Space-Efficient Sampling

Sudipto Guha
Dept. of Computer and Information Sciences
University of Pennsylvania
Philadelphia, PA 19104

Andrew McGregor
Information Theory and Applications Center
University of California
San Diego, CA 92093

Abstract

We consider the problem of estimating non-parametric probability density functions from a sequence of independent samples. The central issue that we address is to what extent this can be achieved with only limited memory. Our main result is a space-efficient learning algorithm for determining the probability density function of a piecewise-linear distribution. However, the primary goal of this paper is to demonstrate the utility of various techniques from the burgeoning field of data stream processing in the context of learning algorithms.

1 Introduction

Sample complexity is a fundamental measure of complexity in many learning problems. A very general scenario is that a learning algorithm may request a sequence of samples from some (unknown) distribution and wishes to make some inference about the distribution from these samples. However, as the systems we are trying to reason about are becoming more complex, the number of samples required is increasing significantly. Typical VC arguments yields sampling complexities which are often quadratic in the error tolerance. Further, more often than not, we need to multiply this by sufficiently non-trivial factors which depend on the parameters of interest, probability bounds, etc. The storage requirement of processing a large number of samples can become expensive. However, we cannot avoid the fact that a minimum number of samples are required from the standpoint of information theory.

In this paper we begin a study to ameliorate the situation. The basic underpinning of information theoretic bounds is that a distribution is generating information at a certain "rate" and we have to collect a minimum amount of information. The question we are interested in posing is whether we need to retain all the samples that are generated, or can we maintain small summaries and still make the desired inferences? In other words, can we separate the space and sample complexities of a learning problem. This is particularly important when we have fast data source. For example, suppose we are sampling traffic at a router in an attempt to monitor network traffic patterns. We can very quickly gather a large sample; but to store the large sample creates a significant problem as the typical monitoring system is only permitted a small memory footprint and writing to disk slows the system down considerably. Even if we were permitted a reasonably sized footprint, we may be able to make more accurate inferences and predictions if we use the space more efficiently than just storing samples. We seek to design algorithms that are restricted to same rate of information as is generated by the source and sees as many (or slightly more) samples as before, but only retains a limited memory of the samples that have been seen. Unfortunately this is not always feasible and Kearns et al. [27] present an example in which a function can only be learned if the samples from which the function is deduced are explicitly stored. However, the example is, in the words of the authors, a rather "artificial" function and the goal is to be able to simulate the distribution in a specific computational model. The simulation can proceed if we store enough samples, even though we have not understood the the central process, which in their example was computing quadratic residues. We think it is worthwhile to investigate further, and possibly characterize, the space of problems for which we would not have to store all the samples.

Data Stream Model: At a high level, being able to process data and make inferences from a sequence of samples without storing all the samples fits into the framework of the data stream model [1, 24, 16]. In this model there is a stream of m data items, in this case samples, and we have memory that is sub-linear, typically poly-logarithmic, in the number of data items. The algorithm accesses these data items in a sequential order and any data item not explicitly stored is rendered inaccessible. The data stream model has gained significant currency in monitoring and query processing systems in recent years, for exam-

ple, Stanford's STREAM system [2], Berkeley's Telegraph system [10] and AT&T's Gigascope system [12]. Algorithms, both heuristic and those with provable guarantees, have been developed for a range of problems including estimating frequency moments such as the number of distinct values [17, 1, 5], quantile estimation [20], computing histograms [22, 18], wavelet decompositions [19, 21] estimating entropy [8] and the various notions of "difference" between to streams [16, 25]. There is a rich emerging literature on the various models of data streams and their applications to large data, we direct the reader to a survey article by Muthukrishnan [28] and Babcock et al. [4] for further details. However, almost all the algorithms developed to date are designed to estimate some empirical property of the data in the stream. In contrast, our goal is to make some inference about the source distribution of the data items on the assumption that these data items are independent samples from some distributions. This is a significant departure from the majority of streaming literature to date.

An important issue that arises is the potential trade-off between sample complexity and space complexity. Because we are not able to store all the samples in the alloted space, our algorithms potentially incur a loss of accuracy. Consequently we may need to investigate a slightly larger set of samples and thereby offset the loss of accuracy. The following example illustrates some of these ideas.

Example 1 (Estimating Medians)**.** *We say y is an ϵ-approximate median of a one dimensional distribution with probability density function μ if,*

$$\int_{-\infty}^{y} \mu(x)dx = 1/2 \pm \epsilon \ .$$

It can be easily shown that the sample complexity of finding an ϵ-approximate median is $\Theta(\epsilon^{-2})$. However, it can also be shown that there exists a constant $c(\delta)$ such that, with probability at least $1-\delta$, any element whose rank is $m/2 \pm \epsilon m/2$ in a set of $m = c/\epsilon^2$ samples is an ϵ-approximate median. But there exist $O(\epsilon^{-1}\log m)$-space algorithms [20] that, when presented with a stream of m values will return an element whose rank is $m/2 \pm \epsilon m/2$. Hence the space complexity of learning an ϵ-approximate median is only $O(\epsilon^{-1}\log \epsilon^{-1})$. However, the space complexity is even lower. By capitalizing on the fact that the samples will be in a random order, there exists an algorithm [23] using $O(1)$ space that returns a element whose rank is $m/2 \pm 10\sqrt{m}\ln^2 m \ln \delta^{-1}$. Hence by increasing the number of samples to $O(\epsilon^{-2}\ln^4 \epsilon^{-1})$ we may decrease the space complexity to $O(1)$.

Related Areas: The data streams model has similarities to the *online* model and competitive analysis. However, in the online setting, space issues are not usually paramount even if when they are part of the motivation for considering a problem in the online setting. (Some recent work considers the issue more closer however [14, 15].) In the data stream model the main computational restriction is limited memory. Furthermore, at every step in the online model, some decision or prediction is being made (see [**?**]). This does not have a direct analogue in the data stream model, where there is only one decision to be made after seeing all the data. Hence a learning algorithm in the data stream model is essentially doing batch learning subject to certain computational constraints. However, there is an indirect analogue: as each data item is presented to an algorithm in the data stream model, the algorithm is forced to make some irrevocable decision about what information to "remember" (explicitly or implicitly) about the new item and what information currently encoded in the current memory state can be "forgotten." This raises the prospect of rich connections existing between the areas. For example, maybe the decision and consequent cost of deciding to remember a data item in the limited space available to a data stream algorithm can be related to the request of a label in the active learning framework.

The relationship between compressibility and learnability is obviously of relevance to the problem of designing small-space learning algorithms. Also, the question considered in this paper is closely connected to the theory of sufficient statistics for parametric problems.

Lastly, in the analysis of online algorithms and data stream algorithms, the trend has been to analyze the "worst case ordering." Although such analyses are important, it is immediate in a learning scenario that we are frequently dealing with a "typical ordering" which can be abstracted as a random permutation of the data items. In a sense, this is an average case analysis where the average is not taken over specific families of distributions but over the order in which the data arrives. The reader may notice a connection to the exchangeability axioms of deFinetti [13].

Our Results: Our main goal is to show how algorithmic techniques for processing data streams can be used to achieve space-efficient learning. In this paper focus on estimating probability density functions.

For discrete distributions, over a domain $[n] = \{1, \ldots, n\}$, Batu et al. [7] provide algorithms with sublinear (in n) sample complexity. These algorithms test whether two distributions are almost identical in variational distance or at least ϵ-far apart. We show that the space complexity of their algorithm can be improved with a small blowup in sample complexity. Furthermore we consider approximating the distance between two distributions

For continuous distributions, Chang and Kannan [11] considered learning continuous distributions which can be expressed succinctly, when the samples are stored and possibly rearranged by an adversary. Our algorithms are closely related to those of Chang and Kannan but we primarily consider a model in which the samples are not stored and ar-

rive as a sequence of independent draws from the underlying distribution. We will show that the space requirement of learning algorithms for succinct representations can be reduced significantly with a very small increase in the number of samples. In the process, we improve the results proved in [11] for worst case orderings. We show that it is possible to learn a distribution whose density function is specified by k piecewise linear segments up to precision ϵ using only $\tilde{O}(k^2/\epsilon^4)$ samples and space $\tilde{O}(k)$. We will focus on the one dimensional case (Section 3) but comment on the case of higher dimensional data (Section 3.3).

We conclude with a section about the importance of the assumption that the samples in the stream are ordered randomly rather than adversarially. We discuss our ideas in the context of estimating frequency moments.

2 Notation and Preliminaries

We use the notation $x = a \pm b$ to denote $x \in [a-b, a+b]$. For a density function D on the real line and a set $J = [a, b)$, let $D(J) = \int_a^b D(y)dy$. The notation $\tilde{O}$ denotes the usual order notation with poly-logarithmic factors omitted.

We quote two results which will be used in this paper. The first result estimates the L_1 difference between two frequency vectors that are defined by the stream. The second result is on finding approximate quantiles of a data stream.

Theorem 1 (Indyk [25])**.** *Consider a stream of $2m$ elements $X = \langle x_1, \ldots, x_{2m} \rangle$ where m elements equal (p, i) for some $i \in [n]$ and m elements equal (q, i) for some $i \in [n]$. Assume that $m = \mathrm{poly}(n)$. For $i \in [n]$, let*

$$p_i = |\{(p, i) \in X\}|/m \text{ and } q_i = |\{(q, i) \in X\}|/m \ .$$

There exists an $O(\epsilon^{-2}\log(n)\log(\delta^{-1}))$-space algorithm L_1-Sketch returning T such that, with probability at least $1-\delta$,

$$(1+\epsilon)^{-1}|p-q| \leq T \leq (1+\epsilon)|p-q| \ .$$

Theorem 2 (Greenwald and Khanna [20])**.** *Consider a stream X of m real numbers. There exists a $O(\epsilon^{-1}\log m)$-space [1] algorithm Quantiles that constructs a data structure that encodes the relative rank of any x in X,*

$$rank_X(x) := m^{-1}|\{y \in X, y \leq x\}| \ ,$$

up to additive error ϵ.

3 Learning Probability Density Functions

3.1 Discrete Distributions

There exists an algorithm that tests whether the L_1 distance between two discrete distributions on n points, is greater than ϵ or less than $\epsilon/(4\sqrt{n})$ using $O(\epsilon^{-4}n^{2/3})$ samples [7]. Here we describe a method that takes more samples but only uses $O(\epsilon^{-2}\log n)$ space. Furthermore, our algorithm will actually ϵ-additively approximate the distance. It will be an integral part of an algorithm in the next section.

Theorem 3. *Consider n-point distributions p and q. Given a stream containing at least $m^* = 12\epsilon^{-2}n\log(4n/\delta)$ samples from each distribution it is possible to find an estimate T such that, with probability at least $1-\delta$,*

$$(1+\gamma)^{-1}(|p-q|-\epsilon) \leq T \leq (1+\gamma)(|p-q|+\epsilon) \ ,$$

using $O(\gamma^{-2}\log(n)\log(\delta^{-1}))$ space.

Proof. After taking m^* samples from p define f_i to be the number of samples equal to i. This defines the empirical distribution $\hat{p}_i = f_i/m^*$. First we show that $|\hat{p}-p|$ is small. Note that $E[f_i] = m^*p_i$ and by an application of Chernoff bounds,

$$\begin{aligned}\Pr\left[|\hat{p}_i - p_i| \leq \max\{\epsilon p_i/2, \epsilon/(2n)\}\right] &\leq 2e^{-\epsilon^2 m^*/(12n)} \\ &\leq \delta/(2n) \ .\end{aligned}$$

Hence with probability at least $1-\delta/2$, for all $i \in [n]$,

$$|\hat{p}_i - p_i| \leq \max\{\epsilon/(2n), \epsilon p_i/2\}$$

and so $|\hat{p}-p| \leq \epsilon/2$.

We can prove $|\hat{q}-q|$ is small in an identical way. Hence $\left||\hat{p}-\hat{q}|-|p-q|\right| \leq \epsilon$. We can approximate $|\hat{p}-\hat{q}|$ upto a multiplicative factor of $1+\gamma$ in $O(\gamma^{-2}\log(n)\log(\delta^{-1}))$ space using the result in Theorem 1. □

One interesting corollary of this result is as follows.

Corollary 1. *Let D be a member of a finite family $\mathcal{F}$ of hypothesis distributions (each one over n points). Finding an $F \in \mathcal{F}$ such that the L_1 difference between D and F is less than ϵ with probability at least $1-\delta$ can be achieved in $O(\log(n)\log(|\mathcal{F}|\delta^{-1}))$ space and $O(\epsilon^{-2}n\log(n|\mathcal{F}|\delta^{-1}))$ samples.*

This follows by setting $\gamma = 1$ and making the error probability sufficiently small such that the L_1 difference between the stream and each hypothesis distribution is accurately estimated.

3.2 Continuous Distributions

Consider a distribution on the real line with a probability generating function D that is *k-linear*, i.e. the support of D can be partitioned into k intervals such that D is linear on each interval. D can be viewed as a mixture of $O(k)$ linear distributions. We wish to find a k-linear probability density function $\hat{D}$ such that

$$\int_{-\infty}^{\infty} |D(x) - \hat{D}(x)|dx \leq \epsilon \ .$$

[1] Throughout the paper we assume that each data item can be stored in one unit of space.

Algorithm *Linear*(J, X, β, δ)
Input: interval $J = [l, u)$, stream of samples X from J, a approx-parameter β, confidence-parameter δ

1. $\eta \leftarrow \beta/8k$
2. Partition range $[l, u)$ into $[l + (i-1)(u-l)\eta, l + i(u-l)\eta)$, $i \in [1/\eta]$
3. Using the algorithm L_1-*Sketch*, let T be a 2-approximation to $\sum_{i \in [1/\eta]} |\tilde{d}_i - l_i|$ where

$$\tilde{d}_i = \frac{|X \cap [l + (i-1)(u-l)\eta, l + i(u-l)\eta)|}{|X|}$$

   and $l_i = (a(2i-1)/2 + b)\eta$ for a and b satisfying $l_1 = \min\{2\eta, \tilde{d}_1\}$ and $a/2 + b = 1$
4. If $T \leq \beta/4$ then accept otherwise reject

Figure 1: An Algorithm for testing if a distribution is linear or β-far from linear.

Algorithm *Learning-Piecewise-Linear-Distributions*(X, ϵ, δ)
Input: stream of samples X in range $[0, 1)$, approx-parameter ϵ, confidence-parameter δ

1. Define the following set of values,

$$t_1 \leftarrow k, t_2 \leftarrow 2 \text{ and } \alpha \leftarrow 1/42$$
$$d_p^l \leftarrow (1-2\alpha)^{-p} t_1 t_2^{p-1} \text{ and } d_p^u \leftarrow (1+2\alpha)^{-p} t_1 t_2^{p-1} \text{ for } p \in [\ell]$$
$$\ell \leftarrow \left\lceil \frac{\log(2kt_2\epsilon^{-1}/t_1)}{\log(t_2/(1+2\alpha))} \right\rceil \text{ and } \delta_1 \leftarrow \delta/(6\ell k)$$

2. Partition the stream into 2ℓ contiguous sub-streams; $X = X_{1,1}, X_{1,2}, X_{2,1}, X_{2,2}, \ldots, X_{\ell,1}, X_{\ell,2}$ where

   $|X_{1,1}| = m_{\text{qua}}(t_1, \alpha, \delta_1), |X_{p,1}| = 3m_{\text{qua}}(t_2, \alpha, \delta_1) d_p^l \log(\delta_1^{-1})$ and $|X_{p,2}| = 3m_{\text{lin}}(k, \epsilon d_p^u/(\ell k), \delta_1) d_p^l \log(\delta_1^{-1})$

   (recall equations Eq. 1 and 3 for the definition of $m_{\text{qua}}(\cdot,\cdot,\cdot)$ and $m_{\text{lin}}(\cdot,\cdot,\cdot)$.)
3. $\mathcal{J}_0 \leftarrow \{[0, 1)\}$
4. **for** $p \in [\ell]$:
5. **do for** $J = [a_0, a_t) \in \mathcal{J}_{p-1}$:
6. **do if** $p = 1$ **then** $t = t_1$ **else** $t = t_2$
7. Using *Quantiles*, find $(a_i)_{i \in [t-1]}$ with $rank_{X_{p,1}}(a_i) = it^{-1} \pm \alpha t^{-1}/2$.
8. Let $\text{partit}(J) = \{[a_0, a_1), \ldots, [a_{t-1}, a_t)\}$.
9. **for** $J' \in \text{partit}(J)$ **do if** *Linear*$(J', J' \cap X_{p,2}, \epsilon d_p^u/(2\ell k), \delta_1)$ rejects **then** $\mathcal{J}_p \leftarrow \{J'\} \cup \mathcal{J}_p$

Figure 2: An Algorithm for Learning D upto ϵ variational error.

Let us assume we know the range of the distribution. Note that this can be learned with error at most ϵ by taking the maximum and minimum values of the first $O(\epsilon^{-1} \log(1/\delta))$ samples. By scaling we will assume that the range of the distribution is $[0, 1)$.

The Linear Algorithm: An important sub-routine in the algorithm will be a test of whether a stream of samples from a k-linear distribution on the range $J = [l, u)$ is linear. The algorithm is presented in Fig. 1. The algorithm is very similar to an algorithm in [11] but our analysis will yield improved results over the analysis in [11]. The intuition behind the algorithm is to quantize the samples to $1/\eta$ equally spaced values between l and u. Let D_J be the distribution formed by conditioning D on the interval J. The sub-routine computes a linear distribution L that is close to D_J if D_J is linear. If D_J is far from all linear distributions then D_J will be far from L in particular. Either way, approximating the distance between L and D_J using L_1-*Sketch* will allow us to test if D_J is linear.

Theorem 4 (The *Linear* Algorithm)**.** *Let X be a stream consisting of*

$$m_{lin}(k, \beta, \delta) := ck\beta^{-3} \log(ck\beta^{-1}\delta^{-1}) \tag{1}$$

(for some sufficiently large constant c) samples drawn independently from D_J. Then, with probability $1 - \delta$, the Linear algorithm will accept if D_J is linear and will reject if D_J is not within L_1 distance β of a linear distribution. Furthermore, if D_J is linear then the algorithm determines a linear distribution at most β from D_J. The algorithm uses $O((\log(1/\beta) + \log k)\log(1/\delta))$ space.

Proof. Without loss of generality we may assume that the range of J is $[l, u) = [0, 1)$. Let L be a linear probability density function on the range $[0, 1)$ of the form $ay+b$ where a and b will be determined shortly.

Let $\eta = \beta/(8k)$. Let $d_i = \int_{(i-1)\eta}^{i\eta} D_J(y)dy$ and

$$l_i = \int_{(i-1)\eta}^{i\eta} L(y)dy = (a(2i-1)/2 + b)\eta$$

where a and b are determined by $l_1 = \min\{2\eta, \tilde{d}_1\}$ and $a/2 + b = 1$ (recall from Fig. 1 that $\tilde{d}_i$ is the probability mass observed in the interval $[(i-1)\eta, i\eta)$). First note that $l_i \leq 2\eta$ for $i \in [1/\eta]$. Then,

$$\begin{aligned}\int_{(i-1)\eta}^{i\eta} |D_J(y) - L(y)|dy &\leq l_i + d_i \\ &\leq 4\eta + |d_i - l_i|\end{aligned}$$

and $\int_{(i-1)\eta}^{i\eta} |D_J(y) - L(y)|dy \geq |d_i - l_i|$. Because D_J has at most k linear segments there are at most k values of i such that $\int_{(i-1)\eta}^{i\eta} |D_J(y) - L(y)|dy \neq |l_i - d_i|$. Therefore,

$$\int_0^1 |D_J(y) - L(y)|dy \leq \beta/2 + \sum_{i\in[1/\eta]} |d_i - l_i| \ . \quad (2)$$

At this point we consider the discrete distributions $(d_1, \ldots, d_{1/\eta})$ and $(l_1, \ldots, l_{1/\eta})$. Appealing to Theorem 3, it is possible to approximate $\sum_{i\in[1/\eta]} |d_i - l_i|$ up to an additive term of $\epsilon = \beta/10$ and multiplicative term of $\gamma = 1/10$ with probability at least $1 - \delta/10$ using $O(\log(1/\delta)\log(\beta^{-2}\eta^{-1}))$ space. Therefore, the estimate T satisfies,

$$\frac{10}{11}\sum |d_i - l_i| - \frac{\beta}{11} \leq T \leq \frac{11}{10}\sum |d_i - l_i| + \frac{11\beta}{100} \ .$$

Combining this with Eq. 2 we get that,

$$\begin{aligned}&\frac{10}{11}\int_0^1 |D_J(y) - L(y)|dy - \frac{6\beta}{11} \leq T \\ &\leq \frac{11}{10}\int_0^1 |D_J(y) - L(y)|dy + \frac{11\beta}{100} \ .\end{aligned}$$

If D_J is β-far from all linear density functions then, D_J is β-far from L. Hence $T \geq 4\beta/11$. Now suppose D_J is linear. By an application of the Chernoff-Hoeffding bounds we know that with probability at least $1-\delta/10$, $|\tilde{d}_1 - d_1| < \beta\eta/10$ and therefore $\int_0^1 |D_J(y) - L(y)|dy \leq \beta/10$. In this case $T \leq 22\beta/100$. Hence *Linear* accepts D_J if it is linear. □

The Learning Algorithm: Our algorithm uses the same template as an algorithm of Chang and Kannan [11]. The algorithm operates in ℓ phases. Each phase p generates a set of intervals $\mathcal{J}_p$. Intuitively these intervals are those on which D does not appear to be close to linear. The union of these intervals is a subset of the union of intervals in $\mathcal{J}_{p-1}$ where $\mathcal{J}_0$ contains only one interval, the range of the distribution $[0, 1)$. In the first phase of the algorithm partitions $[0, 1)$ into t_1 intervals of roughly equal probability mass using the algorithm *Quantiles*. Each of these intervals are tested in parallel to see if the distribution is linear when restricted to that interval. This step uses the algorithm *Linear*, given earlier. In subsequent phases, intervals that do not appear to be linear are further subdivided into t_2 intervals of roughly equal probability mass. Each of the sub-intervals are tested and, if they are not linear, are further sub-divided. This process continues for ℓ phases where ℓ will be a function of t_1 and t_2. At the end of the ℓ-th phase there will be at most k sub-intervals that remain and appear not to be linear. However these sub-intervals will contain so little mass that approximating then by the uniform distribution will not contribute significantly to the error.

The algorithm is given in Fig. 2. In the streaming model each iteration of Lines 5 and 9 are performed in parallel. Note that the algorithm potentially finds $2\ell k$ intervals upon which the distribution is linear. Given these, a good k-linear representation $\hat{D}$, can be found by dynamic programming.

Lemma 2. *Let X be set of*

$$m_{qua}(1/\gamma, \alpha, \delta) := c\gamma^{-2}\alpha^{-2}\log(c\delta^{-1}\gamma^{-1}) \quad (3)$$

(for some sufficiently large constant c) samples from a distribution D and, for $i \in [1/\gamma]$, let $x_i \in X$ be an element whose relative rank (with respect to X) is $i\gamma \pm \gamma\alpha/2$. Then, with probability at least $1 - \delta$, for all $i \in [1/\gamma]$, x_i has relative rank (with respect to D) $i\gamma \pm \gamma\alpha$.

Proof. Let a and b be such that $\int_{-\infty}^{a} D(x)dx = \gamma - \gamma\alpha$ and $\int_{-\infty}^{b} D(x)dx = \gamma + \gamma\alpha$. Consider the set X of n samples. Let Y_a (Y_b) be the number of elements in X that are less (greater) than a (b). Then the probability that an element whose relative rank (with respect to X) is in the range $[\gamma - \alpha\gamma/2, \gamma + \alpha\gamma/2]$ does not have relative rank (with respect to D) in the range $[\gamma - \gamma\alpha, \gamma + \gamma\alpha]$ is bounded above by,

$$\begin{aligned}&\Pr\left[Y_a > (\gamma - \frac{\alpha\gamma}{2})m\right] + \Pr\left[Y_b > (1 - \gamma - \frac{\alpha\gamma}{2})m\right] \\ &\leq \exp\left(\frac{-\alpha^2\gamma^2 m}{12(\gamma - \alpha\gamma)}\right) + \exp\left(\frac{-\alpha^2\gamma^2 m}{12(1 - \gamma - \alpha\gamma)}\right) \\ &\leq 2\exp(-m\alpha^2\gamma^2/12) \ .\end{aligned}$$

Setting $m = 12\gamma^{-2}\alpha^{-2}\log(\delta/(2\gamma))$ ensures that this probability is less than $\delta\gamma$. The lemma follows by the union bound. □

We now prove the main properties of the algorithm.

Lemma 3. *With probability $\geq 1 - \delta$, for all $p \in [\ell]$,*

1. *$|\{\mathcal{J}_p\}| \le k$ and for each $J \in \mathcal{J}_p$, $\frac{1}{d_p^l} \le D(J) \le \frac{1}{d_p^u}$.*

2. *For each $J \in \mathcal{J}_p, J' \in \text{partit}(J)$, in Line 9 the call to Linear "succeeds", i.e., accepts if $D_{J'}$ is linear and rejects if D'_J is $\epsilon d_p^u/(2\ell k)$-far from linear.*

Proof. The proof is by induction on p. Clearly $|\{\mathcal{J}_1\}| \le k$. Since $|X_{1,1} \cap J| = m_{\text{qua}}(t_1, \alpha, \delta_1)$ for all $J \in \{\mathcal{J}_0\}$, by Lemma 2,

$$\forall J \in \mathcal{J}_1,\ \frac{1-2\alpha}{t_1} \le D(J) \le \frac{1+2\alpha}{t_1}$$

with probability at least $1-\delta_1$. Appealing to the Chernoff bound and union bound, with probability at least $1-\delta_1 k$,

$$\forall J \in \mathcal{J}_0, J' \in \text{partit}(J),\ |X_{1,2} \cap J'| \ge m_{\text{lin}}\left(k, \frac{\epsilon d_1^u}{2\ell k}, \delta_1\right) .$$

Hence, by Theorem 4, with probability $1-\delta_1 k$, each call to *Linear* in the first phase succeeds.

Assume the conditions hold for phase $p-1$. Appealing to the Chernoff bound and union bound, with probability at least $1-\delta_1 k$, $|X_{p,1} \cap J| \ge m_{\text{qua}}(t_2, \alpha, \delta_1)$ for all $J \in \mathcal{J}_{p-1}$. Hence by Lemma 2, $\forall J \in \mathcal{J}_{p-1}, J' \in \text{partit}(J)$,

$$\frac{1-2\alpha}{t_2} D(J) \le D(J') \le \frac{1+2\alpha}{t_2} D(J)$$

with probability at least $1-k\delta_1$. Similarly, with probability at least $1-2\delta_1 k$,

$$\forall J \in \mathcal{J}_{p-1}, J' \in \text{partit}(J),\ |X_{1,2} \cap J'| \ge m_{\text{lin}}\left(k, \frac{\epsilon d_p^u}{2\ell k}, \delta_1\right)$$

Hence, by Theorem 4, with probability $1-2\delta_1 k$, each call to *Linear* in the p-th phase succeeds.

Hence, with probability at least $1-6k\ell\delta_1 = 1-\delta$ the conditions hold for all $p \in [\ell]$. □

Theorem 5 (The *Learning-Piecewise-Linear-Distributions* Algorithm)**.** *With probability at least $1-\delta$, on the assumption that $m = \Omega(k^2\epsilon^{-4})$ it is possible to compute an approximation to D within L^1 distances ϵ using a single pass and $\tilde{O}(k)$ space.*

Proof. Assume that the conditions in Lemma 3 hold. When *Linear* determines that an interval is close enough to linear in level p there is the potential of incurring $(\epsilon d_p^u/(2\ell k))/d_p^u = \epsilon/(2\ell k)$ error. This can happen to at most $k\ell$ intervals and hence contributes at most $\epsilon/2$ error.

The only other source of errors is the fact that there might be some intervals remaining at the last phase when $p = \ell$. However the probability mass in each interval is at most $\epsilon/(2k)$. There will be at most k of these intervals and hence the error incurred in this way is at most $\epsilon/2$.

The space complexity of the algorithm is $\tilde{O}(k)$ because at most $\max\{t_1|\{\mathcal{J}_1\}|, t_2|\{\mathcal{J}_p\}|\} \le 2k$ instances of *Linear* are run in parallel. The sample complexity is (see Eq. 1, Eq. 3, and Fig. 2 for the definitions),

$$\begin{aligned}
&\sum_{p\in[\ell]} (|X_{p,1}| + |X_{p,2}|) \\
\le\ & \tilde{O}\left(d_\ell^l + \max_{p\in[\ell]} \frac{d_p^l k}{(\epsilon d_p^u/k)^3}\right) \\
\le\ & \tilde{O}\left(k\epsilon^{-1}\left(\frac{1+2\alpha}{1-2\alpha}\right)^\ell + k^2\epsilon^{-3}\left(\frac{1+2\alpha}{1-2\alpha}\right)^\ell\right) \\
\le\ & \tilde{O}(k^2\epsilon^{-4}) \ .
\end{aligned}$$

□

Remark: The above algorithm can be adapted for the case where the stream of samples is stored and is ordered adversarially with the proviso that the algorithm may make $P = 2\ell$ passes over the samples and use $\tilde{O}(k\epsilon^{-1/\ell})$ space (ℓ can be chosen depending on the relative cost of space and passes.) This is easy to observe; reconsider the algorithm in Figure 2. Assume $P = 2\ell$ and set $t_1 = 10k\epsilon^{-1/\ell}$ and $t_2 = 10\epsilon^{-1/\ell}$. Each phase can be simulated in two passes. Thus $P = 2\ell$ passes are sufficient. This yields the following theorem.

Theorem 6. *With probability at least $1-\delta$, if $m = \tilde{\Omega}((1.25)^\ell \ell k^2 \epsilon^{-4})$, then it is possible to compute an approximation to D within L^1 distances ϵ using 2ℓ passes and $\tilde{O}(k\epsilon^{-1/\ell})$ space.*

As such it is a strict improvement over the result in [11]. The basis for this improvement is primarily a tighter analysis (particularly in Theorem 4) and the use of the quantile estimation algorithm of [20]. Note that the authors of [11] show that $O(k^2/\epsilon^2)$ samples (and space) is sufficient for one pass, but there is a significant blowup in the sample complexity in extending the algorithm to multiple passes. In our analysis this increase is much smaller, as well as the space complexity is better, which is the central point of the paper. A comparison is shown in Table 1.

	Prior Work [11]	This Work	
Length	$\tilde{O}(1.25^\ell \ell k^6 \epsilon^{-6})$	$\tilde{O}(1.25^\ell \ell k^2 \epsilon^{-4})$	$\tilde{O}(k^2\epsilon^{-4})$
Space	$\tilde{O}(k^3\epsilon^{-2/\ell})$	$\tilde{O}(k\epsilon^{-1/\ell})$	$\tilde{O}(k)$
Passes	2ℓ	2ℓ	1
Order	Adversarial	Adversarial	Random

Table 1: Comparison of Results.

3.3 High-Dimensional Generalizations

Chang and Kannan [11] show how the one dimensional result generalizes to two dimensions; and our algorithms

will give similar (improved) results. However the more interesting case is high-dimensional data. In this section we outline a streaming algorithm for learning a mixture of spherically-symmetric Gaussians in $\mathbb{R}^d$. Arora and Kannan [3, Section 3.4] prove that there exists a constant c such that if $t = c \log |S|/\delta$ and the Gaussians are t-separated ([3, Defn. 3]) then for a sample S, all the points belonging to the same Gaussian cluster C_i are within $\sqrt{2}(1 + 3t/\sqrt{d})$ times the radius with probability at least $1 - \delta$. In contrast, they show that the points belonging to separate clusters are separated by at least $\sqrt{2}(1 + 6t/\sqrt{d})$ times the radius. This immediately gives a streaming algorithm, namely take the first point, say x, and compute the smallest distance from this point. If we choose the minimum over the next $c_1 k\epsilon^{-1} \log \delta^{-1}$ points then we are guaranteed that the closest point belongs to the same cluster as the chosen point (because we may assume that the weight of the smallest cluster is at least ϵ/k). We now consider shells of radius increasing by a factor of $(1+t/\sqrt{d})$ starting from this minimum distance centered around x. The number of shells we would have to consider is constant because the minimum pairwise distance cannot be smaller by more than a factor $(1 + 3t/\sqrt{d})$ [3]. Now if we consider the next $c_2 k\epsilon^{-1} \log \delta^{-1}$ points we would have determined the shell that separates the cluster containing x and the rest of the points. At this point we need to (1) validate that if we choose all the points within the shell we get a Gaussian and (2) estimate the means of the Gaussian. If we had $O(\epsilon^{-2})$ points from the cluster this is straightforward. But then reading the next $c_3 k\epsilon^{-3} \log \delta^{-1}$ points achieves this. Note that we do not have to store these points, but only compute the mean and the variance. At this point we have identified one cluster, we repeat this process and rely on the fact that the points in the other cluster are closer to points in that cluster than this current cluster found. This sets up a simple pruning condition. Thus over the k clusters we require $O(k^2\epsilon^{-3} \log \delta^{-1})$ samples. The space complexity is $\tilde{O}(k)$. Note that by the union bound, if we set $\delta = \Theta(\delta'\epsilon/k)$ the procedure succeeds with probability at least $1 - \delta'$.

4 Importance of Random Order

One of the most important aspects of processing a stream of samples is that, as opposed to the usual assumption in the data stream model, the data elements arrive in a random order. Many properties of a stream are provably hard to estimate in small space when the order of the stream is chosen adversarially. This begs the question whether such properties are hard to estimate because a small space algorithm is necessarily forgetful or because an adversary orders the data to create confusion. While we have no definitive answer to this question, it does seem that relaxing the assumption that there exists an adversary ordering the data does significantly increase the range of problems that can be tackled in small space. Estimating Frequency Moments [1, 26], a classic problem from the literature in the streaming model, illustrates this point. The k-th frequency moment of a discrete distribution p on n points is defined as $F_k = \sum_{i\in[n]} p_i^k$. The following theorem demonstrates the enormous power achieved by streaming points in a random order rather than an adversarial order.

Theorem 7. *It is possible to $(1 + \epsilon)$-approximate F_k in a randomly ordered stream with $\tilde{O}((n/t)^{1-2/k})$ space when the stream length is $m = \Omega(ntk^2\epsilon^{-2-1/k} \log(nt\delta^{-1}))$. In particular, if $m = \Omega(n^2k^2\epsilon^{-2-1/k} \log(nt\delta^{-1}))$ then the space only depends on n polylogarithmically. If the stream was ordered adversarially, the same estimation requires $\Omega(n^{1-2/k})$ space.*

Proof. The lower bound is a generalization of the results in [9]. The algorithm is a simple "serialization" of [26] as follows. For $i \in [t]$, let $\hat{p}_j$ be the fraction of occurrences of j among the items occurring between position $1+(i-1)m'$ and im' where

$$m' = \Theta(nk^2(\ln(1 + \epsilon/10))^{-2}\epsilon^{-1/k} \log(2n\delta^{-1})) \ .$$

Use [26] to compute B_i, a $(1+\epsilon/3)$ multiplicative estimate to $A_i = \sum_{j=1+(i-1)n/t}^{in/t} \hat{p}_j^k$ with probability at least $1 - \delta/2$. Return $\sum_{i\in[t]} B_i$. Note that this sum can be computed incrementally rather than by storing each B_i for $i \in [t]$.

We will show that $\sum_{i\in[t]} A_i$ is a $(1+\epsilon/3)$ approximation to F_k with probability at least $1-\delta/2$. Subsequently it will be clear that $\sum_{i\in[t]} B_i$ is a $(1+\epsilon/3)^2 \le (1+\epsilon)$ approximation (assuming $\epsilon < 1/4$) with probability at least $1 - \delta$. By the Chernoff bound and the union bound, with probability at least $1 - \delta/2$,

$$\forall j \in [n],\ |p_j - \hat{p}_j| \le \ln\left(1 + \frac{\epsilon}{10}\right) k^{-1} \max\left(p_j, \frac{(\epsilon/10)^{1/k}}{n}\right) \ .$$

Therefore, with probability at least $1 - \delta/2$,

$$\frac{F_k - n^{1-k}\epsilon/10}{(1 + \epsilon/10)} \le \sum_{i\in[t]} A_i \le (1 + \epsilon/10)\,(F_k + n^{1-k}\epsilon/10) \ .$$

By convexity, $F_k \ge \sum_{i\in[n]} (1/n)^k = n^{1-k}$ and therefore,

$$F_k(1 + \epsilon/10)^{-2} \le \sum_{i\in[t]} A_i \le F_k(1 + \epsilon/10)^2 \ .$$

Hence $\sum_{i\in[t]} A_i$ is a $(1 + \epsilon/3)$ approximation. □

It is worth remarking that [6] showed that any streaming algorithm that merely samples (possibly adaptively) from the stream and results an $(1 + \epsilon)$-multiplicative-approximation of F_k from this sample with probability at least $1 - \delta$, must take at least $O(n^{1-1/k}\epsilon^{-1/k} \log(\delta^{-1}))$ samples.

5 Conclusions

We presented an algorithm for estimating probability density functions given a sequence of independent samples from an unknown distribution. The novelty of our algorithm was that the samples did not need to be stored and that the algorithm had very small space-complexity. This was achieved using algorithmic techniques developed for processing data streams. Unfortunately, in order to reduce the space-complexity, the algorithm needed to process more samples than that required had there been no computational restriction. A natural open question is to what extent it is possible to reduce the space-complexity without increasing the number of samples. More generally, what other types of learning algorithms can be implemented under the constraints of the data stream model?

References

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

[2] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, R. Motwani, I. Nishizawa, U. Srivastava, D. Thomas, R. Varma, and J. Widom. STREAM: The Stanford stream data manager. *IEEE Data Eng. Bull.*, 26(1):19–26, 2003.

[3] S. Arora and R. Kannan. Learning mixtures of arbitrary gaussians. In *ACM Symposium on Theory of Computing*, pages 247–257, 2001.

[4] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 1–16, 2002.

[5] Z. Bar-Yossef, T. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *Proc. 6th International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 1–10, 2002.

[6] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Sampling algorithms: lower bounds and applications. *ACM Symposium on Theory of Computing*, pages 266–275, 2001.

[7] T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing that distributions are close. In *IEEE Symposium on Foundations of Computer Science*, pages 259–269, 2000.

[8] A. Chakrabarti, G. Cormode, and A. McGregor. A near-optimal algorithm for computing the entropy of a stream. In *ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[9] A. Chakrabarti, S. Khot, and X. Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *IEEE Conference on Computational Complexity*, pages 107–117, 2003.

[10] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. A. Shah. TelegraphCQ: Continuous dataflow processing for an uncertain world. In *CIDR*, 2003.

[11] K. L. Chang and R. Kannan. The space complexity of pass-efficient algorithms for clustering. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1157–1166, 2006.

[12] C. D. Cranor, T. Johnson, O. Spatscheck, and V. Shkapenyuk. Gigascope: A stream database for network applications. In *ACM SIGMOD International Conference on Management of Data*, pages 647–651, 2003.

[13] B. de Finetti. La prévision: ses lois logiques, ses sources subjectives. *Annales de l'Institut Henri Poincaré*, 1937.

[14] O. Dekel, S. Shalev-Shwartz, and Y. Singer. The forgetron: A kernel-based perceptron on a fixed budget. In *NIPS*, 2005.

[15] O. Dekel and Y. Singer. Support vector machines on a budget. In *NIPS*, 2006.

[16] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate L^1 difference algorithm for massive data streams. *SIAM Journal on Computing*, 32(1):131–151, 2002.

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

[18] A. C. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *ACM Symposium on Theory of Computing*, pages 389–398, 2002.

[19] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *VLDB*, pages 79–88, 2001.

[20] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *ACM SIGMOD International Conference on Management of Data*, pages 58–66, 2001.

[21] S. Guha and B. Harb. Approximation algorithms for wavelet transform coding of data streams. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 698–707, 2006.

[22] S. Guha, N. Koudas, and K. Shim. Approximation and streaming algorithms for histogram construction problems. *ACM Trans. Database Syst.*, 31(1):396–438, 2006.

[23] S. Guha and A. McGregor. Approximate quantiles and the order of the stream. In *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 273–279, 2006.

[24] M. R. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. *External memory algorithms*, pages 107–118, 1999.

[25] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. *IEEE Symposium on Foundations of Computer Science*, pages 189–197, 2000.

[26] P. Indyk and D. P. Woodruff. Optimal approximations of the frequency moments of data streams. In *ACM Symposium on Theory of Computing*, pages 202–208, 2005.

[27] M. J. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire, and L. Sellie. On the learnability of discrete distributions. In *ACM Symposium on Theory of Computing*, pages 273–282, 1994.

[28] S. Muthukrishnan. Data streams: Algorithms and applications. *Now Publishers*, 2006.