Let O' be the r-approximate set of centers for $A \cup B$, which satisfies the above property. Let $o' \in O'$ be the center to which $a \in A$ maps to. By assumption some $b \in B$ also gets mapped to o'. Finally, let $o \in O$ be the center to which b maps to. We will map a to o. Then by definition, $\max(d(a, o'), d(b, o')) \leq r \cdot \text{OPT}(A \cup B)$ and $d(b, o) \leq \text{OPT}(B \cup C)$. Then, by triangle inequality, $d(a, o) \leq (d(a, o') + d(b, o') + d(b, o)) \leq (2r \cdot \text{OPT}(A \cup B) + \text{OPT}(B \cup C))$

$$
\begin{aligned}
\text{OPT}(A \cup B \cup C) &\leq Cost(A \cup B \cup C, O) \\
&\leq \max\{\text{OPT}(B \cup C), Cost(A, O)\} \\
&\leq \max\{\text{OPT}(B \cup C), (2r \cdot \text{OPT}(A \cup B) + \text{OPT}(B \cup C))\} \\
&\leq (2r \cdot \text{OPT}(A \cup B) + \text{OPT}(B \cup C)) \\
&\leq (2r\gamma \cdot \text{OPT}(B) + \text{OPT}(B \cup C)) && \text{(Given)} \\
&\leq (1 + 2r\gamma)\text{OPT}(B \cup C) && \text{(Using monotonicity of OPT)}
\end{aligned}
$$

□

We use the following distributed streaming algorithm for k-center clustering.

Theorem 3.21 (Theorem 6 of [29] restated)**.** *There is a deterministic distributed streaming algorithm for* 3*-approximate metric* k*-center with cost* $\langle O(kK \log W), O(kK \log W), O(k)\rangle$*, assuming* $\mathrm{OPT}_k = \mathrm{poly}(W)$.

Our dsw algorithm for k-center clustering closely follows that for k-median clustering given in Section 3.4.1. We create a set of 'outer' indices corresponding to a constant factor drop of the cost of the distributed streaming algorithm. We also introduce a set of 'inner' indices between each pair of outer indices to satisfy the additional property of Lemma 3.20. These inner indices are created at a point t, such that there exists a center to which no item from part $(t, aW + W]$ maps. Since, there are at most k centers, at most k inner indices are possible between each pair of outer indices. Hence the total number of indices is $O(k \log W)$. We present the theorem for k-center clustering and skip the proof since it closely follows that of Theorem 3.19.

Theorem 3.22. *There is a deterministic dsw algorithm for* $O(1)$*-approximate metric* k*-center with cost* $\langle O(k^2 K \log^2 W), O(k^2 K \log^2 W + W), O(k^2 \log W)\rangle$ *assuming* $\mathrm{OPT}_k = \mathrm{poly}(W)$.

3.5 Computing frequency moments over a sequence-based dsw

In Section 3.1.1, while defining the dsw model, we made an assumption that every item of the stream comes with a unique (modulo W) time-stamp. This assumption is crucial for designing our algorithms, since in dsw model, the coordinator is not directly aware of the arrival of a new item. Thus most of the dsw algorithms in the literature works with this assumption. This model is specifically called the time-based dsw model. The harder model, where the computation is limited to the most recent

W items, without any time-stamp information, is known as the sequence-based dsw model. To the best of our knowledge only sampling was known in this later model [28] before this work.

In this section, we give an algorithm for computing F_2 over a sequence based dsw. Our algorithm uses the following sampling algorithm.

Theorem 3.23 (Rephrased from [28])**.** *There is a sequence-based dsw algorithm for keeping a uniform sample of size s drawn with replacement from the active window with cost $\langle \tilde{O}(Ks), \tilde{O}(K+s), \tilde{O}(1)\rangle$ with high probability.*

We also use the following algorithm for computing the second moment of an unknown probability distribution over n items. We recall that the second moment of a probability distribution, $P = \langle p_1, p_2, \ldots, p_n\rangle$ is $||P||_2^2 = \sum_i p_i^2$.

Theorem 3.24 (Rephrased from [40])**.** *Given an unknown distribution P, known to have support size at most n, there is an algorithm which computes $||P||_2^2$ up to $(1 \pm \epsilon)$ approximation, using $O(\sqrt{n} \log \frac{1}{\delta}/\epsilon^2)$ samples with probability at least $(1-\delta)$.*

Theorem 3.25. *There is a sequence-based dsw algorithm for computing a $(1 \pm \epsilon)$-approximate F_2, with cost $\langle \tilde{O}(K\sqrt{n}/\epsilon^2), \tilde{O}(K + \sqrt{n}/\epsilon^2), \tilde{O}(1)\rangle$, with high probability.*

Proof. We note that the frequencies of the items from the active window $\langle f_1, f_2, \ldots, f_n\rangle$ naturally induces a distribution with probability values $D = \langle f_1/W, f_2/W, \ldots, f_n/W\rangle$, obtained by sampling with replacement. This distribution satisfies $||D||_2^2 = F_2/W^2$. Thus, a $(1 \pm \epsilon)$ approximation for $||D||_2^2$ could be used to approximate F_2. From Theorem 3.24, a set of $\tilde{O}(\sqrt{n}/\epsilon^2)$ samples from the active window are enough for approximating $||D||_2^2$ with high probability, which we accomplish using the algorithm of Theorem 3.23. □

A similar technique can be used to approximate other frequency moments. We skip the details.

Chapter 4

Introduction to Distribution Testing

In the area of *property testing* we are given limited access to a large object and the goal is to infer certain properties of interest about the object [39]. The total cost of the accesses defined appropriately should be as small as possible. In this part of the thesis we make progress on certain questions from *distribution testing*, where the unknown object is an unknown distribution P, and the access to it is provided by giving independent samples from it.

The related problem of hypothesis testing an unknown distribution have seen extensive research spanning more than 300 years [5, 58]. As a motivating example of distribution testing, we would like to mention the pioneering work of John Snow in epidemiology [61], when during the severe Cholera outbreak of 1854 in London, he studied whether the distribution of the patients are consistent with a unimodal distribution, one that has a single peak and the probability of infection reduces with distance.

More formally let P be a distribution over the sample space $[m] = \{1, 2, \ldots, m\}$. Let $\mathcal{C}$ be a set of distributions over $[m]$, which is of interest to us. In $(\mathcal{C}, \epsilon)$ distribution testing, we need to distinguish between the following two cases:

1. $P \in \mathcal{C}$

2. $dist(P, \mathcal{C}) = \min_{C \in \mathcal{C}} dist(P, C) > \epsilon$,

where $dist$ is a well defined distance function between two probability distributions, and ϵ is a parameter, using as few as samples as possible from P. The following three important classes $\mathcal{C}$, ordered from easier to harder, are thoroughly studied in the distribution testing literature:

1. (Uniformity testing) $\mathcal{C}$ is the singleton set having uniform distribution over $[m]$
2. (Identity testing or 1-sample testing) $\mathcal{C}$ is the singleton set having a particular fully known distribution Q over $[m]$
3. (Closeness tesing or 2-sample testing) $\mathcal{C}$ is the singleton set having another unknwon distribution Q over $[m]$, to which we also have sample access, and the goal is to minimize the total number of samples taken from either P or Q.

Other examples of $\mathcal{C}$ include unimodal (unimodality testing), monotonic (monotonicity testing) and product (independence testing) distributions.

A *tester* T for $(\mathcal{C}, \epsilon)$ distribution testing an unknown distribution P, working with a set S of independent samples from D, should have the following behavior:

1. If $P \in \mathcal{C}$ then $T(S)$ outputs 'yes' with probability 2/3,
2. If $dist(P, \mathcal{C}) > \epsilon$, then $T(S)$ outputs 'no' with probability 2/3.

If $0 < dist(P, \mathcal{C}) \leq \epsilon$, T can answer arbitrarily. The sample complexity of T is the maximum number of samples taken by it for $(\mathcal{C}, \epsilon)$ distribution testing any unknown ditribution P over $[m]$. The sample complexity of $(\mathcal{C}, \epsilon)$ distribution testing is defined as the minimum sample complexity of among all testers T for it. The success probability 2/3 in the above can be replaced by any constant $(1/2, 1]$ and can be amplified to $(1 - \delta)$, using a majority vote of the outputs of $O(\log(1/\delta))$ repititions of T. In

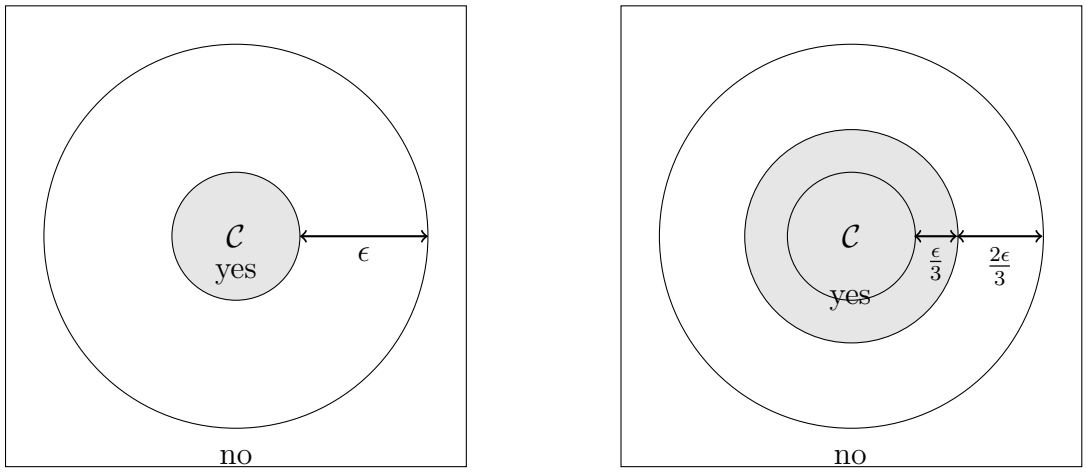

Figure 4.1: Distribution testing (left) and its tolerant version (right). In both cases, the innermost circle represents the class $\mathcal{C}$, the 'yes class' is shaded, and the 'no class' lies outside the outermost circle.

distribution testing, we are interested to know the asymptotic sample complexity of $(\mathcal{C}, \epsilon)$ distribution testing, for various interesting classes of distributions $\mathcal{C}$.

In practice, often we want the following 'tolerant' behavior from T:

1. If $dist(P, \mathcal{C}) \leq \epsilon/3$ then $T(S)$ outputs 'yes' with probability 2/3,

2. If $dist(P, \mathcal{C}) > \epsilon$, then $T(S)$ outputs 'no' with probability 2/3.

If $\epsilon/3 < dist(P, \mathcal{C}) \leq \epsilon$, T can answer arbitrarily. Note that, this test is stronger than that discussed in the last paragraph, because if $P \in \mathcal{C}$ then $dist(P, \mathcal{C}) = 0 \leq \epsilon/3$. Note that, the tolerant behaviour could instead be defined with respect to a second distance function $dist'$ and a second distance parameter ϵ', provided $dist'(P, \mathcal{C}) \leq \epsilon'$ and $dist(P, \mathcal{C}) > \epsilon$ could never happen simultaneously for a distribution P. Such testers are called ($dist'$ versus $dist$) tolerant testers, as opposed to the non-tolerant ones discussed previously. The classes of distributions satisfying $dist(P, \mathcal{C}) \leq \epsilon/3$ and $dist(P, \mathcal{C}) > \epsilon$ are often referred to as the 'yes class' and the 'no class' respectively for convenience. See Figure 4.1 for an illustration.

4.1 Distances between probability distributions

We define the distance measures between distributions that we use in this part.

Definition 4.1. *Let $P = (p_1, p_2, \ldots, p_m)$ and $Q = (q_1, q_2, \ldots, q_m)$ be two distributions over the sample space $[m] = \{1, 2, \ldots, m\}$. Then the distance measures total variational distance, chi-squared distance, Hellinger distance, and KL distance, respectively are defined as follows.*

- $d_{\mathrm{TV}}(P, Q) = \frac{1}{2} \sum_i |p_i - q_i|$
- $d_{\chi^2}(P, Q) = \sum_i (p_i - q_i)^2 / q_i = \sum_i p_i^2 / q_i - 1$
- $d^2_{\mathrm{H}}(P, Q) = \frac{1}{2} \sum_i (\sqrt{p_i} - \sqrt{q_i})^2 = 1 - \sum_i \sqrt{p_i q_i}$, $d_{\mathrm{H}}(P, Q) = \sqrt{d^2_{\mathrm{H}}(P, Q)}$
- $d_{\mathrm{KL}}(P, Q) = \sum_i p_i \ln \frac{p_i}{q_i}$

The most popular choice for *dist* for $(\mathcal{C}, \epsilon)$ distribution testing has been d_{TV}, which we'll assume, unless otherwise mentioned.

4.2 History of distribution testing

Goldreich and Ron [40] formally pioneered the work on distribution testing, by giving a uniformity tester with $O(\sqrt{m}/\epsilon^4)$ samples. Paninski [57] showed uniformity testing has sample complexity $\Omega(\sqrt{m}/\epsilon^2)$ samples and gave an optimal tester assuming $\epsilon > m^{-1/4}$. Batu, Fischer, Fortnow, Kumar, Rubinfeld and White [8] gave the first identity tester working in $O(\sqrt{m}\mathrm{poly}(\log n, \epsilon^{-1}))$ samples, which was improved to optimal $O(\sqrt{m}/\epsilon^2)$ by Valiant and Valiant [65]. The closeness testing problem was first studied by Batu, Fortnow, Rubinfeld, Smith, and White [9] and its sample complexity was settled to $\Theta(\max\{m^{2/3}/\epsilon^{4/3}, \sqrt{m}/\epsilon^2\})$ by Chan et al. [22]. Valiant and Valiant showed that d_{TV} tolerant uniformity testing has sample complexity $\Theta(m/\log m)$ [63, 64]. To

circumvent this lower bound, Acharya, Daskalakis and Kamath [1] gave a chi-squared tolerant identity tester. Daskalakis, Kamath, and Wright [32] thoroughly studied tolerant testing for every pairs of distances among the four defined in 4.1. We refer the reader to a survey by Canonne [18] for more details on distribution testing.

4.3 Poisson sampling

We briefly describe an important analytic tool, known as Poisson sampling that we use in Chapter 5. Let $P = (p_1, p_2, \ldots, p_m)$ be a probability distribution. If we take m samples from P, each symbol $i \in [m]$ marginally follows the distribution $N_i \sim \text{Binomial}(m, p_i)$. However, there are non-zero correlations between N_i and N_j for $i \neq j$. As an example, given the value of N_i for some i is too large, the other N_j values for $j \neq i$ are small, since the total number of samples are fixed. If instead we take $\text{Poi}(m)$ samples, which means we first draw a number $N \sim \text{Poisson}(m)$ and then take N samples from P, these correlations become zero. Moreover, each $N_i \sim Poi(mp_i)$ independent of other N_j values for $j \neq i$, from the properties of the Poisson distribution.

Finally, Poisson sampling does not change the asymptotic sample complexity. Since due to strong concentration of the Poisson distribution around its mean, $N = Theta(m)$ except for a very small constant probability.

4.4 Summary of our results on distribution testing

In Chapter 5 we present our results on tolerant identity and closeness testing of high-dimensional product distributions, over the sample space Σ^n. Without any assumption, testing distributions over Σ^n would require $|\Sigma|^{n/2}$ samples. Thus recently testing problems for distributions over $|\Sigma|^{n/2}$ were studied with additional assump-

tions about them. Our results give the first tolerant testers for this testing problem, when the underlying distribution is known to be a product distribution.

Chapter 5

Efficient Tolerant Testing of High-Dimensional Product Distributions

5.1 Introduction

In this chapter, we initiate the study of *tolerant testing* over an exponential size sample space Σ^n, for an alphabet Σ. Recall from Section 4.2, in this case, the simpler problem of non-tolerant uniformity testing would require $\Omega(|\Sigma|^{n/2})$ samples, which is prohibitive. To circumvent this difficulty, testing problems for distributions over such high-dimensional sample space were investigated with additional structural assumptions, independently and concurrently in the following three papers, [31, 19, 33]. These papers gave efficient non-tolerant testers for such distributions coming out of small degree bayes nets and ising models.

For this work, we entirely focus on product distributions, and show that efficient tolerant testers could be designed for them. In fact, we comprehensively study the tolerant testing problem for product distributions for various pairs of distances from Section 4.1.

5.1.1 Our Contributions

Our contributions regarding the tolerant identity testing problem of product distributions are summarized in Table 5.1, where each cell represents the sample complexity of testing whether the two product distributions are close or far in terms of the distance corresponding to that row and column respectively. Refer to Section 5.2 for notation and definitions. The problems become harder as we traverse the table down or to the right due to the following inequality:

$$d_{\mathrm{H}}^2(P,Q) \leq d_{\mathrm{TV}}(P,Q) \leq \sqrt{2} d_{\mathrm{H}}(P,Q) \leq \sqrt{d_{\mathrm{KL}}(P,Q)} \leq \sqrt{d_{\chi^2}(P,Q)} \tag{5.1}$$

Table 5.1: Sample complexity of identity testing of product distributions. Sample complexity is non-decreasing, as we traverse the table down or to the right. [†] are from [33] and [∗] are from [19].

	$d_{\mathrm{TV}}(P,Q) > \epsilon$	$\sqrt{2} d_{\mathrm{H}}(P,Q) > \epsilon$	$d_{\mathrm{KL}}(P,Q) > \epsilon^2$	$d_{\chi^2}(P,Q) > \epsilon^2$
$P = Q$	$O(\sqrt{n}/\epsilon^2)$ $\Omega(\sqrt{n}/\epsilon^2)$ (for $\lvert\Sigma\rvert = 2$) [†,∗]		Untestable	
$d_{\chi^2}(P,Q) \leq \epsilon^2/9$		$O(\lvert\Sigma\rvert^{3/2}\sqrt{n}/\epsilon^2)$ Theorem 5.2		
$d_{\mathrm{KL}}(P,Q) \leq \epsilon^2/9$	$\Omega(n/\log n)$ Theorem 5.5			
$\sqrt{2} d_{\mathrm{H}}(P,Q) \leq \epsilon/3$		$O(\lvert\Sigma\rvert n \log n/\epsilon^2)$ Theorem 5.3		
$d_{\mathrm{TV}}(P,Q) \leq \epsilon/3$	$\Omega(n/\log n)$ [∗]			

Next, we consider the tolerant closeness testing problem of product distributions. Our results are summarized in Table 5.2 below.

Table 5.2: Sample complexity of closeness testing of product distributions. Sample complexity is non-decreasing, as we traverse the table down or to the right. [∗] are from [19].

	$d_{\mathrm{TV}}(P,Q) > \epsilon$	$\sqrt{2} d_{\mathrm{H}}(P,Q) > \epsilon$	$d_{\mathrm{KL}}(P,Q) > \epsilon^2$	$d_{\chi^2}(P,Q) > \epsilon^2$
$P = Q$	$O(\max(\sqrt{n}/\epsilon^2, n^{3/4}/\epsilon))$ $\Omega(\max(\sqrt{n}/\epsilon^2, n^{3/4}/\epsilon))$ (for $\lvert\Sigma\rvert = 2$) [*] $O\left(\max\left\{\sqrt{n\lvert\Sigma\rvert}/\epsilon^2, (n\lvert\Sigma\rvert)^{3/4}/\epsilon^{3/2}\right\}\right)$ Theorem 5.7	$O((n\lvert\Sigma\rvert)^{3/4}/\epsilon^2)$, Theorem 5.6	Untestable	
$d_{\chi^2}(P,Q) \leq \epsilon^2/9$	$\Omega(n/\log n)$ Theorem 5.4			
$d_{\mathrm{KL}}(P,Q) \leq \epsilon^2/9$				
$\sqrt{2} d_{\mathrm{H}}(P,Q) \leq \epsilon/3$		$O(\lvert\Sigma\rvert n \log n/\epsilon^2)$ Theorem 5.3		
$d_{\mathrm{TV}}(P,Q) \leq \epsilon/3$	$\Omega(n/\log n)$ [∗]			

5.1.2 Related Work

Daskalakis, Dikkala, and Kamath [31] studied testing problems for high-dimensional distributions coming out of ising models. Daskalakis and Pan [33] gave identity tester for high-dimesional distributions coming out of Bayesian networks on small degree graphs. Canonne, Diakonikolas, Kane, and Stewart [19] gave identity testers for high dimensional distributions coming from Bayes nets under certain assumptions.

In particular, the sample complexity of identity testing for product distributions over $\{0,1\}^n$ was settled to be $\Theta(\sqrt{n}/\epsilon^2)$ in [19]. The same for closeness testing was settled to be $\Theta(\max(\sqrt{n}/\epsilon^2, n^{3/4}/\epsilon))$ in [19]. The identity tester of [19] is claimed to have certain weaker ($O(\epsilon^2)$ in d_{TV}, see Remark 8) tolerance. A reduction from testing problems for product distributions over alphabet Σ, to that for the Bayes nets of degree $\lfloor \log_2 |\Sigma| \rfloor - 1$, was given in [19] (Remark 55 of their paper). Canonne et al. [19] also show that for product distributions, $\Omega(n/\log n)$ samples are necessary for tolerant identity and closeness testing with respect to the total variation distance.

Thus, the previous work on testing product distributions entirely focused on the boolean alphabet and the d_{TV} distance without any significant tolerance behavior. Our work in this chapter is aimed at addressing these gaps.

5.1.3 Our techniques

We consider product distributions over an arbitrary alphabet Σ. Let $l = |\Sigma|$. There are nl parameters of the product distributions $P = \prod_{i=1}^{n} P_i$: p_{ij} is the probability of seeing the symbol $j \in \Sigma$ in the i-th coordinate of the sample, for every $i \in [n]$. Similarly for $Q = \prod_{i=1}^{n} Q_i$ and Q_{ij}s.

Our d_{χ^2}-tolerant identity test statistic uses a function of W_{ij} and q_{ij} values (we have complete knowledge of Q), where $W_{ij} \sim \mathrm{Poi}(p_{ij})$ are sampled using Poisson

sampling. Under this sampling, the property of our test statistic is reminiscent of the identity tester of [1] for unstructured distributions over nl items, and its behaviour is well understood. We show a separation of this statistic for the 'yes' and 'no' cases, using a sub-additivity result for d_{H}^2 from [33], and using a super-additivity result for d_{χ^2} which we derive in this chapter. Our non-tolerant closeness testers over arbitrary alphabet are reminiscent of the closeness testers of [22, 32] for unstructured distributions over nl items.

Our Hellinger tolerant (Hellinger-versus-Hellinger) closeness tester uses the testing-by-learning approach. For distributions supported over Σ^n, getting such an approach to work efficiently is not as obvious as it seems. For such an approach to work, we need the following components: 1) the distance satisfies triangle inequality, 2) the distance has a 'localization equality' (Lemma 5.22) and 'localization subadditivity' (Lemma 5.11), and 3) it has an efficient learning algorithm for small support size distributions. Fortunately, all of these properties hold for the Hellinger distance. We note that for other distances such as d_{TV}, d_{KL}, d_{χ^2}; one of the above conditions does not hold or is not known to hold.

As previously mentioned, for our Hellinger tolerant tester, we design a novel efficient and high probability learning algorithm. Such an algorithm was previouly known for total variation distance. We are not aware of such an algorithm for learning efficiently in Hellinger distance. Our algorithm works as follows. We first obtain an empirical distribution by sampling from the unknown distribution with enough number of samples to make the expected d_{H}^2 distance at most ϵ^2. Then from Markov's inequality with at least $2/3$ probability the distance is at most $O(\epsilon^2)$. In order to amplify the success probability, we use a 'clustering trick', reminiscent of the 'median trick'. The 'median trick' cannot be used in the context of learning an unknown distribution.

For our lower bounds, we reduce from the testing problems on product distributions over $\{0,1\}^n$ to the testing problems on unstructured distributions over n items, known from the work of [19]. Our lower bounds for tolerant testing problems depend on certain upper bounds, which we derive in this chapter.

5.1.4 Organization of the rest of the chapter

The rest of the chapter is organized as follows. In the next section, we give necessary definitions and also state the main results of the chapter. In Section 5.3, we present proofs of our upper bound results on tolerant testing. This section also contains proof of the non-tolerant testers. In Section 5.4, we give proofs of our lower bound results.

5.2 Preliminaries

Lemma 5.1. *For two distributions P and Q, the following relation holds.*

$$d_{\mathrm{H}}^2(P,Q) \leq d_{\mathrm{TV}}(P,Q) \leq \sqrt{2} d_{\mathrm{H}}(P,Q) \leq \sqrt{d_{\mathrm{KL}}(P,Q)} \leq \sqrt{d_{\chi^2}(P,Q)}$$

See [32] for a proof of the above chain of inequalities. Let d_1, d_2 be any distance function(s) from Lemma 5.1 together with the constant (for example $d_1 = \sqrt{d_{\chi^2}(P,Q)}$ and $d_2 = \sqrt{2} d_{\mathrm{H}}(P,Q)$).

d_1-versus-d_2 identity (1-sample) testing: Given an unknown distribution P, which we have sample access to, and a known distribution Q, an error parameter $0 < \epsilon < 1$, and a constant gap parameter $0 \leq \alpha < 1$, a *d_1-versus-d_2 identity (1-sample) tester* is an algorithm with the following behavior: (1) If $d_1(P,Q) \leq \alpha\epsilon$ it outputs 'yes' with probability at least 2/3, (2) If $d_2(P,Q) > \epsilon$ it outputs 'no' with probability at least 2/3, (3) If neither of the above two cases hold, it outputs 'yes' or 'no' arbitrarily.

d_1-versus-d_2 closeness (2-sample) testing: Given two unknown distributions P and Q which we have sample access to, an error parameter $0 < \epsilon < 1$, and a constant gap parameter $0 \leq \alpha < 1$, a *a d_1-versus-d_2 closeness (2-sample) tester* is an algorithm with the following behavior: If $d_1(P, Q) \leq \alpha\epsilon$ it outputs 'yes' with probability at least 2/3, (2) If $d_2(P, Q) > \epsilon$ it outputs 'no' with probability at least 2/3, (3) If neither of the above two cases hold, it outputs 'yes' or 'no' arbitrarily.

The special case $\alpha = 0$ is the non-tolerant d_2 tester in both the 1-sample and 2-sample cases. The sample complexity of a tester is the number of samples it takes in the worst case. The sample complexity of *d_1-versus-d_2 identity (closeness) testing problem* is the minimum sample complexity over all *d_1-versus-d_2 identity (closeness) testers*.

The following facts are easy to observe. A d_1-versus-d_2 tester implies an non-tolerant d_2 tester, with the same sample complexity. A d_1-versus-d_2 tester implies a d_1'-versus-d_2' tester, with the same sample complexity, when d_1' is the distance right of d_1 and d_2' is the distance left of d_2 in the chain from Lemma 5.1. A d_1-versus-d_2 closeness tester implies a d_1-versus-d_2 identity tester, with the same sample complexity.

It is known from [32] that there is no non-tolerant $\sqrt{KL}$ identity tester with finite sample complexity. Thus, Hellinger distance is the most general among those considered, we can aim for d_2 (the 'no' case). In this chapter, we remove the constants and the square roots in the distances in Lemma 5.1 when we mention a d_1-versus-d_2 tester.

In this chapter we interchangeably use identity testing (closeness testing) and 1-sample testing (respectively 2-sample testing). Let Bern(δ) be the Bernoulli distribution with $\Pr[1] = \delta$.

5.2.1 Formal statement of our main results

Here we list the formal statements of the main theorems we prove in the chapter. First we state the two main upper bounds, which are proved in Section 5.3.

Theorem 5.2. *(d_{χ^2}-versus-d_{H} identity tester) Let $P = \prod_{i=1}^{n} P_i$ be an unknown distribution and $Q = \prod_{i=1}^{n} Q_i$ be a known distribution, both over the common sample space Σ^n. Then testing $d_{\chi^2}(P, Q) \leq \epsilon^2/9$ versus $\sqrt{2}d_{\mathrm{H}}(P, Q) > \epsilon$ can be performed with $O(\sqrt{n}|\Sigma|^{3/2}/\epsilon^2)$ samples, in time $O(n^{3/2}|\Sigma|^{3/2}/\epsilon^2)$, with success probability at least $2/3$.*

Theorem 5.3. *(d_{H}-versus-d_{H} closeness tester) Let $P = \prod_{i=1}^{n} P_i$ and $Q = \prod_{i=1}^{n} Q_i$ be two distributions over Σ^n. There is an algorithm to distinguish $\sqrt{2}d_{\mathrm{H}}(P, Q) \leq \epsilon/3$ ('yes') versus $\sqrt{2}d_{\mathrm{H}}(P, Q) > \epsilon$ ('no') with $O(|\Sigma| n \log n/\epsilon^2)$ samples from each of P and Q. The running time is $O(|\Sigma| \log |\Sigma| n^2 \log^2 n/\epsilon^2)$ and the success probability is at least $1 - 2/n$.*

We complement the above upper bounds on sample complexity with the following lower bounds. See Section 5.4 for the details.

Theorem 5.4. *(d_{χ^2}-versus-d_{TV} closeness testing lower bound) There exists a constant $0 < \epsilon < 1$ and three product distributions F^{yes}, F^{no} and F, each over the sample space $\{0, 1\}^n$ such that $d_{\chi^2}(F^{yes}, F) \leq \epsilon^2/9$, whereas $d_{\mathrm{TV}}(F^{no}, F) > \epsilon$, and given only sample accesses to F^{yes}, F^{no} and F, distinguishing F^{yes} versus F^{no} with probability $> 2/3$, requires $\Omega(n/\log n)$ samples.*

Theorem 5.5. *(d_{KL}-versus-d_{TV} identity testing lower bound) There exists a constant $0 < \epsilon < 1$ and three product distributions F^{yes}, F^{no} and F, each over the sample space $\{0, 1\}^n$ such that $d_{\mathrm{KL}}(F^{yes}, F) \leq \epsilon^2/9$, whereas $d_{\mathrm{TV}}(F^{no}, F) > \epsilon$, and given only*

sample accesses to F^{yes}, F^{no}, and complete knowledge about F, distinguishing F^{yes} versus F^{no} with probability $> 2/3$, requires $\Omega(n/\log n)$ samples.

Earlier work has designed non-tolerant 2-sample tester for product distribution over $\Sigma = \{0, 1\}$. Here we extend it to arbitrary alphabets. We state the results. Proofs are in Section 5.3.5.

Theorem 5.6. *(Exact-versus-d_{H} closeness tester) Let $P = \prod_{i=1}^{n} P_i$ and $Q = \prod_{i=1}^{n} Q_i$ be two unknown distributions, both over the common sample space Σ^n. Then testing $P = Q$ versus $\sqrt{2}d_{\mathrm{H}}(P, Q) > \epsilon$ can be performed with $m = O((n|\Sigma|)^{3/4}/\epsilon^2)$ samples, in time $O(mn)$, with success probability at least $51/100$.*

Theorem 5.7. *(Exact-versus-d_{TV} closeness tester) Let $P = \prod_{i=1}^{n} P_i$ and $Q = \prod_{i=1}^{n} Q_i$ be two unknown distributions, both over the common sample space Σ^n. Then testing $P = Q$ versus $d_{\mathrm{TV}}(P, Q) > \epsilon$ can be performed with $m = O(\max\{\sqrt{n|\Sigma|}/\epsilon^2, (n|\Sigma|)^{3/4}/\epsilon^{3/2}\})$ samples, in time $O(mn)$, with success probability at least $51/100$.*

5.3 Efficient Tolerant Testers for Product distributions over Σ^n

5.3.1 Chi-squared tolerant 1-sample tester for product distributions

In this section we generalize the testers [33, 19] that distinguishes $P = Q$ ('yes class') versus $d_{\mathrm{TV}}(P, Q) \geq \epsilon$ ('no class') using $O(\sqrt{n}/\epsilon^2)$ samples, where P and Q are product distributions over $\{0, 1\}^n$. Our first contribution is to generalize their tester in the following three ways. Firstly, our 'no class' is defined as $\sqrt{2}d_{\mathrm{H}}(P, Q) \geq \epsilon$, which is more general than $d_{\mathrm{TV}}(P, Q) \geq \epsilon$. Secondly, our tester works for any general alphabet size $|\Sigma| \geq 2$. Finally, we give a d_{χ^2} tolerant tester i.e. our 'yes class' is defined as $d_{\chi^2}(P, Q) < \epsilon^2/9$.

Our tester relies on certain result on the multiplicativity of Chi-squared distance for product distributions.

Lemma 5.8. *Let $P = \prod_{i=1}^{n} P_i$ and $Q = \prod_{i=1}^{n} Q_i$ be two distributions, both over the common sample space Σ^n. Then $d_{\chi^2}(P, Q) = \prod_{i=1}^{n}(1 + d_{\chi^2}(P_i, Q_i) - 1$.*

Proof. Let $P(i)$ and $Q(i)$ be the probability of an item $i = (i_1 i_2 \dots i_n) \in \Sigma^n$ in P and Q respectively. Then $P(i) = P_1(i_1) \dots P_n(i_n)$ and $Q(i) = Q_1(i_1) \dots Q_n(i_n)$, where $P_j(i_j)$ and $Q_j(i_j)$ is the probability of item i_j in P_j and Q_j respectively.

$$\begin{aligned}
d_{\chi^2}(P, Q) &= \sum_{i \in \Sigma^n} (P(i) - Q(i))^2 / Q(i) \\
&= \sum_{i \in \Sigma^n} P^2(i)/Q(i) - 1 \\
&= \sum_{(i_1 i_2 \dots i_n) \in \Sigma^n} \frac{P_1^2(i_1) \dots P_n^2(i_n)}{Q_1(i_1) \dots Q_n(i_n)} - 1 \\
&= \sum_{i_1 \in \Sigma} \frac{P_1^2(i_1)}{Q_1(i_1)} \cdots \sum_{i_n \in \Sigma} \frac{P_n^2(i_n)}{Q_n(i_n)} - 1 \\
&= (1 + d_{\chi^2}(P_1, Q_1)) \dots (1 + d_{\chi^2}(P_n, Q_n)) - 1
\end{aligned}$$

□

We get the following useful corollary from Lemma 5.8.

Corollary 5.9. *Let $P = \prod_{i=1}^{n} P_i$ and $Q = \prod_{i=1}^{n} Q_i$ be two distributions, both over the common sample space Σ^n. Then $d_{\chi^2}(P, Q) \geq \sum_i d_{\chi^2}(P_i, Q_i)$.*

We also need the following upper bound of Hellinger distance by chi-squared distance.

Lemma 5.10. *Let P and Q be two distributions over a common sample space. Then $2d_{\mathrm{H}}^2(P, Q) \leq d_{\chi^2}(P, Q)$.*

Proof. For any item i in the common sample space let $P(i)$ and $Q(i)$ denote the probability values of item i under P and Q respectively.

$$\begin{aligned} 2d^2_{\mathrm{H}}(P,Q) &= \sum_i (\sqrt{P(i)} - \sqrt{Q(i)})^2 \\ &= \sum_i \frac{(P(i)-Q(i))^2}{(\sqrt{P(i)}+\sqrt{Q(i)})^2} \\ &\leq \sum_i \frac{(P(i)-Q(i))^2}{Q(i)} \\ &= d_{\chi^2}(P,Q) \end{aligned}$$

□

We recall the following subadditivity result for product distributions.

Lemma 5.11. *[[33]] Let $P = \prod_{i=1}^n P_i$ and $Q = \prod_{i=1}^n Q_i$ be two product distributions, both over the common sample space Σ^n. Then $d^2_{\mathrm{H}}(P,Q) \leq \sum_i d^2_{\mathrm{H}}(P_i,Q_i)$*

We get the following useful corollary from Lemma 5.10 and Lemma 5.11.

Corollary 5.12. *Let $P = \prod_{i=1}^n P_i$ and $Q = \prod_{i=1}^n Q_i$ be two distributions, both over the common sample space Σ^n. Then $d^2_{\mathrm{H}}(P,Q) \leq \sum_i d_{\chi^2}(P_i,Q_i)/2$.*

For a technical reason, we need to ensure that for each distribution Q_i, each element in the sample space Σ gets at least a sufficiently large probability $\Omega(\epsilon^2/|\Sigma|n)$. We do this by *slightly randomizing* Q to get a new distribution S. The randomization process to get S from Q is given below. This is similar to the reduction given in [33] and [19] for the case $\Sigma = \{0,1\}$ and for the case when the 'no' class is defined with respect to d_{TV}. Let $\mathrm{Bern}(\delta)^n$ be the product distribution of n copies of $\mathrm{Bern}(\delta)$.

Lemma 5.13. *For a product distribution $P = \prod_{i=1}^{n} P_i$, where $P_i's$ are over a sample space Σ, and $0 < \delta < 1$, let P^δ be the distribution over Σ^n defined by the following sampling process. In order to produce a sample $(X_1, X_2, \ldots, X_n)$ of P^δ,*

- *Sample $(r_1, r_2, \ldots, r_n) \sim \mathrm{Bern}(\delta)^n$ and sample $(Y_1, Y_2, \ldots, Y_n) \sim P$*
- *For every i, if $r_i = 0$, $X_i \leftarrow$ uniform sample from Σ, if $r_i = 1$, $X_i \leftarrow Y_i$.*

Then, the following is true.

- *P^δ is a product distribution $\prod_i P_i^\delta$ and each sample from P^δ can be simulated by 1 sample from P.*
- *For every $i : 1 \leq i \leq n$ and $j \in \Sigma$, $P_i^\delta(j) \geq \delta/|\Sigma|$.*
- *$d_{\mathrm{H}}^2(P, P^\delta) \leq 2n\delta$*

Proof. The first part is obvious from the sampling process. For the second part, $P_i^\delta(j) = (1-\delta)P_i(j) + \delta/|\Sigma| \geq \delta/|\Sigma|$ for every i, j.

The proof of the third part also appears in [32]. We give a slightly general version here. For convenience, we denote P^δ by R. Define the event E to be $a_i = 0$ for every i. Then $\Pr(E) = (1-\delta)^n \geq (1 - n\delta)$ and $\Pr[\bar{E}] \leq n\delta$. Also note that conditioned on the event E, for any item $i \in \Sigma^n$, the probability values satisfy $R(i|E) = P(i)$.

$$
\begin{aligned}
d^2_{\mathrm{H}}(P,R) &= \sum_i (\sqrt{R(i)} - \sqrt{P(i)})^2 \\
&= \sum_i (\sqrt{R(i|E)\Pr(E) + R(i|\bar{E})\Pr(\bar{E})} - \sqrt{P(i)})^2 \\
&= \sum_i (\sqrt{P(i)\Pr(E) + R(i|\bar{E})\Pr(\bar{E})} - \sqrt{P(i)})^2 \\
&\leq \sum_i [(\sqrt{P(i)\Pr(E)} - \sqrt{P(i)})^2 + (\sqrt{R(i|\bar{E})\Pr(\bar{E})})^2] \\
&\quad \text{(Using } (\sqrt{a+b} - \sqrt{c+d})^2 \leq (\sqrt{a} - \sqrt{c})^2 + (\sqrt{b} - \sqrt{d})^2 \text{ for non-negative } a,b,c,d) \\
&= (1 - \sqrt{\Pr(E)})^2 \sum_i P(i) + \Pr(\bar{E}) \sum_i R(i|\bar{E}) \\
&= (1 - \sqrt{1 - \Pr(\bar{E})})^2 + \Pr(\bar{E}) \\
&\leq 2\Pr(\bar{E}) \qquad \text{(Using } (1 - \sqrt{1-x})^2 \leq x \text{ for } 0 \leq x \leq 1) \\
&\leq 2n\delta
\end{aligned}
$$

□

Lemma 5.14. *Let $P = \prod_{i=1}^n P_i$ and $Q = \prod_{i=1}^n Q_i$ be two distributions, both over the common sample space Σ^n. Let $l = |\Sigma|$, $R = P^\delta$, $S = Q^\delta$ with $\delta = \epsilon^2/50n$. $R = \prod_{i=1}^n R_i$ and $S = \prod_{i=1}^n S_i$, where $R_i = \langle r_{i1}, r_{i2}, \ldots, r_{il} \rangle$ and $S_i = \langle s_{i1}, s_{i2}, \ldots, s_{il} \rangle$ for every i. Then*

(1) If $d_{\chi^2}(P,Q) \leq \epsilon^2/9$ then $\sum_{i,j} \frac{(r_{ij} - s_{ij})^2}{s_{ij}} < 0.12\epsilon^2$.

(2) If $\sqrt{2} d_{\mathrm{H}}(P,Q) \geq \epsilon$ then $\sum_{i,j} \frac{(r_{ij} - s_{ij})^2}{s_{ij}} > 0.18\epsilon^2$.

Proof. (*Proof of (1)*) We have that $r_{ij} = (1-\delta)p_{ij} + \delta/l$ and $s_{ij} = (1-\delta)q_{ij} + \delta/l$.

$$\begin{aligned}
\sum_{i,j} \frac{(r_{ij}-s_{ij})^2}{s_{ij}} &= \sum_{i,j} \frac{(1-\delta)^2(p_{ij}-q_{ij})^2}{(1-\delta)q_{ij}+\delta/l} \\
&\leq \sum_{i,j} \frac{(1-\delta)^2(p_{ij}-q_{ij})^2}{(1-\delta)q_{ij}} \\
&= (1-\delta)\sum_{i,j} \frac{(p_{ij}-q_{ij})^2}{q_{ij}} \\
&< \sum_{i}\sum_{j} \frac{(p_{ij}-q_{ij})^2}{q_{ij}} \\
&= \sum_{i} d_{\chi^2}(P_i, Q_i) \\
&\leq d_{\chi^2}(P,Q) \qquad \text{(From Corollary 5.9)} \\
&< 0.12\epsilon^2
\end{aligned}$$

(*Proof of (2)*). From Lemma 5.13, for $\delta = \epsilon^2/50n$, it follows that $d_{\mathrm{H}}^2(P,R) \leq \epsilon^2/25$ and $d_{\mathrm{H}}^2(Q,S) \leq \epsilon^2/25$. By triangle inequality we get $d_{\mathrm{H}}(P,Q) \leq d_{\mathrm{H}}(R,S) + d_{\mathrm{H}}(P,R) + d_{\mathrm{H}}(Q,S)$. It follows that if $\sqrt{2}d_{\mathrm{H}}(P,Q) \geq \epsilon$ then $d_{\mathrm{H}}(R,S) \geq \epsilon(1/\sqrt{2} - 2/5)$. Then Corollary 5.12 gives $\sum_{i,j} \frac{(r_{ij}-s_{ij})^2}{s_{ij}} = \sum_i d_{\chi^2}(R_i,S_i)) \geq 2d_{\mathrm{H}}^2(R,S) > 0.18\epsilon^2$. □

At this point it remains to test $\sum_{i,j} \frac{(r_{ij}-s_{ij})^2}{s_{ij}} > 0.18\epsilon^2/10$ versus $< 0.12\epsilon^2/10$, which we perform using the tester of Acharya et al. [1].

Remark The test T of Acharya et al. [1] is given by $T = \Sigma_{i=1}^{n} \frac{(N_i - ms_i)^2 - N_i}{ms_i}$. Their paper gives the upper bound $\mathrm{Var}[T] \leq 4n + 9\sqrt{n}\mathrm{E}[T] + \frac{2}{5}n^{\frac{1}{4}}\mathrm{E}[T]^{3/2}$ under the assumption $s_i \geq \epsilon/50n$ for every i. In our application, l is the alphabet size and we will need the bound to depend on l. In addition, we also need the bounds to work when $s_i \geq \epsilon^2/50nl$. Both these can be achieved by modifying their proof, as follows.

Theorem 5.15. *(Modified from [1]) Let m be an integer and $0 < \epsilon < 1$ be an error parameter. Let $r_1, r_2, \ldots, r_n$ be n non-negative real numbers. For an integer parameter l, let $s_1, s_2, \ldots, s_n$ be non-negative real numbers such that $s_i \geq \epsilon^2/50nl$. For $1 \leq i \leq n$, let $N_i \sim \mathrm{Poi}(mr_i)$ be independent samples from $\mathrm{Poi}(mr_i)$. Then there exists a test statistic T, computable in time $O(n)$ from inputs N_is and s_is, with the following guarantees.*

- $\mathrm{E}[T] = m \sum_i \frac{(r_i - s_i)^2}{s_i}$
- $\mathrm{Var}[T] \leq 2n + \sqrt{n}(l+6)\mathrm{E}[T] + 4n^{1/4}\sqrt{l}(\mathrm{E}[T])^{3/2}$, *for a constant c and $m \geq c\sqrt{n}/\epsilon^2$.*

Proof. The test T of Acharya et al. [1] is given by $T = \Sigma_{i=1}^{n} \frac{(N_i - ms_i)^2 - N_i}{ms_i}$.

$$
\begin{aligned}
\mathrm{E}[T] &= \mathrm{E}\left[\sum_i \frac{(N_i - ms_i)^2 - N_i}{ms_i}\right] \\
&= \sum_i \frac{\mathrm{E}[(N_i - ms_i)^2 - N_i]}{ms_i} \\
&= \sum_i \frac{\mathrm{E}[N_i^2] + m^2 s_i^2 - 2ms_i \mathrm{E}[N_i] - \mathrm{E}[N_i]}{ms_i} \\
&= \sum_i \frac{mr_i(1 + mr_i) + m^2 s_i^2 - 2ms_i \cdot mr_i - mr_i}{ms_i} \qquad \text{(Since } N_i \sim Poi(mr_i)\text{)} \\
&= m \sum_i \frac{(r_i - s_i)^2}{s_i}
\end{aligned}
$$

$$\begin{aligned}
\mathrm{Var}[T] &= \mathrm{Var}\left[\sum_i \frac{(N_i - ms_i)^2 - N_i}{ms_i}\right] \\
&= \sum_i \mathrm{Var}\left[\frac{(N_i - ms_i)^2 - N_i}{ms_i}\right] \\
&\qquad\qquad \text{(Since } N_i\text{s are independent for different } i\text{s)} \\
&= \sum_i \frac{1}{m^2 s_i^2}\mathrm{Var}[N_i^2 - (2ms_i + 1)N_i] \\
&= \sum_i \frac{1}{m^2 s_i^2}[\mathrm{Var}[N_i^2] + (2ms_i+1)^2\mathrm{Var}[N_i] - 2(2ms_i+1)\mathrm{Cov}(N_i^2, N_i)] \\
&= \sum_i \frac{1}{m^2 s_i^2}[\lambda(1 + 5\lambda + 4\lambda^2) + \lambda(2ms_i+1)^2 - 2(2ms_i+1)(\lambda(2\lambda+1))] \\
&\qquad\qquad \text{(Since } N_i \sim Poi(\lambda)\text{, where } \lambda = mr_i\text{)} \\
&= \sum_i \frac{1}{m^2 s_i^2}\lambda[\lambda + (2ms_i - 2\lambda)^2] \\
&= \sum_i \frac{r_i^2}{s_i^2} + \sum_i 4mr_i\frac{(r_i - s_i)^2}{s_i^2}
\end{aligned}$$

We bound the above two summations separately.

$$\begin{aligned}
\sum_i \frac{r_i^2}{s_i^2} &= \sum_i \frac{(r_i - s_i)^2 + 2s_i(r_i - s_i) + s_i^2}{s_i^2} \\
&= \sum_i \frac{(r_i - s_i)^2}{s_i^2} + 2\sum_i \frac{r_i - s_i}{s_i} + \sum_i 1 \\
&\le 2\left(\sum_i \frac{(r_i - s_i)^2}{s_i^2} + \sum_i 1\right) && \text{(Using } a^2 + 1 \ge 2a\text{)} \\
&\le 2\left(\frac{50nl}{\epsilon^2}\sum_i \frac{(r_i - s_i)^2}{s_i} + n\right) && \text{(Using } s_i \ge \epsilon^2/50nl\text{)} \\
&= 2(\frac{50nl}{\epsilon^2}\frac{\mathrm{E}[T]}{m} + n) \\
&\le \sqrt{n}l\mathrm{E}[T] + 2n && \text{(Using } m \ge c\sqrt{n}/\epsilon^2 \text{ for c sufficiently large)}
\end{aligned}$$

$$\sum_i 4mr_i \frac{(r_i - s_i)^2}{s_i^2} \leq 4m \sqrt{\sum_i \frac{r_i^2}{s_i^2}} \sqrt{\sum_i \frac{(r_i - s_i)^4}{s_i^2}}$$

(Using Cauchy-Schwarz inequality)

$$\leq 4m \sqrt{\sqrt{n} l \mathrm{E}[T] + 2n} \sum_i \frac{(r_i - s_i)^2}{s_i}$$

$$\leq 4\mathrm{E}[T](n^{1/4}\sqrt{l\mathrm{E}[T]} + \sqrt{2n})$$

Together we get

$$\mathrm{Var}[T] \leq \sqrt{n} l \mathrm{E}[T] + 2n + 4\mathrm{E}[T](n^{1/4}\sqrt{l\mathrm{E}[T]} + \sqrt{2n})$$

$$\leq 2n + \sqrt{n}(l+6)\mathrm{E}[T] + 4n^{1/4}\sqrt{l}(\mathrm{E}[T])^{3/2}$$

□

It remains to sample numbers $N_{ij} \sim \mathrm{Poi}(mr_{ij})$ independently for every i, j. We do this via poissonization followed by sampling from each coordinate of the product distribution R independently [33] and [19]. We present Algorithm 5 with its correctness.

Proof. (*Proof of Theorem 5.2*) Let $l = |\Sigma|$. First, we transform the distributions P and Q into the distributions R and S respectively according to the modification process mentioned in Lemma 5.14. This gives:

- Each sample from R can be simulated by 1 sample from P.
- R and S are product distributions, $R = \prod_{i=1}^n R_i$ and $S = \prod_{i=1}^n S_i$, where $R_i = \langle r_{i1}, r_{i2}, \ldots, r_{il} \rangle$ and $S_i = \langle s_{i1}, s_{i2}, \ldots, s_{il} \rangle$ for every i.
- For every i, j, $s_{ij} \geq \epsilon^2/50ln$.
- If $d_{\chi^2}(P, Q) \leq \epsilon^2/9$ then $\sum_{i,j} \frac{(r_{ij} - s_{ij})^2}{s_{ij}} < 0.12\epsilon^2$.

Algorithm 5: Given samples from an unknown distribution $R = \prod_{i=1}^{n} R_i$ and a known distribution $S = \prod_{i=1}^{n} S_i$ over Σ^n, decide $d_{\chi^2}(R, S) \leq \epsilon^2/10$ ('yes') versus $d_{\mathrm{H}}(R, S) \geq \epsilon$ ('no'). Let $l = |\Sigma|$, $S_i = \langle s_{i1}, s_{i2}, \ldots, s_{il} \rangle$ with $s_{ij} \geq \epsilon^2/50nl$ for every j, for every i

for $i = 1$ *to* n **do**
  Sample $N_i \sim \mathrm{Poi}(m)$ independently;
end
$N = \max_i N_i$;
$X \leftarrow$ Take N samples from R;
for $i = 1$ *to* n **do**
  $X_i \leftarrow$ Sequence of symbols in the i-th coordinate of first N_i samples of X;
  $\langle N_{i1}, N_{i1}, \ldots, N_{il} \rangle \leftarrow$ histogram of symbols in X_i;
end
Compute statistic T of Theorem 5.15 using N_{ij} and s_{ij} values for every i, j;
if $T \leq 0.15m\epsilon^2$ **then**
  output 'yes.';
else
  output 'no.'
end

- If $\sqrt{2}d_{\mathrm{H}}(P, Q) > \epsilon$ then $\sum_{i,j} \frac{(r_{ij} - s_{ij})^2}{s_{ij}} > 0.18\epsilon^2$.

Henceforth, we focus on distinguishing $\sum_{i,j} \frac{(r_{ij} - s_{ij})^2}{s_{ij}} < 0.12\epsilon^2$ versus $> 0.18\epsilon^2$, under the assumption $s_{ij} \geq \epsilon^2/50ln$ for every i, j, by sampling from R. We use the tester T of [1] stated in Theorem 5.15 for this. Firstly, note that in Algorithm 5, the samples S_i is a set of $N_i \sim \mathrm{Poi}(m)$ samples from R_i, independently for every i's. This is because the set of samples are taken from the product distribution $R = R_1 \times R_2 \times \cdots \times R_n$ and the N_i values are independent for different i's. Due to Poissonization it follows $N_{ij} \sim \mathrm{Poi}(r_{ij})$ independently for every i, j. The tester T requires $m \geq c\sqrt{nl}/\epsilon^2$, for some constant c.

If $\sum_{i,j} \frac{(r_{ij} - s_{ij})^2}{s_{ij}} < 0.12\epsilon^2$ then $\mathrm{E}[T] \leq 0.12m\epsilon^2$ and $\mathrm{Var}[T] \leq (2/c^2 + 0.12(l+6)/c + 4(0.12)^3\sqrt{l/c})m^2\epsilon^4$, the last inequality from Theorem 5.15 using $m \geq c\sqrt{n}/\epsilon^2$, and the upper bound for $\mathrm{E}[T]$. By Chebyshev's inequality $T < 0.15m\epsilon^2$ with probability

at least $4/5$, for $c = \Omega(l)$ for an appropriate constant.

If $\sum_{i,j} \frac{(r_{ij}-s_{ij})^2}{s_{ij}} > 0.18\epsilon^2$ then $\mathrm{E}[T] > 0.18m\epsilon^2 \geq 0.18c\sqrt{n}$ and $\mathrm{Var}[T] \leq (1/(0.18c)^2 + (l+6)/0.18c + 4\sqrt{l/0.18c})\mathrm{E}^2[T]$. By Chebyshev's inequality $T > 0.15m\epsilon^2$ with probability at least $4/5$, for $c = \Omega(l)$ for an appropriate constant.

Hence $m \geq c'\sqrt{n}l^{3/2}/\epsilon^2$ for some constant c' suffices for the tester T to distinguish the above two cases. It also follows from the concentration of the Poisson distribution that the number of samples required is $\max_i N_i \leq 2m$, except for probability at most $n \cdot \exp(-m) < 1/10$, using union bound.

The histograms can be computed by a single pass over the n-dimensional sample set S. The statistic T can be computed in time $O(nl)$. So the time complexity is $O(nl + (nl)^{3/2}/\epsilon^2)$. □

5.3.2 Hellinger tolerant 2-sample tester for product distributions

In this section we give a tester for distinguishing $d_{\mathrm{H}}(P,Q) \leq \epsilon$ versus $d_{\mathrm{H}}(P,Q) > 3\epsilon$ for two unknown product distributions P and Q over support Σ^n. To get Theorem 5.3, we rescale ϵ downto $\epsilon/\sqrt{2}$. We take a testing-by-learning approach. In general, such an approach becomes intractable for high dimensional distributions since due to large support size, it is intractable to compute the distance between the two learnt distributions. But as we show, for product distributions and Hellinger distance this computation is tractable.

We first present a general learning algorithm of an unknown distribution in Hellinger distance with high probability with near-optimal sample complexity. Such a result for learning distributions in variation distance is well-known. We found the same proof does not extend to the tighter problem of learning in Hellinger distance (d_{H} between the empirical and the true distributions has a larger 'sensitivity' than that of d_{TV}). To the best of our knowledge this result was not known before and potentially have

other applications beyond testing [11].

5.3.3 Learning Distributions in Hellinger Distance

Theorem 5.16. *Let P be a distribution over N items. Then there is an algorithm that takes $O(N \log \frac{1}{\delta}/\epsilon^2)$ samples from P and outputs a distribution R which satisfy $d_{\mathrm{H}}(P,R) \leq \epsilon$ except with error probability at most δ. The running time of the algorithm is $O(N \log N \log^2 \frac{1}{\delta}/\epsilon^2)$.*

We first prove that the empirical distribution over $O(N/\epsilon^2)$ independent samples is close in square of the Hellinger distance to the unknown distribution in expectation. We will use the following bound on the expectation of square root of a Binomial random variable.

Fact 5.17 ([41])**.** *Let $X \sim \mathrm{Bin}(n,p)$. Then $E(\sqrt{X}) \geq \sqrt{np} - \frac{(1-p)}{2\sqrt{np}}$*

Lemma 5.18. *Let P be a distribution over a sample space of N items. Let R be the empirical distribution obtained by taking $m \geq (N-1)/2\epsilon^2$ i.i.d. samples from P. Then, $\mathrm{E}[d^2_{\mathrm{H}}(P,R)] \leq \epsilon^2$.*

Proof. Let Ω be the sample space of P with p_i being the probability of item $i \in \Omega$. Let S be the set of m independent samples obtained. For each $i \in \Omega$ we denote by m_i the number of samples in S which are item i. Then m_i is distributed according to $\mathrm{Bin}(m,p_i)$. $d^2_{\mathrm{H}}(P,R) = 1 - \sum_i \sqrt{p_i m_i/m}$. So $\mathrm{E}[d^2_{\mathrm{H}}(P,R)] = 1 - \sum_i \mathrm{E}[\sqrt{p_i m_i/m}] = 1 - \sum_i \sqrt{p_i/m}\mathrm{E}[\sqrt{m_i}]$. Note that $\mathrm{E}[\sqrt{m_i}] \geq \sqrt{mp_i} - (1-p_i)/2\sqrt{mp_i}$ from Fact 5.17. We get $\mathrm{E}[d^2_{\mathrm{H}}(P,R)] \leq 1 - \sum_i \sqrt{p_i/m}[\sqrt{mp_i} - (1-p_i)/2\sqrt{mp_i}] = 1 - \sum_i [p_i - (1-p_i)/2m] = (N-1)/2m \leq \epsilon^2$ for $m \geq (N-1)/2\epsilon^2$. □

We get the following corollary from Markov's inequality.

Corollary 5.19. *Let P and R be as in Lemma 5.18. Then $d_{\mathrm{H}}(P, R) \leq \sqrt{3}\epsilon$ with probability at least $2/3$.*

In order to make the error probability arbitrary small, we repeat the above construction k time to get k distributions and use a "clustering trick." The details follow.

Proof. (of Theorem 5.16) We use the construction of Lemma 5.18 k times using independent samples to obtain the distributions R_i for $1 \leq i \leq k$. From Corollary 5.19, the output distribution R_i from each repetition satisfies $d_{\mathrm{H}}(P, R_i) \leq \sqrt{3}\epsilon$ with probability at least $2/3$. For a given R_i if this event succeeds we call R_i a 'good' distribution. We use the following process to choose the final distribution R from R_is:

AMPLIFY$(R_1, \ldots, R_k)$

1. $d_{ij} \leftarrow d_{\mathrm{H}}(R_i, R_j)$ for every $1 \leq i, j \leq k$
2. $count_i \leftarrow |\{j | d_{ij} \leq 2\sqrt{3}\epsilon\}|$
3. $R \leftarrow R_{i^*}$ where i^* is the least i such that $count_i \geq 7k/12$
4. Output R

□

5.3.4 Learning product distributions with very high probability

We start with an algorithm for learning a product distribution in Hellinger distance.

Lemma 5.20. *Let $P = \prod_{i=1}^{n} P_i$ be a product distribution over Σ^n. Then there is a $O(|\Sigma| \log |\Sigma| n^2 \log^2(n/\delta)/\epsilon^2)$ time algorithm that takes $O(n|\Sigma| \log(n/\delta)/\epsilon^2)$ samples from P and produces a product distribution $R = \Pi_{i=1}^{n} R_i$ such that $d_{\mathrm{H}}(P, R) \leq \epsilon$ with probability at least $(1 - \delta)$.*

Proof. We learn P component wise, ensuring $d_{\mathrm{H}}(P_i, R_i) \leq \epsilon/\sqrt{n}$ using the algorithm of Theorem 5.16, with probability at least $(1 - \delta/n)$, for every i. Let $\ell = |\Sigma|$. By

a union bound, the overall failure probability is $\leq \delta$ and the sample complexity is $O(n\ell \log(n/\delta)/\epsilon^2)$. The running time is $O(n\ell \log \ell \log^2(n/\delta)/\epsilon^2)$ for each marginal.

The Hellinger subadditivity result for product distributions (Lemma 5.11) then directly implies that $d_{\mathrm{H}}(P, R) \leq \epsilon$. □

Corollary 5.21. *Let $P = \prod_{i=1}^{n} P_i$ be a product distribution over Σ^n. Then there is a $O(|\Sigma| \log |\Sigma| n^2 \log^2(n/\delta)/\epsilon^2)$ time algorithm that takes $O(n|\Sigma| \log(n/\delta)/\epsilon^2)$ samples from P and produces a product distribution $R = \Pi_{i=1}^{n} R_i$ such that $d_{\mathrm{TV}}(P, R) \leq \epsilon$ with probability at least $(1 - \delta)$.*

Proof. Follows from $d_{\mathrm{TV}}(P, R) \leq \sqrt{2} d_{\mathrm{H}}(P, R)$ and Lemma 5.20. □

Remark A d_{TV}-learning algorithm with success probability 2/3 for product distributions over $\{0, 1\}^n$ with sample complexity $O(n \log n/\epsilon^2)$ follows from [19] (Section A.1). Corollary 5.21 extends it for the case when the success probability is $1 - \delta$ and the support is Σ^n. It goes through a d_{H}-learning algorithm using Theorem 5.16 and d_{H}-subadditivity. Going through a d_{TV}-learning algorithm and d_{TV}-subadditivity ($d_{\mathrm{TV}}(P, R) \leq \sum_i d_{\mathrm{TV}}(P_i, R_i)$) would give a sample complexity of $O(n^2(|\Sigma| + \log(n/\delta))/\epsilon^2)$.

5.3.4.1 2-sample Tester via Learning

We need the following localization equality result for efficiently computing the Hellinger distance of a pair of known product distributions.

Lemma 5.22. *Let $P = \prod_{i=1}^{n} P_i$ and $Q = \prod_{i=1}^{n} Q_i$ be two distributions over Σ^n. It holds that $1 - d_{\mathrm{H}}^2(P, Q) = \Pi_{i=1}^{n}(1 - d_{\mathrm{H}}^2(P_i, Q_i))$.*

Proof. Using Definition 4.1 for $d^2_{\mathrm{H}}(\cdot,\cdot)$ distance:

$$1 - d^2_{\mathrm{H}}(P,Q) = \sum_{i_1 \in \Sigma} \sqrt{P_1(i_1)Q_1(i_1)} \cdots \sum_{i_n \in \Sigma} \sqrt{P_n(i_n)Q_n(i_n)} = \prod_{i=1}^{n} (1 - d^2_{\mathrm{H}}(P_i, Q_i))$$

□

Proof. (of Theorem 5.3) Given the two unknown distributions P and Q we first learn them as R and S, satisfying $\sqrt{2}d_{\mathrm{H}}(P,R) \leq \epsilon/12$ and $\sqrt{2}d_{\mathrm{H}}(Q,S) \leq \epsilon/12$, using the algorithm of Lemma 5.20. It remains to distinguish $\sqrt{2}d_{\mathrm{H}}(R,S) \leq \epsilon/2$ versus $\sqrt{2}d_{\mathrm{H}}(R,S) > 5\epsilon/6$. We compute $d_{\mathrm{H}}(R,S)$ exactly using Lemma 5.22. □

Remark. The technique introduced in this section could give an analogous tolerant tester for any distance *dist* for which the following properties hold: 1) it satisfies triangle inequality, 2) it has a localization equality and subadditivity as in Lemma 5.22 and Lemma 5.11 respectively, and 3) it has an efficient learning algorithm for small support size distributions.

5.3.5 Non-tolerant 2-sample testers for product distributions over Σ^n

5.3.5.1 In Hellinger distance

A non-tolerant tester for 2-sample testing of product distributions was given in [19]. Their tester distinguishes $P = Q$ from $d_{\mathrm{TV}}(P,Q) \geq \epsilon$ with sample complexity $O(\max\{n^{3/4}/\epsilon, \sqrt{n}/\epsilon^2\})$. Using the relation, $d_{\mathrm{TV}} \geq d^2_{\mathrm{H}}$ from Lemma 5.1, we immediately get a tester for distinguishing $P = Q$ from $d_{\mathrm{H}}(P,Q) \geq \epsilon$, with sample complexity $O(\max\{n^{3/4}/\epsilon^2, \sqrt{n}/\epsilon^4\})$. In the following we show an improved tester with sample complexity $O(n^{3/4}/\epsilon^2)$.

Algorithm 6: Given samples from two unknown distributions $P = P_1 \times \cdots \times P_n$ and $Q = Q_1 \times \cdots \times Q_n$ over Σ^n, decide $P = Q$ ('yes') versus $d_{\mathrm{H}}(P, Q) \geq \epsilon$ ('no'). Let $\ell = |\Sigma|$.

`/* Approximately identify heavy and light partitions */`

Take m samples from P and Q. Let $U' \subseteq [n] \times [\ell]$ be the set of indices (i, j), such that at least one sample from either P or Q has hit symbol $j \in \Sigma$ in the coordinate i;

Let $V' = [n] \times [\ell] \setminus U'$;

`/* Poisson sampling */`

For each $i \in [n]$, sample $M_i \sim \mathrm{Poi}(m)$ independently

For each $i \in [n]$, sample $M'_i \sim \mathrm{Poi}(m)$ independently

Let $M = \max_i\{M_i\}$ and $M' = \max_i\{M'_i\}$

If $\max\{M, M'\} \geq 2m$ output 'no'

Take M samples $X^1, \ldots, X^M$ from P

Take M' samples $Y^1, \ldots, Y^{M'}$ from Q

For every (i, j), let W_{ij} be the number of occurrences of symbol $j \in \Sigma$ in the i-th coordinate of the sample subset $X^1, \ldots, X^{M_i}$

For every (i, j), let V_{ij} be the number of occurrences of symbol $j \in \Sigma$ in the i-th coordinate of the sample subset$Y^1, \ldots, Y^{M'_i}$

`/* Test the heavy partition */`

$W_{heavy} = \sum_{(i,j)\in U'} \frac{(W_{ij}-V_{ij})^2-(W_{ij}+V_{ij})}{(W_{ij}+V_{ij})}$

If $W_{heavy} > m\epsilon^2/20$ output 'no'

`/* Test the light partition */`

$W_{light} = \sum_{(i,j)\in V'}(W_{ij} - V_{ij})^2 - (W_{ij} + V_{ij})$

If $W_{light} > m^2\epsilon^4/2000n\ell$ output 'no'

Output 'yes'

Let $P = \prod_{i=1}^{n} P_i$ and $Q = \prod_{i=1}^{n} Q_i$, with $P_i = \langle p_{i1}, \ldots, p_{i\ell} \rangle$ and $Q_i = \langle q_{i1}, \ldots, q_{i\ell} \rangle$ as probability vectors. We assume $\min_{i,j} p_{ij} \geq \epsilon^2/50n\ell$ and $\min_{i,j} q_{ij} \geq \epsilon^2/50n\ell$, without loss of generality using the reduction of Lemma 5.13.

Analysis of Algorithm 6 can be divided into two cases: 'heavy' and 'light'. Let $V \subseteq [n] \times [\ell]$ be the 'light' set of indices (i, j), such that $\max\{p_{ij}, q_{ij}\} < 1/m$. The remaining indices in $U = [n] \times [\ell] \setminus V$ are 'heavy'. The following important lemma shows that for each case, a certain sum must deviate from zero substantially, for the 'no' class.

Lemma 5.23. *Suppose* $d_{\mathrm{H}}(P, Q) \geq \epsilon$. *Suppose* $\min_{i,j} p_{ij} \geq \epsilon^2/50n\ell$ *and* $\min_{i,j} q_{ij} \geq \epsilon^2/50n\ell$. *Then at least one of the following two must hold: 1)* $\sum_{(i,j)\in V} (p_{ij} - q_{ij})^2 \geq \epsilon^4/50n\ell$, *2)* $\sum_{(i,j)\in U} \frac{(p_{ij}-q_{ij})^2}{p_{ij}+q_{ij}} \geq \epsilon^2$.

Proof. If $d_{\mathrm{H}}(P, Q) \geq \epsilon$, by the subadditivity of squared Hellinger distance from Lemma 5.11, we get $\sum_i d^2_{\mathrm{H}}(P_i, Q_i) \geq \epsilon^2$. We use the following Fact,

Fact 5.24. *[Proposition 2.3 from [32]] For two distributions* $P = \{p_1, \ldots, p_\ell\}$ *and* $Q = \{q_1, \ldots, q_\ell\}$, $\sum_j \frac{(p_j-q_j)^2}{p_j+q_j} \geq 2d^2_{\mathrm{H}}(P, Q)$.

to get $\sum_{i=1}^{n} \sum_{j=1}^{\ell} \frac{(p_{ij}-q_{ij})^2}{p_{ij}+q_{ij}} \geq 2 \sum_i d^2_{\mathrm{H}}(P_i, Q_i) \geq 2\epsilon^2$. It follows at least one of 1) $\sum_{(i,j)\in U} \frac{(p_{ij}-q_{ij})^2}{p_{ij}+q_{ij}}$ or 2) $\sum_{(i,j)\in V} \frac{(p_{ij}-q_{ij})^2}{p_{ij}+q_{ij}}$ is at least ϵ^2.

In the second case, $\sum_{(i,j)\in V} (p_{ij} - q_{ij})^2 = \sum_{(i,j)\in V} (\sqrt{p_{ij}} - \sqrt{q_{ij}})^2 (\sqrt{p_{ij}} + \sqrt{q_{ij}})^2 \geq \epsilon^2/25n\ell \cdot \sum_{(i,j)\in V} (\sqrt{p_{ij}} - \sqrt{q_{ij}})^2 \geq \epsilon^2/25n\ell \cdot \sum_{(i,j)\in V} \frac{(p_{ij}-q_{ij})^2}{(\sqrt{p_{ij}}+\sqrt{q_{ij}})^2} \geq \epsilon^2/50n\ell \cdot \sum_{(i,j)\in V} \frac{(p_{ij}-q_{ij})^2}{p_{ij}+q_{ij}} \geq \epsilon^4/50n\ell$. □

The following lemma shows U' (V'), as obtained in Lines 1-2 of Algorithm 6, could be an acceptable proxy for the heavy (light) partition U (V).

Lemma 5.25. *Let U', V' be as in Algorithm 6. Let $m = \Omega(\sqrt{n\ell}/\epsilon^2)$ for some sufficiently large constant. Then with probability at least 0.63 in each case, the following holds:*

1. $\sum_{(i,j)\in U} \frac{(p_{ij}-q_{ij})^2}{p_{ij}+q_{ij}} \geq \epsilon^2$ *implies* $\sum_{(i,j)\in U\cap U'} \frac{(p_{ij}-q_{ij})^2}{p_{ij}+q_{ij}} \geq 3\epsilon^2/10$
2. $\sum_{(i,j)\in V}(p_{ij}-q_{ij})^2 \geq \epsilon^4/50n\ell$ *implies* $\sum_{(i,j)\in V'}(p_{ij}-q_{ij})^2 \geq \epsilon^4/1000n\ell$

Proof. Note that $(i,j) \in V'$ with probability $= (1-p_{ij})^m(1-q_{ij})^m$, and $(i,j) \in U'$ with the remaining probability.

(*Proof of 1:*) Let $U'' = U \cap U'$. Suppose there exists $(i,j) \in U$ such that $\frac{(p_{ij}-q_{ij})^2}{p_{ij}+q_{ij}} \geq \epsilon^2/50$. Then this $(i,j) \in U''$ with probability $1-(1-p_{ij})^m(1-q_{ij})^m \geq 1-(1-1/m)^m \geq 0.63$, in which case the result follows. Otherwise, we consider sum of the independent random variables, $S = \sum_{(i,j)} 1_{(i,j)\in U''} \frac{50(p_{ij}-q_{ij})^2}{\epsilon^2(p_{ij}+q_{ij})}$, each of which is in $[0,1]$. $\mathrm{E}[S] = \sum_{(i,j)\in U}(1-(1-p_{ij})^m(1-q_{ij})^m)\frac{50(p_{ij}-q_{ij})^2}{\epsilon^2(p_{ij}+q_{ij})} \geq 31$. We apply Chernoff's bound to get $S \geq 15$ except for probability at most $\exp(-31/8)$.

(*Proof of 2:*) Let $V'' = V \cap V'$. We consider sum of the independent random variables, $S = \sum_{(i,j)} 1_{(i,j)\in V''} m^2(p_{ij}-q_{ij})^2$, each of which is in $[0,1]$. $\mathrm{E}[S] = \sum_{(i,j)\in V}(1-p_{ij})^m(1-q_{ij})^m m^2(p_{ij}-q_{ij})^2 > (1-1/m)^{2m}\sum_{(i,j)\in V} m^2(p_{ij}-q_{ij})^2 \geq m^2\epsilon^4/500n\ell$, for $m \geq 4$. We apply Chernoff's bound to get $S \geq m^2\epsilon^4/1000n\ell$ except for probability at most $\exp(-m^2\epsilon^4/4000n\ell)$. □

Combining Lemma 5.23 and Lemma 5.25, we get for the 'no' case, one of the two conditions of Lemma 5.25 must hold. Algorithm 6 uses the two tests W_{heavy} and W_{light} to check these two conditions. To analyze them, we use certain important results from [19], assuming $W_{ij} \sim \mathrm{Poi}(p_{ij})$ and $V_{ij} \sim \mathrm{Poi}(q_{ij})$ for every i,j, which holds due to Poisson sampling. We assume the check of Line 7 goes through except 1/50 probability, using the concentration of Poisson distribution.

Analysis of W_{heavy}

Lemma 5.26 (Obtained from Claims 37 and 38 of [19])**.** *If $P = Q$ then $\mathrm{E}[W_{heavy}] = 0$. If $\sum_{(i,j)\in U\cap U'} \frac{(p_{ij}-q_{ij})^2}{p_{ij}+q_{ij}} \geq 3\epsilon^2/10$ then $\mathrm{E}[W_{heavy}] \geq m\epsilon^2/10$. In both cases $\mathrm{Var}[W_{heavy}] \leq 7n\ell + 15\mathrm{E}[W_{heavy}]$*

By the application of Chebyshev's inequality we get the following.

Lemma 5.27. *Let $m = \Omega(\sqrt{n\ell}/\epsilon^2)$ for a sufficiently large constant. Then the following holds except for probability $\leq 1/25$ in each case,*

1. *$P = Q$ implies $W_{heavy} \leq m\epsilon^2/20$*
2. *$\sum_{(i,j)\in U\cap U'} \frac{(p_{ij}-q_{ij})^2}{p_{ij}+q_{ij}} \geq 3\epsilon^2/10$ implies $W_{heavy} > m\epsilon^2/20$.*

Analysis of W_{light}

Lemma 5.28 (Obtained from Proposition 6 of [22] and Claim 35 of [19])**.** $\mathrm{E}[W_{light}] = m^2 \sum_{(i,j)\in V'} (p_{ij} - q_{ij})^2$ *and* $\mathrm{Var}[W_{light}] \leq 80m^3\sqrt{b}\sum_{(i,j)\in V'} (p_{ij} - q_{ij})^2 + 8m^2 b$*, where* $b = \max\{\sum_{(i,j)\in V'} p_{ij}^2, \sum_{(i,j)\in V'} q_{ij}^2\}$*. Furthermore, $b \leq 50n\ell/m^2$ for a sufficiently large m, except for probability at most $1/50$.*

By the application of Chebyshev's inequality we get the following.

Lemma 5.29. *Let $m = \Omega((n\ell)^{3/4}/\epsilon^2)$ for a sufficiently large constant. Then the following holds except for probability $\leq 1/50$ in each case,*

1. *$P = Q$ implies $W_{light} \leq m^2\epsilon^4/2000n\ell$*
2. *$\sum_{(i,j)\in V'}(p_{ij} - q_{ij})^2 \geq \epsilon^4/1000n\ell$ implies $W_{light} > m^2\epsilon^4/2000n\ell$.*

Together we get $O((n\ell)^{3/4}/\epsilon^2)$ samples are enough to distinguish $P = Q$ versus $d_{\mathrm{H}}(P, Q) \geq \epsilon$.

Theorem 5.6. *(Exact-versus-d_{H} closeness tester) Let $P = \prod_{i=1}^{n} P_i$ and $Q = \prod_{i=1}^{n} Q_i$ be two unknown distributions, both over the common sample space Σ^n. Then testing $P = Q$ versus $\sqrt{2}d_{\mathrm{H}}(P, Q) > \epsilon$ can be performed with $m = O((n|\Sigma|)^{3/4}/\epsilon^2)$ samples, in time $O(mn)$, with success probability at least $51/100$.*

5.3.5.2 In total variation distance

A 2-sample tester for distinguishing $P = Q$ from $d_{\mathrm{TV}}(P, Q) \geq \sqrt{2}\epsilon$, for product distributions over $\{0, 1\}^n$, was given in [19] with sample complexity $O(\sqrt{n}/\epsilon^2, \max\{n^{3/4}/\epsilon\})$. In the following, we generalize this result for product distributions over Σ^n with sample complexity $m = O(\max\{\sqrt{n|\Sigma|}/\epsilon^2, (n|\Sigma|)^{3/4}/\epsilon^{3/2}\})$. Let $\ell = |\Sigma|$. We assume $\min_{i,j} p_{ij} \geq \epsilon/50n\ell$ and $\min_{i,j} q_{ij} \geq \epsilon/50n\ell$ without loss of generality, using the reduction given in [19, 33]. Let $V \subseteq [n] \times [\ell]$ be the set of indices (i, j), such that $\max\{p_{ij}, q_{ij}\} < 1/m$. Let $U = [n] \times [\ell] \setminus V$.

Lemma 5.30. *Suppose $d_{\mathrm{TV}}(P, Q) \geq \sqrt{2}\epsilon$. Suppose $\min_{i,j} p_{ij} \geq \epsilon/50n\ell$ and $\min_{i,j} q_{ij} \geq \epsilon/50n\ell$. Then at least one of the following two must hold: 1) $\sum_{(i,j)\in V}(p_{ij} - q_{ij})^2 \geq \epsilon^3/50n\ell$, 2) $\sum_{(i,j)\in U} \frac{(p_{ij}-q_{ij})^2}{p_{ij}+q_{ij}} \geq \epsilon^2$.*

Proof. $d_{\mathrm{TV}}(P, Q) \geq \sqrt{2}\epsilon$ gives $d_{\mathrm{H}}(P, Q) \geq \epsilon$. If $d_{\mathrm{H}}(P, Q) \geq \epsilon$, by the subadditivity of squared Hellinger distance from Lemma 5.11, we get $\sum_i d^2_{\mathrm{H}}(P_i, Q_i) \geq \epsilon^2$. This gives $\sum_{i=1}^{n}\sum_{j=1}^{\ell} \frac{(p_{ij}-q_{ij})^2}{p_{ij}+q_{ij}} \geq 2\sum_i d^2_{\mathrm{H}}(P_i, Q_i) \geq 2\epsilon^2$, using Fact 5.24. It follows at least one of 1) $\sum_{(i,j)\in U} \frac{(p_{ij}-q_{ij})^2}{p_{ij}+q_{ij}}$ or 2) $\sum_{(i,j)\in V} \frac{(p_{ij}-q_{ij})^2}{p_{ij}+q_{ij}}$ is at least ϵ^2.

In the second case, $\sum_{(i,j)\in V}(p_{ij} - q_{ij})^2 = \sum_{(i,j)\in V}(\sqrt{p_{ij}} - \sqrt{q_{ij}})^2(\sqrt{p_{ij}} + \sqrt{q_{ij}})^2 \geq \epsilon/25n\ell \cdot \sum_{(i,j)\in V}(\sqrt{p_{ij}} - \sqrt{q_{ij}})^2 \geq \epsilon/25n\ell \cdot \sum_{(i,j)\in V} \frac{(p_{ij}-q_{ij})^2}{(\sqrt{p_{ij}}+\sqrt{q_{ij}})^2} \geq \epsilon/50n\ell \cdot \sum_{(i,j)\in V} \frac{(p_{ij}-q_{ij})^2}{p_{ij}+q_{ij}} \geq \epsilon^3/50n\ell$

□

We skip the rest of the details of the algorithm and its analysis since it closely follows that of Section 5.3.5.1. We identify the partitions U and V approximately by checking which indices are hit in m samples. For $m = \Omega(\sqrt{n\ell}/\epsilon^{3/2})$, this approximation is acceptable using a result similar to Lemma 5.25. Lemmas 5.26, 5.27 and 5.28 are as before. Only in Lemma 5.29, the threshold for the light part changes to $m^2\epsilon^3/2000n\ell$, and the sample complexity for the light part changes to $m = \Omega((n\ell)^{3/4}/\epsilon^{3/2})$.

Theorem 5.7. *(Exact-versus-d_{TV} closeness tester) Let $P = \prod_{i=1}^{n} P_i$ and $Q = \prod_{i=1}^{n} Q_i$ be two unknown distributions, both over the common sample space Σ^n. Then testing $P = Q$ versus $d_{\mathrm{TV}}(P, Q) > \epsilon$ can be performed with $m = O(\max\{\sqrt{n|\Sigma|}/\epsilon^2, (n|\Sigma|)^{3/4}/\epsilon^{3/2}\})$ samples, in time $O(mn)$, with success probability at least $51/100$.*

5.3.6 d_{TV} tolerant 2-sample tester for product distributions

In this section, we discuss an exponential time algorithm for checking $d_{\mathrm{TV}}(P, Q) \leq \epsilon/3$ versus $d_{\mathrm{TV}}(P, Q) > \epsilon$, for two product distributions P and Q over Σ^n. We use the testing-by-learning framework. We first learn P and Q as R and S, satisfying $d_{\mathrm{TV}}(P, R) \leq \epsilon/12$ and $d_{\mathrm{TV}}(Q, S) \leq \epsilon/12$, using $O(|\Sigma| n \log n/\epsilon^2)$ samples from the algorithm of Corollary 5.21. It remains to distinguish $d_{\mathrm{TV}}(R, S) \leq \epsilon/2$ versus $d_{\mathrm{TV}}(R, S) > 5\epsilon/6$, which we perform by evaluating $d_{\mathrm{TV}}(R, S)$ exactly in $O(|\Sigma|^n)$ time.

5.4 Lower bounds

In this section, we give lower bounds for tolerant testing of product distributions. Our lower bounds use a reduction from testing the class of unstructured distributions over n items to testing the class of product distributions over $\{0, 1\}^n$, given by [19] (Section

5.3 of their paper). However, in order to apply this reduction, we need to establish certain new bounds relating distances in the unstructured setting to the setting of product distribution. We first define how to construct a product distribution from the corresponding unstructured distribution. In particular, for a $\delta < 1$, this construction produces a product distribution $F_\delta(P)$ over $\{0,1\}^n$ from a given distribution P over n symbols.

Definition 5.31. *(Construction of $F_\delta(P)$) Let P be a distribution over a sample space of n items and $0 < \delta \leq 1$ be a constant. Let S be a random set of* $\mathrm{Poi}(\delta)$ *samples from P. For every item $i \in [n]$, let x_i be the indicator variable such that $x_i = 1$ iff i appears in S. Let $F_\delta(P)$ be the joint distribution of $\langle x_1, x_2, \ldots, x_n \rangle$ over the sample space $\{0,1\}^n$.*

The following property can be observed using the property of Poissonization.

Fact 5.32. *Let P be a distribution over a sample space of n items with probability vector $\langle p_1, p_2, \ldots, p_n \rangle$ and $0 < \delta \leq 1$ be a constant. Then $F_\delta(P)$ is a product distribution such that $F_\delta(P) = \prod_{i=1}^n F_\delta(P_i)$ where $F_\delta(P_i) \sim \mathrm{Bern}(1 - e^{-\delta p_i})$.*

We use the following crucial lemma.

Lemma 5.33. *For any $0 < \delta \leq 1$ and distributions P, Q, $d_{\mathrm{TV}}(F_\delta(P), F_\delta(Q)) \geq \delta e^{-\delta} d_{\mathrm{TV}}(P, Q)$, with equality holding iff $P = Q$.*

Proof. Let $P = \langle p_1, \ldots, p_i, \ldots, p_n \rangle$ and $Q = \langle q_1, \ldots, q_i, \ldots, q_n \rangle$ be the probability

values of P and Q.

$$\begin{aligned}
d_{\mathrm{TV}}(F_\delta(P), F_\delta(Q)) &= \sum_{x\in\{0,1\}^n} |F_\delta(P)(x) - F_\delta(Q)(x)| \\
&\geq \sum_{i=1}^{n} |d_{\mathrm{TV}}(F_\delta(P)(e_i) - F_\delta(Q))(e_i)| \\
&\qquad\qquad \text{(unit vector } e_i \text{ has } i \text{ th value 1)} \\
&= \sum_{i=1}^{n} |(1-e^{-\delta p_i})\Pi_{j\neq i} e^{-\delta p_j} - (1-e^{-\delta q_i})\Pi_{j\neq i} e^{-\delta q_j}| \\
&= \sum_{i=1}^{n} e^{-\delta} |e^{\delta p_i} - e^{\delta q_i}| \qquad\qquad (\text{Since } \Pi_j e^{-\delta p_j} = \Pi_j e^{-\delta q_j} = e^{-\delta}) \\
&= e^{-\delta} \sum_{i=1}^{n} |\delta(p_i - q_i) + \delta^2(p_i^2 - q_i^2)/2! + \cdots + \delta^j(p_i^j - q_i^j)/j! + \ldots|
\end{aligned}$$

We analyze the expression under modulus under two cases: 1) if $p_i > q_i$, it is $> \delta(p_i - q_i)$, 2) if $p_i < q_i$, it is $> \delta(q_i - p_i)$.

$$\begin{aligned}
d_{\mathrm{TV}}(F_\delta(P), F_\delta(Q)) &\geq e^{-\delta} \sum_{i=1}^{n} |\delta(p_i - q_i)| \\
&= \delta e^{-\delta} d_{\mathrm{TV}}(P, Q)
\end{aligned}$$

□

5.4.1 Hardness of χ^2-tolerance for 2-sample testing of product distributions

Here we show that for two unknown product distribution P, Q over $\{0,1\}^n$, distinguishing $d_{\chi^2}(P,Q) \leq \epsilon^2/9$ versus $d_{\mathrm{TV}}(P,Q) > \epsilon$, for a constant ϵ, cannot be decided in general with a truly sublinear sample complexity. We use a reduction to the following difficult problem, for hardness of χ^2-tolerance for 2-sample testing of unstructured

distributions over n items, given in [32]. We restate the theorem with changes in the constants.

Theorem 5.34. *There exists a constant $0 < \epsilon < 1$ and three distributions P^{yes}, P^{no} and Q, each over the sample space $[n]$ such that: (1) $d_{\chi^2}(P^{yes}, Q) \leq \epsilon^2/216$, whereas $d_{\mathrm{TV}}(P^{no}, Q) \geq \epsilon$ and (2) given only sample accesses to one of P^{yes} or P^{no}, and Q, distinguishing P^{yes} versus P^{no} with probability $> 4/5$, requires $\Omega(n/\log n)$ samples.*

We use the following important property about the χ^2-distance between the reduced distributions.

Lemma 5.35. $d_{\chi^2}(F_\delta(P), F_\delta(Q)) \leq \exp(4\delta\chi^2(P, Q)) - 1$, *for any* $0 < \delta \leq 1$.

Proof. From Fact 5.32, both $F_\delta(P)$ and $F_\delta(Q)$ are product distributions, the distribution of the i-th component being $F_\delta(P_i)$ and $F_\delta(Q_i)$ respectively. Let $P = \langle p_1, p_2, \ldots, p_n \rangle$ and $Q = \langle q_1, q_2, \ldots, q_n \rangle$. Then $F_\delta(P_i) \sim \mathrm{Bern}(1 - e^{-\delta p_i})$ and $F_\delta(Q_i) \sim$

$\text{Bern}(1 - e^{-\delta q_i})$.

$$
\begin{aligned}
d_{\chi^2}(F_\delta(P), F_\delta(Q)) &= \prod_i (1 + d_{\chi^2}(F_\delta(P_i), F_\delta(Q_i))) - 1 && \text{(From Lemma 5.8)} \\
&\leq \prod_i \exp(d_{\chi^2}(F_\delta(P_i), F_\delta(Q_i))) - 1 \\
& && \text{(Since } e^x \geq (1+x) \text{ for } x \geq 0) \\
&= \exp(\sum_i d_{\chi^2}(F_\delta(P_i), F_\delta(Q_i))) - 1 \\
&= \exp\left(\sum_i (e^{-\delta p_i} - e^{-\delta q_i})^2 \left(\frac{1}{e^{-\delta q_i}} + \frac{1}{1 - e^{-\delta q_i}}\right)\right) - 1 \\
& \text{(Since } F_\delta(P_i^{yes}) \sim \text{Bern}(1 - e^{-\delta p_i}) \text{ and } F_\delta(Q_i) \sim \text{Bern}(1 - e^{-\delta q_i})) \\
&= \exp(\sum_i (e^{-\delta p_i} - e^{-\delta q_i})^2 / e^{-\delta q_i}(1 - e^{-\delta q_i})) - 1 \\
&= \exp(\sum_i (e^{\delta(q_i - p_i)} - 1)^2 / (e^{\delta q_i} - 1)) - 1 \\
&= \exp(\sum_i (e^{\delta(q_i - p_i)} - 1)^2 / (e^{\delta q_i} - 1)) - 1 \\
&\leq \exp(\sum_i (2\delta(p_i - q_i))^2 / \delta q_i)) - 1 \\
& \qquad \text{(Since } (e^x - 1) \geq x \text{ and } (|e^x - 1| \leq 2|x| \text{ for } 0 < |x| < 1) \\
&= \exp(4\delta\chi^2(P, Q)) - 1
\end{aligned}
$$

□

We are set to present the main lower bound result of this section.

Theorem 5.4. *(d_{χ^2}-versus-d_{TV} closeness testing lower bound) There exists a constant $0 < \epsilon < 1$ and three product distributions F^{yes}, F^{no} and F, each over the sample space $\{0,1\}^n$ such that $d_{\chi^2}(F^{yes}, F) \leq \epsilon^2/9$, whereas $d_{\text{TV}}(F^{no}, F) > \epsilon$, and given only*

sample accesses to F^{yes}, F^{no} and F, distinguishing F^{yes} versus F^{no} with probability $> 2/3$, requires $\Omega(n/\log n)$ samples.

Proof. We start with the distributions P^{yes}, P^{no} and Q from the hardness Theorem 5.34. Then $d_{\chi^2}(P^{yes}, Q) \leq \epsilon^2/216$ and $d_{\mathrm{TV}}(P^{no}, Q) \geq \epsilon$ for some constant $0 < \epsilon < 1$. We apply the reduction of Definition 5.31, with $\delta = 1/3$ to these three distributions. Then from Lemma 5.33 and Lemma 5.35 we get the following two inequalities:

- $d_{\chi^2}(F_\delta(P^{yes}), F_\delta(Q)) \leq \exp(4\chi^2(P^{yes}, Q)/3) - 1 < \epsilon^2/160$.
- $d_{\mathrm{TV}}(F_\delta(P^{no}), F_\delta(Q)) > (1/3e^{1/3})\epsilon$.

It follows if we can distinguish $d_{\chi^2}(F_\delta(P^{yes}), F_\delta(Q)) \leq \epsilon^2/160$ versus $d_{\mathrm{TV}}(F_\delta(P^{no}), F_\delta(Q)) > (1/3e^{1/3})\epsilon$, then we are able to decide the hard instance of Theorem 5.34. Moreover, in order to simulate each sample from the distribution $F_{1/2}(P)$, we need $\mathrm{Poi}(1/2)$ samples from P. So, if we need m samples in total, from the additive property of the Poisson distribution, we need $\mathrm{Poi}(m/2) = O(m)$ samples from P in total, except for $\exp(-m)$ probability. It follows, if we can decide the problem given in the theorem statement in $o(n/\log n)$ samples, we can decide the hard problem of Theorem 5.34 in $o(n/\log n)$ samples as well. This leads to a contradiction. Replacing the constant ϵ by $3e^{1/3}\epsilon_1$, we get Theorem 5.4. □

5.4.2 Hardness of KL-tolerance for 1-sample testing of product distributions

In this section we show that for an unknown product distribution P and a known product distribution Q over $\{0,1\}^n$, distinguishing $d_{\mathrm{KL}}(P, Q) \leq \epsilon^2/9$ versus $d_{\mathrm{TV}}(P, Q) > \epsilon$, for a constant ϵ, cannot be decided in general with a truly sublinear sample complexity. We use a reduction to the following hardness result, for KL-tolerance for 1-sample

testing of unstructured distributions over n items, given in [32]. We restate the theorem with changes in the constants. For a probability distribution $P = \langle p_1, p_2, \ldots, p_n \rangle$ over n items, $||P||_2^2 = \sum_i p_i^2$.

Theorem 5.36. *There exists a constant $0 < \epsilon < 1$ and three distributions P^{yes}, P^{no} and Q, each over the sample space $[n]$ such that: (1) $d_{\mathrm{KL}}(P^{yes}, Q) \leq \epsilon^2/216$, whereas $d_{\mathrm{TV}}(P^{no}, Q) \geq \epsilon$, (2) $||P^{yes}||_2^2 = O(\log^2 n/n)$, and (3) given only sample accesses to one of P^{yes} or P^{no}, and complete knowledge of Q, distinguishing P^{yes} versus P^{no} with probability $> 4/5$, requires $\Omega(n/\log n)$ samples.*

Proof. The proof of this Theorem appears in [32], in Theorem 6.2 of this version, except the fact $||P^{yes}||_2^2 = O(\log^2 n/n)$ is not explicitly claimed. We prove this claim in the following, by observing from the original construction given in the paper by Valiant and Valiant [63].

The hard distribution P^{yes} is the distribution $p^-_{\log k,\phi}$ as defined in Definition 12 of [63]. We use the following facts about this distribution $p^-_{\log k,\phi}$, given in Fact 11, Definition 12 and (in the end of the second paragraph in the proof of) Lemma 13 in [63]:

- ϕ is a small enough constant
- The support size n and the parameter k are related as $n = 32k \log k/\phi$
- The 'un-normalized' mass at each point is $x/32k$, where $j = \log k$ and $x \leq 4j$
- The 'normalizing constant' c_2 (which makes the probability values sum up to 1) is at most ϕ/j where $j = \log k$

From these facts we conclude each probability mass is $c_2 \cdot x/32k \leq \phi/8k$, where $n = 32k \log k/\phi$ for some constant ϕ. Hence, $||P^{yes}||_2^2 \leq \phi^2/64k^2 \cdot 32k \log k/\phi = \phi \log k/2k = O(\log^2 n/n)$. □

We use the reduction given in Definition 5.31. We establish the following lemma, relating KL distances between the original and the reduced distributions.

Lemma 5.37. *$d_{\mathrm{KL}}(F_\delta(P), F_\delta(Q)) \leq \delta d_{\mathrm{KL}}(P, Q) + \delta^2 ||P||_2^2$, for any $0 < \delta \leq 1$.*

Proof. We use the following fact about the KL-distance between two product distributions.

Fact 5.38. *For two distributions $P = \prod_{i=1}^n P_i$ and $Q = \prod_{i=1}^n Q_i$ over the same discrete sample space, it holds that $d_{\mathrm{KL}}(P, Q) = \sum_{i=1}^n d_{\mathrm{KL}}(P_i, Q_i)$.*

From Fact 5.32, both $F_\delta(P)$ and $F_\delta(Q)$ are product distributions, the distribution of the i-th component being $F_\delta(P_i)$ and $F_\delta(Q_i)$ respectively. Let $P = \langle p_1, p_2, \ldots, p_n \rangle$

and $Q = \langle q_1, q_2, \ldots, q_n \rangle$. Then $F_\delta(P_i) \sim \text{Bern}(1-e^{-\delta p_i})$ and $F_\delta(Q_i) \sim \text{Bern}(1-e^{-\delta q_i})$.

$$
\begin{aligned}
d_{\mathrm{KL}}(F_\delta(P), F_\delta(Q)) &= \sum_i d_{\mathrm{KL}}(F_\delta(P_i), F_\delta(Q_i)) \\
&= \sum_i \left[(1-e^{-\delta p_i}) \ln \left(\frac{1-e^{-\delta p_i}}{1-e^{-\delta q_i}} \right) + e^{-\delta p_i} \ln \frac{e^{-\delta p_i}}{e^{-\delta q_i}} \right] \\
&= \sum_i \ln \left(\frac{1-e^{-\delta p_i}}{1-e^{-\delta q_i}} \right) + \sum_i e^{-\delta p_i} \ln \frac{e^{-\delta p_i}(1-e^{-\delta q_i})}{e^{-\delta q_i}(1-e^{-\delta p_i})} \\
&= \sum_i \ln \frac{e^{\delta q_i}(e^{\delta p_i}-1)}{e^{\delta p_i}(e^{\delta q_i}-1)} + \sum_i e^{-\delta p_i} \ln \left(\frac{e^{\delta q_i}-1}{e^{\delta p_i}-1} \right) \\
&= \sum_i \ln \frac{e^{\delta q_i}}{e^{\delta p_i}} + \sum_i \ln \left(\frac{e^{\delta p_i}-1}{e^{\delta q_i}-1} \right) + \sum_i e^{-\delta p_i} \ln \left(\frac{e^{\delta q_i}-1}{e^{\delta p_i}-1} \right) \\
&= \sum_i (q_i - p_i) + \sum_i (1-e^{-\delta p_i}) \ln \left(\frac{e^{\delta p_i}-1}{e^{\delta q_i}-1} \right) \\
&= \sum_i (1-e^{-\delta p_i}) \ln \left(\frac{e^{\delta p_i}-1}{e^{\delta q_i}-1} \right) \qquad (\text{Since } \textstyle\sum_i p_i = \sum_i q_i = 1) \\
&\leq \sum_i \delta p_i \ln \left(\frac{\delta p_i (1+\delta p_i)}{\delta q_i} \right) \\
&\qquad (\text{Since } 1-e^{-x} \leq x, \text{ and } x \leq e^x - 1 \leq x(1+x) \text{ for } x < 1) \\
&= \delta \sum_i p_i \ln \frac{p_i}{q_i} + \delta \sum_i p_i \ln(1+\delta p_i) \\
&\leq \delta d_{\mathrm{KL}}(P, Q) + \delta^2 \sum_i p_i^2 \qquad (\text{Using } \ln(1+x) \leq x)
\end{aligned}
$$

□

We present the lower bound for closeness testing of product distributions.

Theorem 5.5. *(d_{KL}-versus-d_{TV} identity testing lower bound) There exists a constant $0 < \epsilon < 1$ and three product distributions F^{yes}, F^{no} and F, each over the sample space $\{0,1\}^n$ such that $d_{\mathrm{KL}}(F^{yes}, F) \leq \epsilon^2/9$, whereas $d_{\mathrm{TV}}(F^{no}, F) > \epsilon$, and given only sample accesses to F^{yes}, F^{no}, and complete knowledge about F, distinguishing F^{yes} versus F^{no} with probability $> 2/3$, requires $\Omega(n/\log n)$ samples.*

Proof. We start with the distributions P^{yes}, P^{no} and Q from the hardness result Theorem 5.36. Then $d_{\mathrm{KL}}(P^{yes}, Q) \leq \epsilon^2/216$, $||P^{yes}||_2^2 = O(\log^2 n/n)$ and $d_{\mathrm{TV}}(P^{no}, Q) \geq \epsilon$ for some constant $0 < \epsilon < 1$. We apply the reduction of Definition 5.31, with $\delta = 1/3$ to these three distributions. Then from Lemma 5.33 and Lemma 5.37 we get the following two:

- $d_{KL}(F_\delta(P^{yes}), F_\delta(Q)) \leq \epsilon^2/160$, for any large enough n.
- $d_{\mathrm{TV}}(F_\delta(P^{no}), F_\delta(Q)) > (1/3e^{1/3})\epsilon$.

It follows if we can distinguish $d_{\mathrm{KL}}(F_\delta(P^{yes}), F_\delta(Q)) \leq \epsilon^2/160$ versus $d_{\mathrm{TV}}(F_\delta(P^{no}), F_\delta(Q)) > (1/3e^{1/3})\epsilon$, then we will be able to decide the hard instance of Theorem 5.34. Moreover, in order to simulate each sample from the distribution $F_{1/2}(P)$, we need $\mathrm{Poi}(1/2)$ samples from P. So, if we need m samples in total, from the additive property of the Poisson distribution, we need $\mathrm{Poi}(m/2) = O(m)$ samples from P in total, except for $\exp(-m)$ probability. It follows, if we can decide the problem given in the Theorem statement in $o(n/\log n)$ samples, we can decide the hard problem of Theorem 5.36 in $o(n/\log n)$ samples as well. This leads to a contradiction. Replacing the constant ϵ by $3e^{1/3}\epsilon_1$, we get Theorem 5.5. □

Chapter 6

Conclusion

In this dissertation we made progress on certain computational problems over broadly two models of computation: the streaming model for large datasets and the distribution testing model for large distributions.

In Chapter 2 we gave the first formal algorithm for estimating the number of low frequency items in a large dataset of items. Unlike previous approaches we formally analyzed the space complexities our algorithm and established its optimality. This algorithm was motivated by the bioinfomatics problem of estimating k-mer abundance histogram and gave state-of-the-art empirical performance.

In Chapter 3 we worked in the distributed variation of the streaming model and its sliding window counterpart. We looked at the problem of estimating frequency moments and clustering problems in the above models. We are not aware of any lower bounds for these problems. Hence showing optimality or better upper bounds for them would be interesting.

In Chapter 5 we studied tolerant testing problems for high dimensional product distributions. We gave near-optimal upper and lower bounds for a number of these problems. A natural question is to find more general structured classes of high-dimensional distributions for which tolerant or non-tolerant testing problems would be tractable.

Bibliography

[1] Jayadev Acharya, Constantinos Daskalakis, and Gautam Kamath. Optimal testing for properties of distributions. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3591–3599, 2015.

[2] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137 – 147, 1999.

[3] Emmanuelle Anceaume, Yann Busnel, Nicolo Rivetti, and Bruno Sericola. Identifying global icebergs in distributed streams. In *34th IEEE Symposium on Reliable Distributed Systems, SRDS 2015, Montreal, QC, Canada, September 28 - October 1, 2015*, pages 266–275, 2015.

[4] Chrisil Arackaparambil, Joshua Brody, and Amit Chakrabarti. Functional monitoring without monotonicity. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming*, pages 95–106, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[5] John Arbuthnott. An argument for divine providence, taken from the constant regularity observ'd in the births of both sexes. by dr. john arbuthnott, physitian

in ordinary to her majesty, and fellow of the college of physitians and the royal society. *Philosophical Transactions (1683-1775)*, 27:186–190, 1710.

[6] Brian Babcock, Mayur Datar, Rajeev Motwani, and Liadan O'Callaghan. Maintaining variance and k-medians over data stream windows. In *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*, pages 234–243, 2003.

[7] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *Randomization and Approximation Techniques, 6th International Workshop, RANDOM 2002, Cambridge, MA, USA, September 13-15, 2002, Proceedings*, pages 1–10, 2002.

[8] T. Batu, L. Fortnow, E. Fischer, R. Kumar, R. Rubinfeld, and P. White. Testing random variables for independence and identity. In *Proceedings of the 42Nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 442–, Washington, DC, USA, 2001. IEEE Computer Society.

[9] Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D Smith, and Patrick White. Testing closeness of discrete distributions. *Journal of the ACM (JACM)*, 60(1):4, 2013.

[10] Sairam Behera, Sutanu Gayen, Jitender S. Deogun, and N. V. Vinodchandran. Kmerestimate: A streaming algorithm for estimating k-mer counts with optimal space usage. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, BCB 2018, Washington, DC, USA, August 29 - September 01, 2018*, pages 438–447, 2018.

[11] Arnab Bhattacharyya, Sutanu Gayen, Saravanan Kandasamy, and N. V. Vinodchandran. Tolerant testing of product distributions. *Manuscript*, 2019.

[12] Avrim Blum, John Hopcroft, and Ravindran Kannan. Foundations of data science. 2019.

[13] Vladimir Braverman, Stephen R. Chestnut, Nikita Ivkin, Jelani Nelson, Zhengyu Wang, and David P. Woodruff. Bptree: An ℓ_2 heavy hitters algorithm using constant memory. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 361–376, 2017.

[14] Vladimir Braverman, Harry Lang, Keith Levin, and Morteza Monemizadeh. Clustering on sliding windows in polylogarithmic space. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, pages 350–364, 2015.

[15] Vladimir Braverman, Harry Lang, Keith Levin, and Morteza Monemizadeh. Clustering problems on sliding windows. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1374–1390, 2016.

[16] Vladimir Braverman, Adam Meyerson, Rafail Ostrovsky, Alan Roytman, Michael Shindler, and Brian Tagiku. Streaming k-means on well-clusterable data. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 26–40, 2011.

[17] Vladimir Braverman and Rafail Ostrovsky. Effective computations on sliding windows. *SIAM J. Comput.*, 39(6):2113–2131, 2010.

[18] Clément L. Canonne. A survey on distribution testing: Your data is big. but is it blue? *Electronic Colloquium on Computational Complexity (ECCC)*, 22:63, 2015.

[19] Clément L. Canonne, Ilias Diakonikolas, Daniel M. Kane, and Alistair Stewart. Testing bayesian networks. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 370–448, 2017.

[20] Cern. Opendata cern. `http://opendata.cern.ch`. Accessed: November 16, 2018.

[21] Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-hamming-distance. *SIAM J. Comput.*, 41(5):1299–1317, 2012.

[22] Siu-On Chan, Ilias Diakonikolas, Paul Valiant, and Gregory Valiant. Optimal algorithms for testing closeness of discrete distributions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1193–1203. SIAM, 2014.

[23] Moses Charikar, Liadan O'Callaghan, and Rina Panigrahy. Better streaming algorithms for clustering problems. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 30–39, 2003.

[24] Jiecao Chen and Qin Zhang. Improved algorithms for distributed entropy monitoring. *Algorithmica*, 78(3):1041–1066, Jul 2017.

[25] Vincent Cohen-Addad, Chris Schwiegelshohn, and Christian Sohler. Diameter and k-center in sliding windows. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 19:1–19:12, 2016.

[26] Graham Cormode. Algorithms for continuous distributing monitoring: A survey. In *First International Workshop on Algorithms and Models for Distributed Event Processing 2011, Rome, Italy, September 19, 2011. Proceedings*, pages 1–10, 2011.

[27] Graham Cormode, S. Muthukrishnan, and Ke Yi. Algorithms for distributed functional monitoring. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 1076–1085, 2008.

[28] Graham Cormode, S. Muthukrishnan, Ke Yi, and Qin Zhang. Optimal sampling from distributed streams. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, pages 77–86, 2010.

[29] Graham Cormode, S. Muthukrishnan, and Wei Zhuang. Conquering the divide: Continuous clustering of distributed data streams. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 1036–1045, 2007.

[30] Graham Cormode and Ke Yi. Tracking distributed aggregates over time-based sliding windows. In *Scientific and Statistical Database Management - 24th International Conference, SSDBM 2012, Chania, Crete, Greece, June 25-27, 2012. Proceedings*, pages 416–430, 2012.

[31] Constantinos Daskalakis, Nishanth Dikkala, and Gautam Kamath. Testing Ising models. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1989–2007. Society for Industrial and Applied Mathematics, 2018.

[32] Constantinos Daskalakis, Gautam Kamath, and John Wright. Which distribution distances are sublinearly testable? In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 2747–2764, Philadelphia, PA, USA, 2018. Society for Industrial and Applied Mathematics.

[33] Constantinos Daskalakis and Qinxuan Pan. Square hellinger subadditivity for bayesian networks and its applications to identity testing. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 697–703, 2017.

[34] Domo. Data never sleeps 6.0. `https://www.domo.com/learn/data-never-sleeps-6`. Accessed: April 8, 2019.

[35] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[36] Sutanu Gayen and N. V. Vinodchandran. Algorithms for k-median clustering over distributed streams. In *Computing and Combinatorics - 22nd International Conference, COCOON 2016, Ho Chi Minh City, Vietnam, August 2-4, 2016, Proceedings*, pages 535–546, 2016.

[37] Sutanu Gayen and N. V. Vinodchandran. New algorithms for distributed sliding windows. In *16th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2018, June 18-20, 2018, Malmö, Sweden*, pages 22:1–22:15, 2018.

[38] Phillip B Gibbons and Srikanta Tirthapura. Estimating simple functions on the union of data streams. In *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, pages 281–291. ACM, 2001.

[39] Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

[40] Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 68–75. Springer, 2011.

[41] Ori Gurel-Gurevich. Expectation of square root of binomial r.v. `https://mathoverflow.net/questions/121411/expectation-of-square-root-of-binomial-r-v`, 2013. Accessed: 2018-10-25.

[42] WALTER ROBERT J HARRISON. Scalable, network-wide telemetry with programmable switches. 2019.

[43] Zengfeng Huang, Ke Yi, and Qin Zhang. Randomized algorithms for tracking distributed count, frequencies, and ranks. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '12, pages 295–306, New York, NY, USA, 2012. ACM.

[44] Piotr Indyk and David Woodruff. Tight lower bounds for the distinct elements problem. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 283–288. IEEE, 2003.

[45] T. S. Jayram, Ravi Kumar, and D. Sivakumar. The one-way communication complexity of hamming distance. *Theory of Computing*, 4(1):129–135, 2008.

[46] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, pages 41–52, 2010.

[47] Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Computational Complexity*, 8(1):21–49, 1999.

[48] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, New York, NY, USA, 1997.

[49] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

[50] Mario Lipovsky, Tomas Vinar, and Brona Brejova. Approximate Abundance Histograms and Their Use for Genome Size Estimation. In *Proceedings of the 17th Conference on Information Technologies - Applications and Theory*, ITAT 2017, pages 27–34, 2017.

[51] Andrew Mcgregor, A. Pavan, Srikanta Tirthapura, and David P. Woodruff. Space-efficient estimation of statistics over sub-sampled streams. *Algorithmica*, 74(2):787–811, February 2016.

[52] Pall Melsted and Bjarni V. Halldorsson. Kmerstream: streaming algorithms for k -mer abundance estimation. *Bioinformatics*, 30(24):3541–3547, 2014.

[53] Adam Meyerson. Online facility location. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 426–431, 2001.

[54] Hamid Mohamadi, Hamza Khan, and Inanc Birol. ntcard: a streaming algorithm for cardinality estimation in genomics data. *Bioinformatics*, 33(9):1324–1330, 2017.

[55] J.I. Munro and M.S. Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12(3):315 – 323, 1980.

[56] NIH. The cancer genome atlas. `https://cancergenome.nih.gov`. Accessed: November 16, 2018.

[57] Liam Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Transactions on Information Theory*, 54(10):4750–4755, 2008.

[58] Karl Pearson. X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, 1900.

[59] Nicolo Rivetti, Yann Busnel, and Achour Mostéfaoui. Efficiently summarizing data streams over sliding windows. In *14th IEEE International Symposium on Network Computing and Applications, NCA 2015, Cambridge, MA, USA, September 28-30, 2015*, pages 151–158, 2015.

[60] Naveen Sivadasan, Rajgopal Srinivasan, and Kshama Goyal. Kmerlight: fast and accurate k-mer abundance estimation. *CoRR*, abs/1609.05626, 2016.

[61] John Snow. *On the mode of communication of cholera*. John Churchill, 1855.

[62] Srikanta Tirthapura and David P. Woodruff. Optimal random sampling from distributed streams revisited. In *Distributed Computing - 25th International*

Symposium, DISC 2011, Rome, Italy, September 20-22, 2011. Proceedings, pages 283–297, 2011.

[63] Gregory Valiant and Paul Valiant. A CLT and tight lower bounds for estimating entropy. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:183, 2010.

[64] Gregory Valiant and Paul Valiant. The power of linear estimators. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 403–412. IEEE, 2011.

[65] Gregory Valiant and Paul Valiant. An automatic inequality prover and instance optimal identity testing. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, FOCS '14, pages 51–60, Washington, DC, USA, 2014. IEEE Computer Society.

[66] David P. Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 167–175, 2004.

[67] David P. Woodruff and Qin Zhang. Tight bounds for distributed functional monitoring. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 941–960, 2012.

[68] H. Zhao, A. Lall, M. Ogihara, and J. Xu. Global iceberg detection over distributed data streams. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 557–568, March 2010.