

PASS-EFFICIENT ALGORITHMS FOR LEARNING MIXTURES OF UNIFORM DISTRIBUTIONS*

KEVIN L. CHANG[†] AND RAVI KANNAN[‡]

Abstract. We present multiple pass streaming algorithms for a basic statistical clustering problem for massive data sets. If our algorithm is allotted 2ℓ passes, it will produce an approximation with error at most ϵ using $\tilde{O}(k^3/\epsilon^{2/\ell})$ bits of memory, the most critical resource for streaming computation. We demonstrate that this tradeoff between passes and memory allotted is intrinsic to the problem and model of computation by proving lower bounds on the memory requirements of any ℓ pass *randomized* algorithm that are nearly matched by our upper bounds.

In this problem, we are given a set of n points drawn randomly according to a mixture of k uniform distributions and wish to approximate the density function of the mixture. The points are placed in a data stream (possibly in adversarial order), which may only be read in sequential passes by the algorithm.

The algorithm is quite general and can be adapted to solve the problems of learning a mixture of linear distributions in \mathbb{R} and a mixture of uniform distributions in \mathbb{R}^2 .

1. Introduction. The modern phenomenon of our growing ability to store extremely large amounts of information on computer systems has been accompanied by the proliferation of such massive data sets for many different problem domains. Computation on massive data sets presents new challenges for algorithm designers.

The memory space of a typical computer system may be abstractly viewed as organized into two broad categories: random access memory, which can be accessed very efficiently but has a limited capacity, and storage, which relatively has a very large capacity but is very slow to access. Storage devices usually consist of secondary storage in the form of disks, or sometimes tertiary storage, in the form of tapes and optical devices. The main difficulties of massive data set computation arise from the constraint that sufficiently large data sets cannot fit into the main memory of a computer and thus must be placed into storage. Computing thus cannot occur with the data *entirely in memory*, but rather must be *performed with the data on disk*.

It is well known that reading and writing to secondary and tertiary storage is the performance limiting factor for computations on data in storage, and that the throughput of the I/O subsystem is optimized for sequential access of the data in storage devices. Tapes are obvious examples of storage devices that exhibit this phenomenon, but disks also exhibit this behavior: random access requires that the disk head physically move and then wait for the disk to rotate to the appropriate position before reading can occur. These penalties, known respectively as seek times and rotational latency, are very expensive and can be avoided by reading the data sequentially without seeking. For massive data set calculations, frequent random access will severely lower performance.

The streaming model of computation has received much attention in the recent literature for its suitability for “streaming data” and massive data sets. In this model, the input to an algorithm may only be read in a single, sequential pass over the data; no random access of the input is allowed. The algorithm may use $o(n)$ bits of random access memory during the pass in order to keep state, buffer some input, keep statistics

*A preliminary version of this paper appeared in *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp 1157-1166, 2006 under the title “The space complexity of pass-efficient algorithms for clustering.”

[†]Yahoo! Labs, Santa Clara, CA. kevin.chang@yahoo-inc.com

[‡]Microsoft Research Labs, India.

or a sketch, etc. Furthermore, the amount of memory used by the algorithm must be small (typically $o(n)$, where n is the size of the data stream).

The *pass-efficient* model has been proposed by Drineas and Kannan [10] as a more flexible model for massive data set computation. Intuitively, the rigid restriction of the streaming model to a single pass over the data unnecessarily limits its potential utility for data that are actually stored. The pass-efficient model allows for multiple sequential passes over the input (usually a constant number of passes), and a constant amount of extra space. While processing each element of the data stream, the algorithm may only use computing time that is independent of n , but after each pass, it is allowed more computing time (typically $o(n)$). In this model of computation, we are most concerned with two resources: *the number of passes* and *additional storage space (memory)*. The model has been applied to a two pass algorithm for computing a succinct, approximate representation of a matrix [10] (see also [2]).

We will assume that our input consists of data drawn according to a probability distribution known as a mixture model. A *mixture of k distributions* is itself a probability distribution constructed as a linear combination of k simpler distributions. Thus, the data can be interpreted as being drawn from k different sources. Formally, we have probability distributions F_1, \dots, F_k over a universe Ω , each with a *mixing weight* $w_i > 0$, such that $\sum_i w_i = 1$. Both the constituent distributions and the mixing weights are unknown. A point drawn according to the *mixture* of these k distributions is defined by choosing the i th distribution with probability w_i and then picking a point according to F_i . Ideally, given data drawn according to the mixture, we could reconstruct the k clusters corresponding to the different probability distributions (i.e. learn the parameters of the k distributions). However, this problem can be ill-defined in the sense that two different sets of constituent distributions can create the exact same mixture.

If F is the density function of the mixture, the problem we consider is: can we approximately reconstruct F from samples placed in a read-only input array? More precisely, our goal is to find a function G such that the L^1 distance between F and G is at most ϵ : $\int_{\Omega} |F - G| \leq \epsilon$. We first consider the case where $\Omega = \mathbb{R}$ and each F_i is a uniform distribution over some contiguous interval $(a, b) \subset \mathbb{R}$. We call the resulting mixture a *mixture of k uniform distributions*.

Suppose that X is a set of n points randomly chosen from \mathbb{R} according to a mixture of k -uniform distributions with density function F . X can be ordered (possibly by the adversary) to produce a sequence that constitutes the read-only input array. Our problem is then: design a pass-efficient algorithm that approximates F from the input array X . Our pass-efficient learning algorithm for this problem has the property that its memory requirement drops off sharply as a function of the number of passes allotted. We note that the assumption of adversarial ordering makes any algorithm stronger than one that relies on random ordering, and that this is a natural assumption. In many practical settings, even if the data are generated by a mixture of distributions, one cannot always assume that they are ordered randomly: consider the situation where the data are sorted by some field other than the specific one of interest.

Vapnik-Červonenkis arguments show that a random sample of size $\tilde{O}(k^2/\epsilon^2)$ ¹ from X will have about the right number of points in any interval; thus intuitively clustering the sample should give us an ϵ approximation to F . Some care is needed to rigorously prove this. With multiple passes, however, one can do much better.

¹ $\tilde{O}(\cdot)$ and $\tilde{\Omega}(\cdot)$ denote asymptotic notation where poly-logarithmic factors have been omitted.

The two main results of our paper are **a)** a multiple pass algorithm whose space requirements decrease significantly when allowed extra passes and **b)** a lower bound that shows that our tradeoff between passes and memory usage is close to tight.

We now describe the general technique for our multiple pass algorithm. In a single pass, we partition the domain into a set of intervals, based on samples of the data stream. Our algorithm will then estimate F on each of these intervals separately. In a second pass, we count the number of points in X that lie in each of these intervals, and run subroutine `CONSTANT`, which determines whether or not F is (approximately) constant on each interval. If F is constant on an interval I , then the density of F can be estimated easily by the number of points of X that fall in the interval. If F is not constant on I , the algorithm recursively estimates the density in the interval using subsequent passes, in effect “zooming in” on these more troublesome intervals until we are satisfied with our approximation.

A crucial part of the algorithm is subroutine `CONSTANT`, which solves a problem of independent interest: determining in a single pass over X whether or not F is approximately constant on some interval. For any $\beta > 0$, `CONSTANT` uses only $O((\log(1/\beta) + \log k) \log(1/\delta))$ bits of memory, provided that $n = \tilde{\Omega}(k^5/\beta^5)$, and determines whether F is within β in L^1 distance of the uniform distribution. `CONSTANT` considers a slightly perturbed version of the data so that each point is drawn from an η -smoothed distribution, F_η , which is very close to F in variational distance. F_η has the useful property that it is constant on intervals that form a regular partition of the domain. We then would like to compute the number of samples of X in each of the $1/\eta$ subintervals; we look upon this as a vector v with $1/\eta$ components. $1/\eta$ will be too large for us to store the entire vector; however, using the known algorithm in [20], we find the projection of v into a random low dimensional subspace, from which we can glean a good approximation to $\|v\|_1$. Pseudorandom generators for small space computation are then used to compress the projection matrix (these generators do not require the existence of one-way functions).

In order to prove lower bounds for multiple pass algorithms, we appeal to known results for the communication complexity of the GT (Greater Than) function, which determines for two inputs $a, b \in \{0, 1\}^m$ whether $a > b$. Specifically, we show that any ℓ -pass algorithm that solves our problem will induce a $(2\ell - 1)$ -round protocol for the GT function. Lower bounds on the $(2\ell - 1)$ -round communication complexity of GT will then provide lower bounds for the memory usage of ℓ -pass streaming algorithms.

1.1. Our Results. We present the following results. Here ℓ is any positive integer chosen by the user.

1. In Section 3, we give a one pass algorithm for learning a mixture of k uniform distributions with error at most ϵ that requires $\tilde{O}(k^2/\epsilon^2)$ bits of memory.
2. In Section 4, we give a 2ℓ -pass algorithm for learning a mixture of k uniform distributions with error at most ϵ using $\tilde{O}(k^3/\epsilon^{2/\ell})$ bits of RAM, for any integer $\ell > 0$, provided that the number of samples in the input array satisfies $n = \tilde{\Omega}(1.25^\ell k^6/\epsilon^6)$. Alternatively, we can view this as an algorithm with error at most ϵ^ℓ that uses $\tilde{O}(k^3/\epsilon^2 + k\ell/\epsilon)$ bits of memory, provided that $n = \tilde{\Omega}(1.25^\ell k^6/\epsilon^{6\ell})$.
3. In Section 5, we slightly generalize our learning problem and prove a lower bound of $\Omega\left(\left(\frac{1}{2c}\right)^{1/(2\ell-1)} c^{-2\ell+1}\right)$ for the number of bits of memory needed by any ℓ -pass randomized algorithm that solves the general problem, where c is a fixed constant. We strengthen our pass-efficient algorithm in order to provide an

upper bound of $\tilde{O}(1/\epsilon^{4/\ell})$ for the number of bits of memory needed by an ℓ -pass algorithm.

4. In Sections 6 and 7 we generalize our multiple pass algorithm for uniform distributions to algorithms for *learning a mixture of k bounded linear distributions* and *learning a mixture of k bounded, two dimensional uniform distributions*, respectively. The former is a mixture of distributions over the domain \mathbb{R} such that each F_i has a density function that is linear over some contiguous interval. The latter is a mixture of distributions over the domain \mathbb{R}^2 such that each F_i is uniform over some axis-aligned rectangle.

The error of our multiple pass algorithms in Sections 4, 6, and 7 falls exponentially with the number of passes while holding the amount of memory required constant. An alternative interpretation is that memory falls sharply with the number of passes, while holding the error constant. Making more passes over the data allows our algorithms to use less space, thus trading one resource in the pass-efficient model for the other. This feature demonstrates the potential for multiple passes by providing flexible performance guarantees on space needed and passes used; the algorithm can be tailored to the specific needs and limitations of a given application and system configuration.

The lower bound proves that this tradeoff between the number of passes and the amount of memory required is nearly tight for a slightly more general problem. Thus, this exponential tradeoff is an intrinsic property of the problem.

1.2. Related Work. Chang [7] generalized the algorithm to arbitrary dimension, \mathbb{R}^d , although the space and sample complexity bounds are exponential in d . The algorithm in this paper was subsequently improved by Guha and McGregor [18], who used a similar algorithmic framework, but improved the space complexity of the multiple pass algorithm, as well as showing that if the input is randomly ordered, then the multiple pass algorithm can be simulated in a single pass using only $\tilde{O}(k)$ space.

Lower bounds for multipass algorithms. The first paper on streaming algorithms in the theory literature was by Munro and Paterson [24]. They proved a lower bound of $\Omega(n^{1/\ell})$ for the number of storage locations needed for an ℓ pass deterministic algorithm for median finding, and a nearly matching upper bound. The model of computation considered by Munro and Paterson is somewhat limited, since it is deterministic and assumes that only input elements may be stored (not some other type of encoding of information). Guha and McGregor [17] later generalized this lower bound to the standard model of computation.

Papers with space lower bounds for multiple pass algorithms include [11, 19]. These papers consider streamed graphs, and problems of the nature: find all nodes at distance exactly k from a particular vertex. The former paper considers lower bounds for deterministic algorithms. The latter paper contains a lower bound of $\Omega(nk/\ell)$ on randomized computation, as well as a nearly matching upper bound. Bar-Yossef *et al.* [5] have proved lower bounds for multiple pass algorithms approximating frequency moments, which are independent of the number of passes taken. Matching upper bounds are provided by Alon *et al.* [3].

Histogram problems. Another related problem that has been studied in the theory and database literature is the problem of histogram construction [1, 8, 12, 15, 21, 22].

Guha, Koudas, and Shim [16] considered the following problem: given a stream of n real numbers a_1, \dots, a_n in sorted order, construct a piecewise constant function F with k pieces such that $\sum_i |F(i) - a_i|^2$ is minimized. They gave a $1 + \epsilon$ factor approximation algorithm using space $O(k^2/\epsilon \log n)$. A variant of the problem is that

of histogram maintenance [13, 14], in which the a_i s are presented as a stream of updates of the form “add 7 to a_3 ”, and the algorithm must always be able to output a histogram for the current state of the a_i s. Gilbert *et al.* [14] provide a one pass $1 + \epsilon$ approximation algorithm that uses time per update and total space polynomial in $k, 1/\epsilon$, either $\log \|a\|_1$ or $\log \|a\|_2$, and $\log n$.

Note that the histogram maintenance problem is more general than learning mixtures of uniform distributions in \mathbb{R} , since it does not assume a generative model on the input. However, as we show in this paper, our algorithm is easily adapted to learning mixtures of linear distributions and mixtures of uniform distributions in \mathbb{R}^2 , which are very different from histogram maintenance.

Learning Mixtures of Gaussians. Algorithms for learning mixtures of Gaussian distributions in high dimension have been studied previously [4, 9, 28], but these are not designed for massive data sets. The objective is generally to learn the mean, covariance matrix, and mixing weight of each Gaussian in the mixture. Assumptions on the separation of the Gaussians need to be made in order to prove rigorous results about the accuracy of these algorithms; the problems are ill-defined if the Gaussians overlap too much.

2. Preliminaries.

2.1. Structure of Mixtures. We first prove that the probability density function induced by a mixture of uniform distributions can be described as a piecewise constant function.

LEMMA 2.1. *Given a mixture of k uniform distributions $(a_i, b_i), w_i$, let F be the induced probability density function on \mathbb{R} . F (except possibly at at most $2k$ points) can be represented as a collection of at most $2k - 1$ disjoint open intervals, with a constant value on each interval.*

Proof. We show how to construct the partition of the real line and the constant values that F holds on the partition.

First, consider the sequence defined by combining the a_i s and b_i s into a single sorted sequence of $2k$ points and eliminating all duplicate points. Call this new sequence c_i , for $i = 1, \dots, m \leq 2k$.

Then, consider the following $m - 1$ intervals: $I_i = (c_i, c_{i+1})$ for $i = 1, \dots, m - 1$. These intervals cover the entire support of F (the set $\{x : F(x) \neq 0\}$), except possibly at the c_i s (of which there are at most $2k$.)

Note that F is constant on each interval (c_i, c_{i+1}) because, by definition of the c_i s, no endpoint of the original intervals falls in the open interval (c_i, c_{i+1}) . Let d_i be the value of F on I_i .

If χ_{I_i} is the characteristic function of interval I_i (i.e. $\chi_{I_i}(x) = 1$ if $x \in I_i$ and 0 if $x \notin I_i$) then

$$F(x) = \sum_i d_i \chi_{I_i}(x), \tag{2.1}$$

except possibly at the points c_i . \square

Since the above characterization of F holds except on a set of measure zero, it will suffice for our algorithm to learn F with this representation. We will be somewhat lax in our language, and say that a set of intervals partitions some larger interval I , even if there are a finite number of points of I that are not included in any interval of the partition.

DEFINITION 2.2. *We define the step function representation of a mixture of k uniform distributions to be the set of at most $2k - 1$ pairs, $(x_i, x_{i+1}), h_i$, for $i =$*

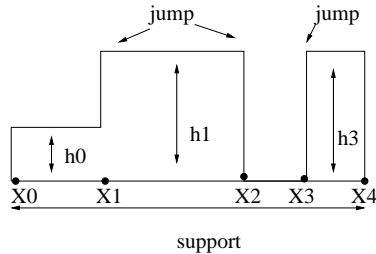


FIG. 2.1. A mixture of 3 uniform distributions, labeled as a step function with 4 steps. The x -axis is the domain, while the y -axis represents the density of the distribution. $h_2 = 0$ is not labeled; it is the density of the third step.

$0, \dots, m-1 \leq 2k$ such that the density of the mixture is a constant h_i on the interval (x_i, x_{i+1}) . Implicit in this definition is the fact that the m steps form a partition of the interval (x_0, x_m) .

DEFINITION 2.3. The support of a mixture of uniform distributions given by step function representation $(x_i, x_{i+1}), h_i$ consisting of m steps is the interval (x_0, x_m) . If F is the density function of the mixture, we denote its support by $\text{Support}(F)$.

DEFINITION 2.4. A jump of a step function is a point where the density changes (i.e. an endpoint shared by two different steps of the mixture). We will work exclusively with the step function representation of the mixture.

2.2. The VC Bound. Our algorithm makes use of a powerful statistical tool called the *Vapnik-Červonenkis (VC) Dimension* and its associated VC-Bound. Computational learning theory makes use of these theorems; the most notable application is an algorithm for PAC learning by Blumer *et al* [6].

The nature of this bound is as follows. We are given a family of sets \mathcal{C} and a sample S of m points from some probability distribution. Given a set $U \in \mathcal{C}$, we can estimate empirically the probability that a point falls in U using our m sample points. A natural question to ask is how close is this empirical estimate to the true probability of a point falling in U . Chernoff bounds give sharp relative bounds on this difference for a *fixed* $U \in \mathcal{C}$. VC bounds, on the other hand, give absolute bounds on this difference for *all* $U \in \mathcal{C}$ *simultaneously*.

VC Bounds are given in terms of the VC dimension of \mathcal{C} , which is a combinatorial measure of the complexity of the family of sets \mathcal{C} . The VC dimension of the set of all open intervals in \mathbb{R} is two.

Given a (measurable) probability density function F , let $\mu(U) = \int_U F$ be the probability that a point drawn from the distribution falls in the set U . Given a sequence X of m points randomly chosen according to the distribution F , define $|X \cap U|$ to be the set of points in X that lie in set U . Note that $|X \cap U|/m$ is the obvious empirical estimate of $\int_U F$.

FACT 2.5 (Vapnik and Červonenkis, Talagrand bound). (*Theorem 1.2 from [26]*) Let \mathcal{C} be a family of measurable sets with VC dimension d , and let $\epsilon > 0, \delta > 0$. Then there exists a constant c_0 such that if X is a set of m samples drawn according to μ , and

$$m \geq c_0 \frac{1}{\epsilon^2} \left(d \log \left(\frac{1}{\epsilon} \right) + \log \left(\frac{1}{\delta} \right) \right), \quad (2.2)$$

then

$$\Pr \left[\sup_{U \in \mathcal{C}} \left| \frac{|X \cap U|}{m} - \mu(U) \right| \geq \epsilon \right] \leq \delta.$$

The power of the VC bound lies in the fact that the error is bounded *simultaneously* for all sets contained in \mathcal{C} .

Vapnik and Červonenkis [27] were the first to use VC dimension in order to prove more general probabilistic results, from which Theorem 2.5 is derived. Talagrand [26] later improved their results to optimality, but this does not change the asymptotic sample complexity for our purposes.

3. A Single Pass Algorithm. In this section, we present a single pass algorithm that with probability $1 - \delta$ will learn the density function of a mixture of k uniform distributions within L^1 distance ϵ using at most $\tilde{O}(k^2/\epsilon^2)$ bits of memory.

ONEPASS is a divide and conquer algorithm reminiscent of merge sort. Given a uniform random sample of $m = \tilde{O}(k^2/\epsilon^2)$ points drawn from the data stream, ONEPASS first initializes the computation, then sorts the m points. It calls subroutine CLUSTER, which divides the sample into two halves and recursively approximates the density function F in each half as a step distribution. CLUSTER then applies a *merging* procedure that decides whether or not the rightmost interval of one half can be merged with the leftmost interval of the other half.

Before describing the algorithm, we make the following definition.

DEFINITION 3.1. *Suppose we have a set of points S and two intervals $J \subset J'$. We say that the empirical density of J is consistent with J' on S , or simply J is consistent with J' on S , if*

$$\left| \frac{|J \cap S|}{|S| \text{length}(J)} - \frac{|J' \cap S|}{|S| \text{length}(J')} \right| \leq \frac{\epsilon}{97k \log m} \cdot \frac{1}{\text{length}(J)}.$$

The algorithm is described in Figure 3.1.

We define the *base intervals* to be the intervals containing exactly two sample points created in the base case of CLUSTER. We define an *intermediate interval* to be an interval that was output by one of the calls to CLUSTER in the recursion, but not necessarily one of the final output intervals. We say that CLUSTER *merges* two intermediate intervals I_{k_1} and J_1 if in Step 5 it chooses to output a set of intervals that contains the interval $K = I_{k_1} \cup J_1$.

If viewed from a bottom-up perspective, the basic idea of ONEPASS is that the base case consists of a partition of $\text{Support}(F)$ into base intervals. Adjacent intermediate intervals are then repeatedly merged together if the empirical densities of the two intervals are similar enough. After all mergers, the resulting intervals define the steps of the output function, G . Care must be taken in choosing which adjacent intervals to consider for merging; each time an interval is merged, a small error is induced. If we consider merging indiscriminately, then we can potentially induce a large aggregate error if an interval participates in too many mergers. The recursion defined by CLUSTER gives a structure for choosing the mergers such that no interval is involved in more than $\log m$ mergers. This fact is explicitly used for the proof of Lemma 3.3.

A simple calculation will show that we have chosen m in Equation 3.1 such that, with the appropriate constants, the VC bound guarantees that with probability $1 - \delta$,

$$\left| \frac{|S \cap I|}{m} - \int_I F \right| \leq \frac{\epsilon}{194k \log m}$$

for all intervals $I \subseteq \text{Support}(F)$. It follows that with probability $1 - \delta$,

$$\left| \frac{\int_I F}{\text{length}(I)} - \frac{|S \cap I|}{m \text{length}(I)} \right| \leq \frac{\epsilon}{194k \log m} \cdot \frac{1}{\text{length}(I)}, \quad (3.2)$$

for all intervals $I \subseteq \text{Support}(F)$. The results of this section are conditioned on this event occurring.

LEMMA 3.2. *The output of ONEPASS contains at most $4k$ steps.*

Proof. Let N be the number of base intervals. Note that CLUSTER is called exactly $N - 1$ times. If we can show that all but $4k - 1$ of those calls to CLUSTER result in a merger, then we have shown the total number of intervals in the output of ONEPASS is $4k$, since each merger reduces the number of intervals by one.

Thus, we seek to prove that all but $4k - 2$ of the calls to CLUSTER result in a merger, using the following claim.

CLAIM: Suppose two adjacent intermediate intervals I_1 and I_2 are contained in the same step of F , which has density h . Then both I_1 and I_2 are consistent with $I_1 \cup I_2$ on S .

Proof of Claim. By Inequality (3.2), we know that both I_1 and I_2 satisfy:

$$\left| h - \frac{|I_i \cap S|}{m \text{length}(I_i)} \right| \leq \frac{\epsilon}{194k \log m \cdot \text{length}(I_i)}$$

and that $I_1 \cup I_2$ satisfies:

$$\left| h - \frac{|(I_1 \cup I_2) \cap S|}{m \text{length}(I_1 \cup I_2)} \right| \leq \frac{\epsilon}{194k \log m \cdot \text{length}(I_1 \cup I_2)}.$$

Combining these two inequalities yields

$$\begin{aligned} \left| \frac{|(I_i) \cap S|}{m \text{length}(I_i)} - \frac{|(I_1 \cup I_2) \cap S|}{m \text{length}(I_1 \cup I_2)} \right| &\leq \frac{\epsilon}{194k \log m} \left(\frac{1}{\text{length}(I_i)} + \frac{1}{\text{length}(I_1 \cup I_2)} \right) \\ &\leq \frac{\epsilon}{97k \log m \cdot \text{length}(I_i)}. \end{aligned}$$

This proves the claim.

Thus, two adjacent intervals will fail to merge in Step 5 of CLUSTER only if one or both contains a jump in F . Since there are at most $2k - 1$ jumps in F , and each can induce at most two failed mergers (one that fails to merge an intermediate interval containing the jump with the intermediate interval to its right, and the other with the intermediate interval to its left). The algorithm may fail to merge two intermediate intervals at most $4k - 2$ times. This proves the lemma. \square

LEMMA 3.3. *Suppose J is an intermediate interval output by some call to CLUSTER. Then*

$$\left| \frac{\int_J G}{\text{length}(J)} - \frac{\int_J F}{\text{length}(J)} \right| \leq \frac{\epsilon}{96k \text{length}(J)}.$$

Proof. Since all intervals output by some (possibly intermediate) call to CLUSTER are contained in one of the steps output by ONEPASS, we know that $J \subseteq \tilde{J}_i$, where \tilde{J}_i is some step of the output distribution G with density $|S \cap \tilde{J}_i| / (m \text{length}(\tilde{J}_i))$.

If J were produced by some intermediate call to CLUSTER (i.e. J is not part of the output, but rather J is strictly contained in \tilde{J}_i), then J must have been merged into some larger intermediate interval J_1 in Step 5 of some call to CLUSTER. If J_1 is a strict subset of \tilde{J}_i , then in turn J_1 must have been merged into some larger interval J_2 , and so forth until we finally reach \tilde{J}_i . If we set $J_0 = J$ and $J_t = \tilde{J}_i$, we have the following chain of inclusions: $J = J_0 \subset J_1 \subset \dots \subset J_{t-1} \subset J_t = \tilde{J}_i$. Since the depth of the recursion of ONEPASS is $\log m$, the number of mergers J is involved in is at most $\log m$. Hence, $t \leq \log m$.

By the design of the algorithm, J_j is merged into J_{j+1} only if J_j is consistent with J_{j+1} on S ; thus, we must have for all $0 \leq j \leq t-1$

$$\left| \frac{|S \cap J_j|}{\text{mlength}(J_j)} - \frac{|S \cap J_{j+1}|}{\text{mlength}(J_{j+1})} \right| \leq \frac{\epsilon}{97k \log \text{mlength}(J_j)}.$$

Thus, we can estimate the error:

$$\begin{aligned} \left| \frac{|S \cap J|}{\text{mlength}(J)} - \frac{|S \cap \tilde{J}|}{\text{mlength}(\tilde{J})} \right| &\leq \sum_{j=0}^{t-1} \left| \frac{|S \cap J_j|}{\text{mlength}(J_j)} - \frac{|S \cap J_{j+1}|}{\text{mlength}(J_{j+1})} \right| \\ &\leq \sum_{j=0}^{t-1} \frac{\epsilon}{97k \log m \cdot \text{length}(J_j)} \leq \frac{\epsilon}{97k \text{length}(J)}, \end{aligned}$$

where the last inequality follows from the fact that $t \leq \log m$ and $\text{length}(J) \leq \text{length}(J_j)$, for all j .

The Lemma then follows from applying Inequality (3.2) and the fact that $|S \cap \tilde{J}_i|/\text{mlength}(\tilde{J}_i)$ is precisely the value of G on interval J :

$$\begin{aligned} \left| \frac{\int_J G}{\text{length}(J)} - \frac{\int_J F}{\text{length}(J)} \right| &\leq \left| \frac{|S \cap J|}{\text{mlength}(J)} - \frac{|S \cap \tilde{J}_i|}{\text{mlength}(\tilde{J}_i)} \right| + \left| \frac{|S \cap J|}{\text{mlength}(J)} - \frac{\int_J F}{\text{length}(J)} \right| \\ &\leq \frac{\epsilon}{97k \text{length}(J)} + \frac{\epsilon}{194k \log m \cdot \text{length}(J)} \leq \frac{\epsilon}{96k \text{length}(J)}, \end{aligned}$$

for m sufficiently large. \square

THEOREM 3.4. *One pass algorithm ONEPASS will learn a mixture of k uniform distributions to within L^1 distance ϵ with probability at least $1 - \delta$, using at most $\tilde{O}\left(\frac{k^2}{\epsilon^2}\right)$ bits of memory.*

Proof. Consider a step \tilde{J}_i of the approximate distribution G , and let t_i be the number of jumps of F that occur in \tilde{J}_i . We prove the following bound on the error induced by G on \tilde{J}_i :

$$\int_{\tilde{J}_i} |F - G| \leq (t_i + 1) \frac{\epsilon}{6k}. \quad (3.3)$$

We know that there are at most $4k$ of the \tilde{J}_i s from Lemma 3.2, and $\sum t_i \leq 2k - 1$, since there are at most $2k - 1$ jumps in F . Inequality (3.3) implies that

$$\int |F - G| = \sum_i \int_{\tilde{J}_i} |F - G| \leq \sum_i (t_i + 1) \frac{\epsilon}{6k} \leq \epsilon,$$

from which the theorem follows.

We now prove Inequality (3.3). Consider one of the intervals that comprises a step of the output distribution, \tilde{J}_i . Recall that the intervals created by CLUSTER in the base case of the recursion formed a partition of $\text{Support}(F)$, and that for such an interval C , $|C \cap S| < 3$. The step \tilde{J}_i can be partitioned into a set of these base intervals. Enumerate the t_i jumps of F that lie in \tilde{J}_i as $x_1, \dots, x_{t_i} \in \tilde{J}_i$. Let C_j be the base interval such that $x_j \in C_j$.

Now consider the set $U = \tilde{J}_i \setminus (\cup_j C_j)$. Since \tilde{J}_i is an interval and the C_j s are intervals, of which there are at most t_i , U is a union of at most $t_i + 1$ intervals. Let these intervals be D_1, \dots, D_l , where $l \leq t_i + 1$. Note that since the C_j s are all the intervals that contain jumps, it follows that the D_j s do not contain jumps and hence each is contained in a step of F .

We first bound the error induced by the C_j s. Fix a C_t . From the VC-bound,

$$\left| \frac{|S \cap C_t|}{m} - \int_{C_t} F \right| \leq \frac{\epsilon}{194k \log m}.$$

Since $|S \cap C_t|/m < 3/m$, it follows that $\int_{C_t} F \leq \epsilon/(97k \log m)$. Lemma 3.3 thus implies that $\int_{C_t} G < 2\epsilon/(96k)$. It then follows that

$$\int_{C_t} |F - G| \leq \left| \int_{C_t} F + \int_{C_t} G \right| \leq 3 \frac{\epsilon}{96k \log m} \leq \frac{\epsilon}{32k}.$$

We now bound the error induced by the D_j s. Fix such a D_t . We may assume that $|S \cap D_t|/m \geq \epsilon/24k$; otherwise, we can show that the induced error on D_j is small using the same argument as for the C_t s.

CLAIM: If $|S \cap D_t| \geq 16$, there exists an intermediate interval I such that $I \subseteq D_t$ and $|S \cap I| \geq |S \cap D_t|/4$.

Proof of claim. Let $q = |S \cap D_t|$. Recall that $S = \{s_1, \dots, s_m\}$ was the set of sample points, in sorted order. Then $S \cap D_t = \{s_p, s_{p+1}, \dots, s_{p+q}\}$, for some integer p . Note that $2^{\lceil \log(q/2) \rceil} > q/4$. Thus, there must exist an integer j such that $p \leq j \cdot 2^{\lceil \log(q/2) \rceil} \leq (j+1) \cdot 2^{\lceil \log(q/2) \rceil} \leq p+q$. The set $\{s_r | j \cdot 2^{\lceil \log(q/2) \rceil} \leq r \leq (j+1) \cdot 2^{\lceil \log(q/2) \rceil}\} \subset D_t$ will be the input set T to one of the calls to CLUSTER, and thus will comprise an entire intermediate interval I . This proves the claim.

Since F and G are both constant on D_t and hence are both constant on $I \subseteq D_t$, $|\int_I F - \int_I G| = \int_I |F - G|$. By Lemma 3.3 it follows that $|\int_I F - \int_I G| \leq \epsilon/96k$. Since the value of F and G are constant on all of D_t , we know that

$$\int_{D_t} |F - G| = \frac{\text{length}(D_t)}{\text{length}(I)} \int_I |F - G| \leq \frac{\text{length}(D_t)}{\text{length}(I)} \frac{\epsilon}{96k}.$$

By the VC bound, we can show that $\int_I F \geq 1/8 \int_{D_t} F$, which implies that $\text{length}(I) \geq 1/8 \text{length}(D_t)$. Combining this fact with the above inequality yields

$$\int_{D_t} |F - G| \leq \frac{\epsilon}{12k}.$$

This implies that the total error induced by \tilde{J}_i is

$$\int_{\tilde{J}_i} |F - G| = \sum_j \int_{C_j} |F - G| + \sum_j \int_{D_j} |F - G| \leq t_i \frac{\epsilon}{12k} + (t_i + 1) \frac{\epsilon}{12k} \leq (t_i + 1) \frac{\epsilon}{6k},$$

which proves Inequality (3.3). \square

4. Multiple Passes. In this section, we show that additional passes can significantly reduce the amount of memory needed to achieve a fixed level of accuracy. Given a mixture of k uniform distributions with density function F and an integer $\ell > 0$, algorithm SMALLRAM can approximate F to within L^1 distance ϵ with 2ℓ passes, using at most $\tilde{O}(k^3/\epsilon^{2/\ell})$ memory, provided that $n = \tilde{\Omega}((1.25^\ell k^6)/\epsilon^{6\ell})$. We first give an algorithm with accuracy ϵ^ℓ that uses space at most $\tilde{O}(k^3/\epsilon^2 + \ell k/\epsilon)$.

SMALLRAM requires a subroutine called CONSTANT. CONSTANT is a single pass algorithm that will determine whether or not a data stream contains points drawn from a distribution with a constant density function. In Section 4.2, we describe the algorithm and prove the following theorem about its performance:

THEOREM 4.1. *Let H be a mixture of at most k uniform distributions. Given a number $\beta > 0$ and data stream X consisting of $|X| = \tilde{\Omega}(\frac{k^5}{\beta^5} \log(1/\delta))$ samples drawn according to H , with probability $1 - \delta$ single pass algorithm CONSTANT will **accept** if H is uniform, and will **reject** if H is not within L^1 distance β of uniform. CONSTANT uses at most $O((\log(1/\beta) + \log k) \log(1/\delta))$ bits of memory.*

4.1. The Algorithm. Subroutine ESTIMATE draws a random sample S of size $m = \tilde{O}(k^2/\epsilon^2)$ from the input stream and stores it in memory. It sorts S and partitions it into $O(k/\epsilon)$ intervals. ESTIMATE then calls CONSTANT on each of the intervals; CONSTANT uses the entire data stream X to determine if the distribution is constant on each interval. If F is close to constant on an interval I , we can output the interval as a step of the final output G with a constant density estimate derived from the quantity $|X \cap I|$, which we can determine in one pass over the data stream. Since $|X|$ is large, this estimate will be very accurate. However, if F is far from constant on I , then estimating its density by a constant is too coarse; thus we repeat the process by recursively calling ESTIMATE on I . This recursive call can be thought of as “zooming in on the jumps.” See Figure 4.2.

Given a sequence of points X and an interval J , we define $X|_J$ to be the subsequence of X that consists of points that lie in J . If X is presented as a data stream, a pass over X can simulate a pass over the data stream $X|_J$.

Our main algorithm is given in Figure 4.1.

REMARK The order in which this recursive algorithm computes each call to ESTIMATE has a natural structure defined by the level of the call (defined by its parameter p): All level p copies of ESTIMATE are run in parallel, then all level $p + 1$ copies are run in parallel, and so forth. All copies of ESTIMATE running in parallel may read the data stream simultaneously, thus limiting the total number of passes per level to two. The total number of passes is therefore 2ℓ .

We say that an interval J_i^p created in Step 3 of a level p call is a level p interval.

The following lemma states that the weight of F is roughly the same in all level p intervals.

LEMMA 4.2. *Let J_i^p be an interval created in Step 3 of a level p call to ESTIMATE. With probability at least $1 - \delta\epsilon/(25k^2\ell)$, J_i^p contains at most $\epsilon^p/2k$ proportion of the total weight of F , and at least $(8/10)^p\epsilon^p/2k$ proportion of the total weight. That is,*

$$\left(\frac{8}{10}\right)^p \cdot \frac{\epsilon^p}{2k} \leq \int_{J_i^p} F \leq \frac{\epsilon^p}{2k}.$$

Proof. We use a simple inductive argument to prove that an interval J_i^p created at level p contains at least $(8/10)^p\epsilon^p/2k$ proportion of the total weight of F . Assume that all level $p - 1$ intervals have at least $(8/10)^{p-1}\epsilon^{p-1}/2k$ proportion of the total

weight of F , with probability at least $1 - (p-1)\delta\epsilon/(50k^2\ell^2)$. Suppose interval J_i^p is marked in a level p call to ESTIMATE. Note that J_i^p was contained in some level $p-1$ interval $J_{i'}^{p-1}$ that was marked in level $p-1$.

CLAIM: If $J_{i'}^{p-1}$ contains at least $(8/10)^{p-1}\epsilon^{p-1}/2k$ proportion of the weight of F , then J_i^p contains at least $(8/10)\epsilon$ fraction of the weight of $J_{i'}^{p-1}$ with probability at least $1 - \delta\epsilon/(50k^2\ell^2)$.

Proof of Claim. Consider the level p call to ESTIMATE with $J_{i'}^{p-1}$ as input that created J_i^p . Since we chose $m = \Omega(k^2/\epsilon^2 \log(\ell k/\delta\epsilon))$, applying the VC bound will show that with probability at least $1 - \delta\epsilon/(50k^2\ell^2)$,

$$\left| \frac{|S \cap J_i^p|}{m} - \int_{J_{i'}^{p-1}} \frac{F}{\int_{J_{i'}^{p-1}} F} \right| \leq \frac{1}{10}\epsilon.$$

Since J_i^p was chosen so that $|X \cap J_i^p| = (9/10)\epsilon|X \cap J_{i'}^{p-1}|$, J_i^p contains at least $(8/10)\epsilon$ fraction of the total weight of $J_{i'}^{p-1}$. This proves the claim.

By the inductive hypothesis, $J_{i'}^{p-1}$ contains at least $(8/10)^{p-1}\epsilon^{p-1}/2k$ fraction of the weight of F with probability $1 - (p-1)\delta\epsilon/(50k^2\ell^2)$; thus it follows from the union bound that J_i^p contains at least $(8/10)^p\epsilon^p/2k$ fraction of the weight of F with probability $1 - p\delta\epsilon/(50k^2\ell^2)$.

The base case of the induction follows from the same argument as the proof of the claim, except we note that at the first level, in Step 2 we had set $q = \frac{9}{10} \cdot \frac{\epsilon}{2k} \cdot m$. It follows from the VC bound that

$$\left| \int_{J_i^1} F - \frac{|S \cap J_i^1|}{m} \right| \leq \frac{1}{10} \cdot \frac{\epsilon}{2k},$$

from which it follows that J_i^1 will contain at least $\frac{8}{10} \cdot \frac{\epsilon}{2k}$ of the weight of F .

We can prove that J_i^p contains at most $\epsilon^p/2k$ of the weight of F with probability at least $1 - p\delta\epsilon/(50k^2\ell^2)$ with an analogous argument. \square

COROLLARY 4.3. *Let J_i^p be some interval created in step 3 of ESTIMATE, and suppose $\epsilon < 8/10$. With probability at least $1 - \delta\epsilon/(20k^2\ell)$, the data stream X satisfies*

$$|X \cap J_i^p| = \Omega\left(\frac{k^5}{\epsilon^{5\ell}} \cdot \ell \cdot \log\left(\frac{k\ell}{\delta\epsilon}\right)\right). \quad (4.2)$$

Proof. By Lemma 4.2, we know that any given interval contains at least $(8/10)^\ell(\epsilon^\ell/2k)$ proportion of the weight of F with probability at least $1 - \delta\epsilon/(25k^2\ell)$. Since there are a sufficient number of points in X , ($|X|$ is given in Equation (4.1)), the lemma follows from an application of the Chernoff bound. \square

LEMMA 4.4. *With probability at least $1 - \delta/2$, the aggregate number of intervals marked in all level p calls to ESTIMATE is at most $2k - 1$, for all p .*

Proof. We prove this Lemma by induction on the level. Assume that at most $2k - 1$ intervals are marked in the $(p-1)$ th iteration with probability at least $1 - (p-1)\delta/2\ell$.

Since level p calls to ESTIMATE are only executed on intervals marked in the $p-1$ th iteration, by the inductive hypothesis, there are at most $2k - 1$ level p calls to ESTIMATE, with probability $1 - (p-1)\delta/2\ell$. Thus, the total number of level p intervals is at most $(10/9)2k/\epsilon \cdot (2k - 1)$. With probability $1 - \delta/4\ell$, Corollary 4.3 holds for all level p intervals.

Because Corollary 4.3 ensures that we have an adequate number of sample points fall in all level p intervals, we may apply Theorem 4.1. Thus, with probability $1 - \delta/2\ell$, CONSTANT will work as guaranteed on all level p intervals. CONSTANT will reject interval J_i^p only if F is not uniform on J_i^p (i.e. J_i^p contains a jump). Since there are at most $2k - 1$ jumps in F , at most $2k - 1$ intervals can be rejected, and thus marked in the p th iteration, with probability at least $1 - p\delta/2\ell$. \square

We summarize all the events from the above lemmas and corollary:

COROLLARY 4.5. *With probability at least $1 - \delta/2$, the following are true:*

1. *Any level p interval contains at most $\epsilon^p/2k$ proportion of the weight.*
2. *Equation (4.2) holds for all intervals created in Step 3 of ESTIMATE.*
3. *There are at most $2k - 1$ calls to ESTIMATE at any level.*
4. *No call to CONSTANT fails. (Recall that CONSTANT is a Monte Carlo algorithm.)*

We now prove the main theorem of this section.

THEOREM 4.6. *With probability at least $1 - \delta$, SMALLRAM will compute an approximation to F within L^1 distance ϵ^ℓ of F , using at most 2ℓ passes and $\tilde{O}(k^3/\epsilon^2 + \ell k/\epsilon)$ bits of memory.*

Proof. We condition this proof on all four items of Corollary 4.5 being true, which has probability at least $1 - \delta/2$.

We first bound the amount of memory used. At any level of the computation, we are running at most $2k - 1$ parallel copies of ESTIMATE. Since each copy uses at most $O(m)$ bits of memory, the total amount of memory used by ESTIMATE is at most $2k \cdot O(m)$.

Each call to CONSTANT requires at most $O((\ell \log(1/\epsilon) + \log k) \log(\ell k/\epsilon\delta))$ bits of memory. Since we run at most $O(k/\epsilon)$ copies of CONSTANT in parallel, the total memory used by our algorithm is at most:

$$2k \cdot O(m) + O\left(\frac{\ell k}{\epsilon} \left(\log \frac{1}{\epsilon} + \log k\right) \log\left(\frac{1}{\epsilon\delta}\right)\right) = O\left(\left(\frac{k^3}{\epsilon^2} + \frac{\ell k}{\epsilon} \log \frac{1}{\epsilon}\right) \log\left(\frac{\ell k}{\epsilon\delta}\right)\right).$$

We next prove that the algorithm outputs a good approximation to F . Note that after all ℓ levels of the algorithm, our approximation G consists of a partition of $\text{Support}(F)$ into disjoint intervals, with a constant density on each interval.

We classify the intervals that define the steps of G into two types.

1. Unmarked intervals that had their densities estimated in Step 6 of some call to ESTIMATE. A type (1) interval J_i^p can further be classified into one of two cases:
 - (a) F is constant on J_i^p .
 - (b) F has a jump in J_i^p , but this was not detected by CONSTANT.
2. Intervals marked in level ℓ that had their density estimated by 0 in Step 7 of a level ℓ call to ESTIMATE.

A bound for the error from type (1), case (a) intervals

Suppose J_i^p is a type (1), case (a) interval. It follows from the Chernoff bound and Corollary 4.5, item 2, that with probability at least $1 - \delta\epsilon/(32k^2\ell)$,

$$\left| \frac{|X \cap J_i^p|}{n} - \int_{J_i^p} F \right| \leq \frac{\epsilon^\ell}{4k} \int_{J_i^p} F.$$

This inequality then implies that

$$\begin{aligned} \frac{\epsilon^\ell}{4k} \int_{J_i^p} F &\geq \left| \frac{|X \cap J_i^p|}{n} - \int_{J_i^p} F \right| = \left| \int_{J_i^p} G - \int_{J_i^p} F \right| \\ &= \int_{J_i^p} |F - G|, \end{aligned}$$

where the last equality follows from the fact that F and G are both constant on J_i^p .

Let Γ be the set of all type (1) case (a) intervals. Since there are at most $((10/9)4\ell k^2/\epsilon)$ intervals in Γ , the probability that the above inequality holds for all of them is at least $1 - \delta/4$. The total error induced by all such intervals is at most:

$$\sum_{J_i^p \in \Gamma} \int_{J_i^p} |F - G| \leq \frac{\epsilon^\ell}{4k} \sum_{J_i^p \in \Gamma} \int_{J_i^p} F \leq \frac{\epsilon^\ell}{4k}.$$

A bound for the error from type (1), case (b) intervals

Suppose that J_i^p belongs to case (b). It follows from the Chernoff bound and item 2 of Corollary 4.5 that with probability $1 - \delta\epsilon/(32k^2\ell)$,

$$\begin{aligned} \left| \frac{|X \cap J_i^p|}{n \text{length}(J_i^p)} - \frac{\int_{J_i^p} F}{\text{length}(J_i^p)} \right| &\leq \frac{\epsilon^\ell}{8k^2 \text{length}(J_i^p)} \int_{J_i^p} F \\ &\leq \frac{\epsilon^\ell}{8k^2 \text{length}(J_i^p)}. \end{aligned}$$

Define $\bar{F}|_{J_i^p} = \frac{1}{\text{length}(J_i^p)} \int_{J_i^p} F$ to be the *average density* of F on J_i^p . Since J_i^p was not marked, we know that subroutine CONSTANT accepted when run on J_i^p . This means that F is at most $\epsilon^\ell/2k$ in L^1 distance from constant on J_i^p :

$$\int_{J_i^p} |F - \bar{F}|_{J_i^p}| \leq \frac{\epsilon^\ell}{2k}.$$

Combining the two inequalities above gives us

$$\frac{\epsilon^\ell}{2k} + \frac{\epsilon^\ell}{8k^2} \geq \int_{J_i^p} \left| F - \frac{|X \cap J_i^p|}{n \text{length}(J_i^p)} \right| = \int_{J_i^p} |F - G|.$$

If t_1 is the number of case (b) intervals, the total error induced by all case (b) intervals is at most $(\epsilon^\ell/2k + \epsilon^\ell/8k^2) \cdot t_1$, with probability at least $1 - \delta/4$.

A bound for the error from type (2) intervals

Each interval of type (2) was created at a level ℓ call to CONSTANT. Thus by Corollary 4.5, item 1, each contains at most $\epsilon^\ell/2k$ proportion of the weight of F . Estimating F on these intervals by 0 will induce an error of at most $\epsilon^\ell/2k$ for each interval. The total error induced by type (2) intervals is at most $\epsilon^\ell/2k \cdot t_2$, where t_2 is the number of type 2 intervals.

A bound for the total error

Since type (1) case (b) and type (2) intervals only occur at jumps in steps of F , $t_1 + t_2 \leq 2k - 1$. Thus, with probability at least $1 - \delta$, the total error is at most

$$\begin{aligned} \frac{\epsilon^\ell}{4k} + (t_1 + t_2) \left(\frac{\epsilon^\ell}{2k} + \frac{\epsilon^\ell}{8k^2} \right) &\leq \frac{\epsilon^\ell}{4k} + (2k - 1) \left(\frac{\epsilon^\ell}{2k} + \frac{\epsilon^\ell}{8k^2} \right) \\ &\leq \frac{\epsilon^\ell}{4k} + \frac{(2k - 1)\epsilon^\ell}{2k} + \frac{\epsilon^\ell}{4k} = \epsilon^\ell. \end{aligned}$$

□

If we analyze the algorithm with $\epsilon^{1/\ell}$ in place of ϵ , we get an ϵ approximation requiring $\tilde{O}(k^3/\epsilon^{2/\ell})$ bits of memory. Note that the $k\ell/\epsilon$ term disappears, because we assume that $\ell = O(\log(1/\epsilon))$ (memory usage does not decrease beyond a constant factor if $\ell \geq \log(1/\epsilon)$).

COROLLARY 4.7. *With probability at least $1 - \delta$, there exists an algorithm that will compute an approximation to F within L^1 distance ϵ of F , using 2ℓ passes and $\tilde{O}(k^3/\epsilon^{2/\ell})$ bits of memory.*

4.2. Testing intervals for uniformity: Proof of Theorem 4.1. We now describe the single pass subroutine `CONSTANT` called by `SMALLRAM`. Given a data stream of n samples from a mixture of k uniform distributions with density function H , `CONSTANT` will accept H if it is constant, and reject if it is not within β in L^1 distance of constant.

By suitably scaling and translating the input, we may assume without loss of generality that the support of H is exactly the interval $[0, 1]$. Thus, our subroutine will determine whether $\int_0^1 |H - 1| \leq \beta$ or not.

4.2.1. Preliminaries. Approximating the ℓ_1 length of a vector given as a stream of dynamic updates. Consider a data stream X consisting of pairs of the form $\langle i, a \rangle$, $i \in [n]$ and $a \in \{-M, \dots, M\}$. This stream represents vector $v \in \mathbb{R}^n$, where each component of the vector is given by $v_i = \sum_{\langle i, a \rangle \in X} a$. In words, each time we see a pair $\langle i, a \rangle$ from the data stream, we add the integer a to the i th component of v . We define the quantity $L_1(X) = \left(\sum_{i \in [n]} \left| \sum_{\langle i, a \rangle \in X} a \right| \right)$. The ℓ_p norm of a vector $v \in \mathbb{R}^n$ is defined as $\|v\|_p = \left(\sum_{i=1}^n |v_i|^p \right)^{1/p}$. Note that $L_1(X) = \|v\|_1$ is the ℓ_1 length of v . Indyk designed a one pass algorithm for this problem. We state a weaker version of his result that is suitable for our purposes:

FACT 4.8. [20] *There is an algorithm that will find an estimate of $L_1(S)$ that is within an additive error of $\pm \frac{L_1(S)}{2}$ of $L_1(S)$ with probability $1 - \delta$ and that uses $O(\log M \log(1/\delta))$ bits of memory, $O(\log M \log(n/\delta))$ random bits, and $O(\log(n/\delta))$ arithmetic operations per pair $\langle i, a \rangle$.*

Indyk's algorithm works roughly as follows. In order to avoid storing all components of the vector v at a cost of $\Omega(n)$ bits of memory, the algorithm projects the vector into a randomly chosen subspace of low dimension d . Suppose the random projection is defined by a random $d \times n$ matrix A . The algorithm maintains the d dimensional vector \tilde{v} , which it initializes to 0. Upon reading the element $\langle i, a \rangle$, it will update \tilde{v} by setting $\tilde{v} \leftarrow \tilde{v} + A(a\vec{e}_i)$, where \vec{e}_i is the i th basis vector of \mathbb{R}^n with 1 in the i th component and 0 in all other components. After all elements of the data stream have been processed, we have $\tilde{v} = Av$.

The random projection will approximately preserve the norm of the vector, but vastly reduce its dimension to d , and hence reduce the number of components that need to be stored to d . At first glance, it may seem necessary to store the projection matrix of size at least dn ; however, pseudorandom generators for space bounded computations allow us to instead store a small, truly random seed and generate the dn (pseudorandom) entries of the projection matrix. Thus, instead of storing the matrix, we run the pseudorandom generator on the same random seed to generate each matrix element as we need it, and then delete it from memory immediately after using it. These pseudorandom generators for space bounded computation were designed by Nisan [25] and do not use hard core bits and thus do not require cryptographic assumptions such as the existence of one way functions.

The η -smoothed distribution of H . For ease of exposition of the proof of Theorem 4.1, we introduce the η -smoothed distribution of H_η , which is a smoothed version of H .

DEFINITION 4.9. *Given a mixture of k uniform distributions that defines a probability density H and a number $0 < \eta \leq 1$, define the η -smoothed distribution to be the following distribution, with density H_η . Let $i \leq 1/\eta - 1$ be the integer such that $x \in (i\eta, (i+1)\eta]$. Then*

$$H_\eta(x) = \frac{1}{\eta} \int_{i\eta}^{(i+1)\eta} H(y) dy.$$

In words, H_η is a step distribution whose $1/\eta$ steps are intervals that form a regular partition of $(0, 1]$. The constant value of H_η on the step $(i\eta, (i+1)\eta]$ is simply the average value of H on the step. Let h_i be the density of H_η on the intervals $(i\eta, (i+1)\eta)$. Let $\alpha_i = |h_i - 1|$ be the difference between the density of H_η and 1 on the i th interval. Our interest in H_η lies in the following useful properties.

LEMMA 4.10. *If H is uniform, then H_η is also uniform.*

LEMMA 4.11. *Suppose that H is constant with value w_i on the interval I , and that*

$$|w_i \text{length}(I) - \text{length}(I)| > \frac{\beta}{2k}.$$

If $\eta < \beta/5k$ and H_η is the density function of the η -smoothed distribution of H , then for some step I' of H_η , we have $|H_\eta(x) - 1| > \beta/2k$ for $x \in I'$.

Proof. There are two cases.

Case 1: $w_i \text{length}(I) > \text{length}(I) + \beta/2k$. Case 1 can be broken down into two subcases.

Subcase A: $\text{length}(I) \geq 2\eta$. Immediately, we know that $w_i > 1 + \beta/2k$. Since $\text{length}(I) \geq 2\eta$, there must exist an integer j such that $(j\eta, (j+1)\eta) \subset I$. On the domain $(j\eta, (j+1)\eta)$, H_η will have the same density as H . Thus, the density of H_η on $(j\eta, (j+1)\eta)$ will be at least $1 + \beta/2k$.

Subcase B: $\text{length}(I) < 2\eta$. Then we know that at least half of I is contained in a single interval $(j\eta, (j+1)\eta)$ of the η partition. Also, we know that $w_i \text{length}(I) \geq \beta/2k$. By definition, for $x \in (j\eta, (j+1)\eta)$,

$$H_\eta(x) = \frac{1}{\eta} \int_{j\eta}^{(j+1)\eta} H dy \geq \frac{1}{\eta} \int_{I \cap (j\eta, (j+1)\eta)} H dy \geq \frac{1}{2\eta} \int_I H = \frac{1}{2\eta} w_i \text{length}(I) \geq \frac{5}{4} > 1 + \frac{\beta}{2k},$$

for β/k small enough.

Case 2: $w_i \text{length}(I) \leq \text{length}(I) - \beta/2k$. In this case, we must have $\text{length}(I) \geq 2\eta$. The proof follows the treatment in **Subcase A** above. \square

COROLLARY 4.12. *If $\eta \leq \beta/5k$ and H is not within L^1 distance β of uniform, there exists a step of H_η such that $\alpha_i \geq \beta/2k$.*

The two lemmas and the corollary establish the connection between H and H_η . We will prove that CONSTANT will accept if H_η is uniform, and will reject if there exists a step with density h_i of H_η such that $|h_i - 1| > \beta/2k$. Lemma 4.10 and Corollary 4.12 establish that this is sufficient for the guarantee of Theorem 4.1.

4.2.2. The algorithm and its proof of correctness. First, we define two random variables based on the data stream that can be used to estimate α_i .

1. Let $N_i = |X \cap (i\eta, (i+1)\eta)|$ be the number of points of X that fall into the interval $(i\eta, (i+1)\eta)$. Note that $\mathbf{E}[N_i/n\eta] = h_i$.
2. Let $\hat{\alpha}_i = \left| \frac{1}{\eta} \frac{N_i}{n} - 1 \right|$ be an estimate of α_i based on the data stream. Note that $\mathbf{E}[\hat{\alpha}_i] = \alpha_i$.

The random variable $\hat{\alpha}_i$ gives an estimate of α_i , our quantity of interest. However, in a single pass we do not have enough space to compute and store $\hat{\alpha}_i$ for all $1/\eta$ values of i . The proof of the next lemma uses Indyk's algorithm, which we described above, to estimate $\|\hat{\alpha}\|_1$. From the value of $\|\hat{\alpha}\|_1$, we can glean our desired information: a small value of $\|\hat{\alpha}\|_1$ implies that all $\hat{\alpha}_i$ are small, whereas a large value implies that at least one $\hat{\alpha}_i$ is large.

LEMMA 4.13. *In a single pass over X , Indyk's algorithm can compute an estimate ζ of $\|\hat{\alpha}\|_1$ such that $(1/2)\|\hat{\alpha}\|_1 \leq \zeta \leq (3/2)\|\hat{\alpha}\|_1$ with probability at least $1 - \delta$, using space at most $O(\log(\eta n) \log(1/\delta))$.*

Proof. First consider the construction of a data stream S_X derived from X :

1. For each element $x \in X$, let $j_x \in [1/\eta]$ be the integer such that $x \in (j_x\eta, (j_x+1)\eta)$. We append $\langle j_x, 1 \rangle$ to S_X .
2. We append to S_X the $1/\eta$ elements $\langle i, -\eta n \rangle$, for all $i \in [1/\eta]$.

Indyk's algorithm on input S_X can be simulated in a single pass over X . We have:

$$L_1(S_X) = \sum_{i=0}^{1/\eta} |N_i - \eta n| = \eta n \sum_i \left| \frac{1}{\eta} \frac{N_i}{n} - 1 \right| = \eta n \|\hat{\alpha}\|_1.$$

Thus, in a single pass we can derive an estimate ζ of $\|\hat{\alpha}\|_1$ such that $(1/2)\|\hat{\alpha}\|_1 \leq \zeta \leq (3/2)\|\hat{\alpha}\|_1$ with Indyk's algorithm using space at most $O(\log(\eta n) \log(1/\delta))$. \square

See Figure 4.3 for the formal description of the algorithm.

Proof of Theorem 4.1. Since $n = \tilde{\Omega}(k^2/(\beta^2\eta^3) \log(1/\delta))$, an application of the Chernoff bound implies that with probability at least $1 - \delta$, for all $1/\eta = O(k/\beta)$ choices of i simultaneously: if $h_i \leq 2$ then $|N_i - h_i\eta n| \leq \frac{\beta\eta}{40k}\eta n$, which implies that

$$|\hat{\alpha}_i - \alpha_i| \leq \frac{\beta\eta}{20k}.$$

If $h_i \geq 2$ then the Chernoff bound yields $|N_i - h_i\eta n| \leq \frac{\beta\eta}{20k}h_i\eta n$, which implies that

$$\hat{\alpha}_i \geq \frac{1}{2} \geq \frac{\beta}{k}.$$

We prove the correctness of CONSTANT by considering two cases:

Case 1: H_η is uniform. Then $h_i = 1$ for all i , and $\alpha_i = 0$ for all i . Thus, $\hat{\alpha}_i \leq \frac{1}{20k}\beta\eta$ for all i . By Lemma 4.13, this implies that $\zeta \leq \frac{3}{2}\|\hat{\alpha}\|_1 \leq \beta/5k$; therefore CONSTANT will accept H_η . Appealing to Lemma 4.10, it follows that CONSTANT will accept if H is uniform.

Case 2: There exists at least one h_i such that $|h_i - 1| \geq \beta/2k$. Then $\|\hat{\alpha}\|_1 \geq \beta/2k - \frac{1}{20k}\beta\eta > \beta/(2.5k)$. By Lemma 4.13, this implies that $\zeta \geq \frac{1}{2}\|\hat{\alpha}\|_1 > \beta/5k$; therefore CONSTANT will reject H_η . Appealing to Corollary 4.12, it follows that CONSTANT will reject if H is not within L^1 distance β of uniform. \square

5. Nearly matching bounds on memory usage of randomized algorithms. In this section, we slightly generalize our learning problem and consider lower bounds on the amount of memory needed by any ℓ pass randomized algorithm for this new problem. We then present nearly matching upper bounds for this general problem by adapting our algorithm from the previous section.

These lower bounds will confirm that a compelling performance feature of our algorithm, its exponential tradeoff between the amount of memory required and the number of passes taken, is close to the best possible. We first define the following generalized learning problem:

Generalized Learning Problem: *Let F be the density function of a mixture of at most $1/\epsilon$ uniform distributions over the domain $[0, 1] \subset \mathbb{R}$. Let $t \in [0, 1]$ be the largest number such that F is a step distribution with at most k steps on $[0, t]$. Given a data stream X consisting of a sufficient number of independent samples drawn according to F , with probability at least $1 - \delta$, find a function G and a number $t' > t$ such that $\int_0^{t'} |F - G| < \epsilon$.*

Intuitively, the problem is to learn the density function of the first k steps of a step distribution that contains at most $1/\epsilon$ steps.

We use known lower bounds for the communication complexity of r -round protocols to prove that any ℓ pass algorithm needs at least $\Omega\left(\left(\frac{1}{2\epsilon}\right)^{1/(2\ell-1)} c^{-2\ell+1}\right)$ bits of memory to solve this problem for the $k = 3$ case. We then generalize our algorithms for learning mixtures of k -uniform distributions to algorithms for solving the generalized learning problem. These algorithms will provide an upper bound of $\tilde{O}(k^3/\epsilon^{4/\ell})$ on the number of bits of memory needed by an ℓ pass algorithms, if $\ell > 0$ is an even integer.

5.1. A Lower Bound.

5.1.1. Brief review of communication complexity. Consider the following communication problem: Alice and Bob are given vectors $a, b \in \{0, 1\}^n$, respectively, and wish to compute $f(a, b)$ for some Boolean function f . In order to perform the computation, Alice and Bob may pass messages to each other, may use randomization, and are allowed unlimited computing power. In general, their goal is to exchange the smallest number of bits possible to compute $f(a, b)$.

An r -round protocol is a protocol with the restriction that Alice and Bob may exchange at most r messages, where Alice must send the first message and only one of the players need output the answer. The r -round probabilistic communication complexity of f , denoted by $R^r(f)$, is the number of bits in the largest message, minimized over all r -round protocols that output f correctly with probability at least $2/3$.

Define the function $\text{GT}_n : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ by $\text{GT}_n(a, b) = 1$ if and only if $a \geq b$. A lower bound of $R^r(\text{GT}_n) = \Omega(n^{1/r} c^{-r})$, for some fixed constant c , and a nearly matching upper bound of $R^r(\text{GT}_n) = O(n^{1/r} \log n)$ are known [23]. We will use $R^r(\text{GT}_n)$ to prove lower bounds for the memory requirements of multiple pass algorithms.

5.1.2. Main Theorem. **THEOREM 5.1.** *Any ℓ -pass randomized algorithm that solves the Generalized Learning Problem for $k = 3$ with probability at least $2/3$ and error ϵ requires at least $\Omega\left(\left(\frac{1}{2\epsilon}\right)^{1/(2\ell-1)} c^{-2\ell+1}\right)$ bits of memory, for some fixed constant c .*

Proof. Fix $k = 3, \epsilon > 0$ and $m = 1/(2\epsilon)$. We will reduce the communication problem GT_m to the streaming Generalized Learning Problem with input parameters k and ϵ . Suppose there exists an ℓ pass algorithm A that solves the Generalized Learning Problem with probability $2/3$ and that uses at most $M(A)$ bits of memory. We will prove that A induces a $2\ell - 1$ -round protocol for GT_m that uses at most $M(A)$ bits of communication per message. This will prove that $M(A) \geq R^{2\ell-1}(\text{GT}_m) = \Omega(1/(2\epsilon)^{2\ell-1})$.

Suppose that Alice and Bob are given vectors $a, b \in \{0, 1\}^m$. Alice will construct a probability density function μ_a^A . μ_a^A will be a piecewise constant function on each of the $2m$ intervals $(i/m, (i + 1/2)/m]$ and $((i + 1/2)/m, (i + 1)/m]$ as follows: set $\mu_a^A(x) = 2a_{m-i}$ for $x \in (i/m, (i + 1/2)/m]$ and $\mu_a^A(x) = 2(1 - a_{m-i})$ for $x \in ((i + 1/2)/m, (i + 1)/m]$.

Bob will construct a similar function μ_b^B , but “reversed”: $\mu_b^B(x) = 2(1 - b_{m-i})$ for $x \in (i/m, (i + 1/2)/m]$ and $\mu_b^B(x) = 2b_{m-i}$ for $x \in ((i + 1/2)/m, (i + 1)/m]$.

Alice (Bob) generates an arbitrarily large set of independent samples drawn according to μ_a (μ_b) of arbitrary precision and places them in a data stream X_a (X_b). We require that $|X_a| = |X_b|$. The data stream formed by appending X_b to X_a , denoted by $X_a \circ X_b$, can be viewed as having been drawn from the distribution defined by density function $\mu = \frac{\mu_a^A}{2} + \frac{\mu_b^B}{2}$. Note that μ is a step function with at most $1/\epsilon$ steps.

CLAIM: If Bob is given the solution to the Generalized Learning Problem on data stream $X = X_a \circ X_b$ with parameter $k = 3$, he can compute $\text{GT}_m(a, b)$.

Proof of Claim. Bob has in his possession the vector b and the output of the algorithm: a number t' such that on $(0, t')$, μ is comprised of at least 3 steps, and a function μ_{approx} such that

$$\int_0^{t'} |\mu - \mu_{\text{approx}}| < \epsilon. \quad (5.1)$$

In order to compute $\text{GT}_m(a, b)$, all Bob needs to do is learn the bits a_i for $i \geq j^*$, where j^* is the highest order bit for which the vectors a and b differ. Note that μ is a step function consisting of 3 steps on the interval $(0, (j^* + 1)/m)$: it has a constant value of 1 on $(0, j^*/m)$ since $a_i = b_i$ for bits $i > j^*$. If $a_{j^*} = 1$ (and correspondingly $b_{j^*} = 0$) then it has a constant value of 2 on $(j^*/m, (j^* + 1/2)/m)$ and a value of 0 on $((j^* + 1/2)/m, (j^* + 1)/m)$. If $a_{j^*} = 0$ (and correspondingly $b_{j^*} = 1$), the situation is reversed. Note that $t' \geq (j^* + 1)/m$.

Bob can compute $\text{GT}_m(a, b)$ in the following manner. He finds (by enumeration) a vector $\tilde{a} \in \{0, 1\}^m$ that induces the probability density $\tilde{\mu} = \frac{\mu_{\tilde{a}}^A}{2} + \frac{\mu_b^B}{2}$, such that $\int_0^t |\tilde{\mu} - \mu_{\text{approx}}| \leq \epsilon$. Bob then outputs 1 iff $\tilde{a} \geq b$.

The correctness of this scheme follows from the observation that $\tilde{a}_i = a_i$ for all bits $i \geq j^*$. Suppose this were not the case, that \tilde{a} differed from a by a bit a_i for $i \geq j^*$. Then $\int_0^t |\tilde{\mu} - \mu| \geq 2\epsilon$, but since Bob is guaranteed inequality (5.1), it follows from the triangle inequality that $\int_0^t |\tilde{\mu} - \mu_{\text{approx}}| \geq \epsilon$, which is a contradiction. Thus, $\tilde{a}_{j^*} = a_{j^*}$. This proves the claim.

We now describe the $2\ell - 1$ -round protocol for computing $\text{GT}_m(a, b)$. Alice simulates the first pass of algorithm A on X_a to find the contents of memory at this intermediate stage of the pass. She sends these $M(A)$ bits of memory to Bob. Bob

continues the simulation of the first pass of A on X_b and sends his $M(A)$ bits of memory to Alice. They simulate each pass of A in this fashion until all ℓ passes have been completed, sending a total of $2\ell - 1$ messages of size at most $M(A)$.

After the completion of the communication, Bob will have the output of A run on data stream $X_a \circ X_b$, from which he can determine $\text{GT}_m(a, b)$. \square

5.2. An Upper Bound. In this section we generalize our algorithm for learning a mixture of k -uniform distributions to one that will solve our generalized learning problem. Our algorithm will provide the following upper bound on the amount of memory needed by a 2ℓ pass algorithm:

THEOREM 5.2. *There exists a 2ℓ pass algorithm that will solve the Generalized Learning Problem with probability at least $1 - \delta$ and error ϵ^ℓ , using at most $\tilde{O}(k^3/\epsilon^2 + \ell k/\epsilon)$ bits of memory.* By transforming the parameters ϵ and ℓ , we arrive at the following corollary, which is an upper bound comparable to the lower bound of Theorem 5.1.

COROLLARY 5.3. *For even ℓ , there exists an ℓ -pass algorithm that can solve the Generalized Learning problem with probability $2/3$ and error ϵ using at most $\tilde{O}(k^3/\epsilon^{4/\ell})$ bits of memory.*

We show how to modify algorithm SMALLRAM from the previous section to solve our generalized problem. Recall the recursive structure of SMALLRAM: The 2ℓ passes were organized into ℓ successive calls to ESTIMATE. The p th call to ESTIMATE was assigned level p .

The input to a call to ESTIMATE is an interval J . In the first pass of this call, we drew a sample from $X|_J$ and partitioned J into level p subintervals with weight $\Theta(\epsilon)$ (or $\Theta(\epsilon/k)$ if $p = 1$) of the weight that fell in J . In the second pass, we tested each subinterval for uniformity by calling subroutine CONSTANT; if F was close to constant on a subinterval then we estimated F 's density on J with high accuracy using the entire data stream. If F was not close to constant on J , we marked the subinterval and recursively estimated F on the subinterval in subsequent passes. In the final pair of passes, marked intervals were estimated by a constant 0.

In order to adapt SMALLRAM to solve the Generalized Learning Problem, we make the following three modifications:

1. In Step 5, call CONSTANT with error parameter $\epsilon^{\ell+1}/2$ and with input parameter $1/\epsilon$ in lieu of k (i.e. CONSTANT will assume that its input is a mixture of at most $1/\epsilon$ uniform distributions, rather than k). By Theorem 4.1, CONSTANT will need $O(\ell \log(1/\epsilon) \log(\ell k/\epsilon\delta))$ bits of memory.
2. After all level p calls to ESTIMATE terminate, we may potentially have a large number of level $p + 1$ recursive calls to ESTIMATE. We do not execute them all, but rather only the calls on the k intervals that are farthest to the left (i.e. closest to 0). We “drop” the recursive calls on all other intervals, and thus do not have estimates of F on the entire domain.
3. In addition to outputting the densities of intervals, output $t' \in (0, 1)$, which is the largest number such that we have estimates of $F(x)$ for all $x \in (0, t')$.

Proof sketch of Theorem 5.2. We first bound the amount of memory needed by the modified algorithm. Due to our second modification, we know that there are at most k recursive calls at any level. Each of these parallel calls takes a sample of size $\tilde{O}(k^2/\epsilon^2)$ in its first pass, and in its second pass makes $O(k/\epsilon)$ calls to CONSTANT, each of which uses at most $O(\ell \log(1/\epsilon) \log(\ell k/\epsilon\delta))$ bits of memory. Thus, the total memory usage of the algorithm is $\tilde{O}(k^3/\epsilon^2 + \ell k/\epsilon)$.

We next bound the error of our algorithm for the Generalized Learning Problem. The crux of the analysis of the error of SMALLRAM relies on four properties. We present their analogs for the generalized algorithm. The first three properties are proved in a similar manner to those of SMALLRAM by applying VC and Chernoff bounds; we will not repeat these arguments here. The fourth follows immediately from the modifications given above.

1. The weight of F that lies in each level ℓ interval is at most $\epsilon^\ell/2k$.
2. Estimating F on unmarked intervals on which F is constant incurs a negligible error, due to the sufficient number of samples in the data stream.
3. Estimating F by a constant on unmarked intervals on which F contains a jump incurs an error of at most $\epsilon^{\ell+1}/2$, due to our guarantees for CONSTANT.
4. At most k recursive calls were made at each level p .

Let G be the approximation to the first k steps of F produced by our modified algorithm. The error of the approximation is defined by: $\int_0^{t'} |F - G|$. The algorithm outputs G as a constant on a set of intervals that partitions $(0, t')$. There are three types of intervals: (a) unmarked intervals in which F does not contain a jump, (b) unmarked intervals in which F does contain a jump (there are at most $1/\epsilon$ since F is a mixture of at most $1/\epsilon$ distributions), (c) marked level ℓ intervals estimated as 0 (there are at most k).

The total error is then:

$$\int_0^{t'} |F - G| = \text{error from (a)} + \text{error from (b)} + \text{error from (c)} \leq 1/\epsilon \cdot \epsilon^{\ell+1}/2 + k \cdot \epsilon^\ell/2k \leq \epsilon^\ell.$$

□

6. Learning mixtures of bounded linear distributions. A linear distribution has a density function that is a linear function defined over a continuous interval in \mathbb{R} . In this section we consider the problem of learning a mixture of k linear distributions in \mathbb{R} . Let F be the density function of the mixture. We will make the assumption that the mixture lies in the interval $(0, 1)$, and will need to make the additional assumption that the algorithm is provided with an upper bound on F , which is given by some scalar $w > 0$, such that $F(x) \leq w$, for all x in the domain of F . Note that if the mixture lies in an interval larger than $(0, 1)$, one would suitably scale the interval down and correspondingly scale w up in order to apply the algorithm. As with mixtures of uniform distributions, we can describe F as a piecewise linear density function with at most $O(k)$ different pieces: $F(x) = a_i x + b_i$ for $x \in (x_i, x_{i+1})$. Note that a mixture of uniform distributions is a special case of a mixture of linear distributions.

6.1. Testing intervals for linearity. CONSTANT can be strengthened to test whether or not a data stream of samples is drawn from a distribution that is within L^1 distance β of a single linear distribution. For the rest of this section, we will write “linear” in lieu of “single linear distribution.” We will prove the following analog to Theorem 4.1.

THEOREM 6.1. *Let H be the density function of a probability distribution that is piecewise linear on $(0, 1)$, with at most k pieces. Let $w > 0$ be an upper bound on H (i.e. $H(x) \leq w$ for all $x \in \mathbb{R}$). Given a data stream X consisting of $\tilde{\Omega}(k^5 w^5 / \beta^5)$ samples drawn according to H , there exists an algorithm LINEAR that with probability $1 - \delta$ will **accept** if H is linear and will **reject** if H is not within L^1 distance β of linear. The algorithm uses $O((\log(1/\beta) + \log k + \log w) \log(1/\delta))$ bits of memory.*

As before, for ease of exposition of the proof of correctness of LINEAR, we will work with the η -smoothed distribution of H . Note that the η -smoothed distribution of H will be constant on each of the $1/\eta$ intervals, $(j\eta, (j+1)\eta)$ for $j = 0, \dots, 1/\eta - 1$, that define the partition for the η -smoothed distribution. Let h_i be the density of the distribution on the interval $(i\eta, (i+1)\eta)$.

Denote $\text{interp}_H(x)$ as the linear function that satisfies $\text{interp}_H(\eta/2) = H_\eta(\eta/2)$ and $\text{interp}_H(1 - \eta/2) = H_\eta(1 - \eta/2)$. More explicitly, it is given by the function $\text{interp}_H(x) = ax + b$, where

$$a = \left(\frac{h_{\frac{1}{\eta}-1} - h_0}{1 - \eta} \right) \quad (6.1)$$

and

$$b = h_0 - a\eta/2.$$

It is immediate that if H is linear, then $H(x) = \text{interp}_H(x)$.

Define $\alpha_j = \left| h_j - \frac{1}{\eta} \int_{j\eta}^{(j+1)\eta} \text{interp}_H(x) dx \right|$ to be the difference between h_i and the average value of $\text{interp}_H(x)$ on the interval $(j\eta, (j+1)\eta)$.

LEMMA 6.2. *If H is linear, then $\alpha_j = 0$ for all j .*

LEMMA 6.3. *Let $\eta \leq \beta/(6kw)$. Suppose that there exists some interval $I = (x_i, x_{i+1})$ such that H is linear on I (i.e. $H(x) = a_i x + b_i$ for $x \in I$), and $\int_I |H(x) - \text{interp}_H(x)| dx \geq \beta/k$. Then there exists a j such that $\alpha_j \geq \beta/2k$.*

Proof. Define $J = \{j \in \mathbb{Z} | (j\eta, (j+1)\eta) \cap I \neq \emptyset\}$ to be the set of indices that correspond to intervals in the partition that intersect I . Let γ_j be the L_1 distance between $H(x)$ and $\text{interp}_H(x)$ on the interval $(j\eta, (j+1)\eta)$, given by $\gamma_j = \int_{j\eta}^{(j+1)\eta} |H(x) - \text{interp}_H(x)| dx$.

CLAIM: There are at most 3 values of $j \in J$ such that $\frac{1}{\eta} \gamma_j \neq \alpha_j$.

Proof of Claim. There are only three ways this can occur:

1. When $(j+1)\eta > x_{i+1}$. This only occurs for one value of j , $\max_{j \in J} j$.
2. When $j\eta \leq x_i$. Similarly, this only occurs for one value of j , $\min_{j \in J} j$.
3. The definitions of α_j and H_η imply that

$$\alpha_j = \left| h_j - \frac{1}{\eta} \int_{j\eta}^{(j+1)\eta} \text{interp}_H(x) dx \right| = \frac{1}{\eta} \left| \int_{j\eta}^{(j+1)\eta} (H(x) - \text{interp}_H(x)) dx \right|.$$

It is apparent that $\alpha_j \neq \frac{1}{\eta} \gamma_j$ only when $H(x) - \text{interp}_H(x)$ changes sign in $(j\eta, (j+1)\eta)$. For j such that $(j\eta, (j+1)\eta) \subset I$, this occurs for at most one j , since $H(x)$ and $\text{interp}_H(x)$ are both linear in I .

This proves the claim.

The assumption that $a_i x + b_i \leq w$ for all $x \in I$ implies that

$$\gamma_j = \int_{j\eta}^{(j+1)\eta} |a_i x + b_i - \text{interp}_H(x)| \leq w\eta, \quad (6.2)$$

for all $j \in J$.

From Inequality (6.2) with $\eta \leq \beta/(6kw)$ and $\sum_{j \in J} \gamma_j \geq \beta/k$, it follows that $4 \leq |J| \leq 1/\eta$. Again from Inequality (6.2) and then averaging, it follows that for at least four of $j \in J$, $\gamma_j \geq \beta\eta/2k$.

From our claim, we know that $\alpha_j = \gamma_j/\eta \geq \beta/2k$ for at least one of these four j .

□

COROLLARY 6.4. *If H is not within L^1 distance β of linear, there exists a j such that $\alpha_j \geq \beta/2k$.*

Proof sketch of Theorem 6.1. As in the uniform case, we estimate the value of α_i from the data stream.

1. Let interp_H be some estimate of interp_H , such that

$$\int_{j\eta}^{(j+1)\eta} \left| \text{interp}_{\hat{H}}(x) - \text{interp}_H(x) \right| dx = O\left(\frac{\beta\eta^2}{k}\right) \quad (6.3)$$

for all j (we will show how to compute $\text{interp}_{\hat{H}}$ later).

2. Let $N_i = |X \cap (i\eta, (i+1)\eta)|$.
3. Let $\hat{\alpha}_i = \left| \frac{N_i}{n\eta} - \frac{1}{\eta} \int_{i\eta}^{(i+1)\eta} \text{interp}_{\hat{H}}(x) dx \right|$.

If the data stream contains $n = \tilde{\Omega}(k^5 w^2 / \beta^5)$ points, then we can decide if any α_i satisfies $\alpha_i \geq \beta/2k$ by approximating $\|\hat{\alpha}\|_1$ using Indyk's algorithm as we did for **CONSTANT**. With Lemma 6.2 and Corollary 6.1 in lieu of their counterparts from Section 4.2, the proof of correctness is the same as for Theorem 4.1, except for the added (but straightforward) accounting of the error caused by estimating $\text{interp}_H(x)$.

All that remains to be shown is how to find the estimate $\text{interp}_{\hat{H}}$ such that it satisfies inequality (6.3). In order to do so, we estimate the values of h_0 and $h_{\frac{1}{\eta}-1}$. Our estimates are $\hat{h}_0 = N_0/(n\eta)$ and $\hat{h}_{1/\eta-1} = N_{1/\eta-1}/(n\eta)$, respectively, which can be computed in the pass simultaneously with Indyk's algorithm in small space. (Note that when we use Indyk's algorithm to approximate α_i , we do not need to know $\text{interp}_{\hat{H}}$ until after the data stream has been examined. See the proof of Lemma 4.13.) Since $n = \tilde{\Omega}(k^2 w^2 / (\beta^2 \eta^3 \log(1/\delta)))$, it follows from the Chernoff bound that $|N_i - n\eta h_i| = O(\beta\eta/k)h_i\eta n$, for $i = 0, 1/\eta - 1$. Thus, $|\hat{h}_i - h_i| = O(\beta\eta/k)$ for $i = 0, 1/\eta - 1$, under the assumption that $h_i \leq \max_x H(x) \leq w$. By Equation (6.1), our error in calculating the slope and intercept of $\text{interp}_H(x)$ will be at most $O(\beta\eta/k)$. As desired, for all j we have

$$\int_{j\eta}^{(j+1)\eta} \left| \text{interp}_H(x) - \text{interp}_{\hat{H}}(x) \right| dx = 2O(\beta\eta/k)\eta = O(\beta\eta^2/k).$$

REMARK When we use **LINEAR** as a subroutine for our algorithm, we will need to determine if some subinterval $I \subseteq [0, 1]$ of the domain of H is close to linear. Call the length of this subinterval l . Testing I for linearity can be done with the same algorithm as above, except we only analyze the subinterval (without scaling it to be $[0, 1]$). In this case, the η smoothed distribution won't consist of $1/\eta$ intervals, but rather only l/η intervals. We compute the line $\text{interp}_{H|_I}$ from the first and last of the l/η intervals. □

6.2. The algorithm. The 2ℓ pass algorithm for finding a function G that approximates F to within L^1 distance ϵ^ℓ is the same as **SMALLRAM**, except we use subroutine **LINEAR** in lieu of **CONSTANT**, and for those intervals J that we do not mark (i.e. they are within $O(\epsilon^\ell/k)$ of linear) we approximate F by $\text{interp}_{F|_J}$, which we showed how to find accurately in the previous section.

THEOREM 6.5. *There exists an algorithm that, with probability at least $1 - \delta$, can learn a mixture of k linear distributions on $(0, 1)$ within L^1 distance ϵ^ℓ using at*

most $\tilde{O}(k^3/\epsilon^2)$ bits of memory. The algorithm requires the data stream to have size $\tilde{\Omega}(k^5w^5/\epsilon^5)$. The proof of the theorem follows the proof of Theorem 4.6.

7. Learning mixtures of bounded uniform distributions in two dimensions. We now consider the problem where F is the density function of a mixture of k uniform distributions on axis-aligned rectangles in $(0, 1) \times (0, 1) \subset \mathbb{R}^2$. Again, we assume that the density of the mixture is bounded by a scalar $w > 0$. We call such a distribution a *mixture of k uniform(2) distributions*. Analogously, we will call a mixture of uniform distributions over intervals in \mathbb{R} a *mixture of uniform(1) distributions*. Note that if the mixture is contained in a rectangle larger than $(0, 1) \times (0, 1)$, in order to apply the algorithm it is necessary to scale the rectangle down and correspondingly scale w up.

F can be described by k axis-aligned rectangles, $R_i \subseteq (0, 1) \times (0, 1)$, $i = 1, \dots, k$, each with weight w_i . Our algorithms will handle the general case where the rectangles intersect. Note that each of these rectangles is defined by four boundary lines.

DEFINITION 7.1. *We say that F is vertical on an axis-aligned rectangle R if R does not contain any of the horizontal boundary lines of the R_i s that define F .*

7.1. Testing rectangles for verticality. Our algorithm will require an analog to Theorem 4.1. In this section we will prove the following Theorem, using the same technique as in the proof of Theorem 4.1. In \mathbb{R}^2 , the L^1 distance between two functions $F, G : \mathbb{R}^2 \rightarrow \mathbb{R}$ is defined by $\int_{\mathbb{R}^2} |F - G|$.

THEOREM 7.2. *Let H be the density function of a mixture of k uniform(2) distributions over universe $R \subseteq \mathbb{R}^2$, let $w > 0$ be an upper bound on H (i.e. $H(x) \leq w$ for all $x \in \mathbb{R}^2$) and let X be a data stream of samples drawn according to H , such that $|X| = \tilde{\Omega}(w^2k^{12}/\beta^{12})$. There exists a single pass algorithm that with probability $1 - \delta$ will **accept** if H is vertical on R and will **reject** if H is not within L^1 distance β of a vertical mixture of $8k$ -uniform(2) distributions, using at most $O((\log 1/\beta + \log k + \log w) \log(1/\delta))$ bits of memory. We may assume that $R = (0, a) \times (0, b)$ is a rectangle. We first define the ζ -area partition of R , which is a two dimensional analog to the partition associated with the η -smoothed distribution.*

DEFINITION 7.3. *Given a rectangle $R = (0, a) \times (0, b)$, a ζ -area partition of R is a partition of R into a set of $1/\zeta$ vertical rectangles of equal area, $\{V_i\}$, such that $V_i = (i\zeta a, (i+1)\zeta a) \times (0, b)$.*

See Figure 7.1a. Let the $\beta/8kw$ -area partition of R be given by the set of rectangles $\{V_i\}$. Given a rectangle $S = (a_1, a_2) \times (b_1, b_2) \subseteq R$, define the function $\tilde{H}_S : (b_1, b_2) \rightarrow \mathbb{R}$ by $\tilde{H}_S(y) = \int_{a_1}^{a_2} H(x, y) dx$.

LEMMA 7.4. *If H is vertical on R , then each \tilde{H}_{V_i} is a constant function, for all V_i in the ζ -area partition of R .*

LEMMA 7.5. *If $\zeta < \beta/(8kw)$ and each \tilde{H}_{V_i} is within L^1 distance $\beta^2/16wk$ of a constant function for all V_i , then H is within L^1 distance β of a vertical mixture of at most $8k + 1$ uniform(2) distributions on R .*

Proof. Assume the condition of the lemma holds. We will show the existence of a vertical mixture of $8k + 2$ uniform(2) distributions whose density function G is within L^1 distance β of H .

Recall that H is a mixture of uniform distributions on k different axis-aligned rectangles. Each of these constituent rectangles has four corners. At most $2k$ of the V_i s contain corners of rectangles.

CLAIM: Assuming the condition of the lemma, if V_i and V_{i+1} are adjacent rectangles of the ζ -area partition that do not contain corners, then H on $V_i \cup V_{i+1}$ can be

approximated by a constant function with error in L^1 distance at most $2\beta^2/16wk$.

Proof of claim. Since V_i does not contain any corners of mixture rectangles, it does not contain any vertical boundary edges (since V_i itself takes up an entire vertical strip of R). Furthermore, any horizontal boundary edge must cut through all of V_i . Thus, $H(x_1, y) = H(x_2, y) = \tilde{H}_{V_i}(y)$ for all $(x_1, y), (x_2, y) \in V_i$. If $\tilde{G}_{V_i}(y) = c$ is a constant approximation to the function \tilde{H}_{V_i} with L^1 error at most $\beta^2/16wk$, then the constant function $G_{V_i}(x, y) = c/\zeta$ is a good approximation to H in V_i : $\int_{V_i} |H - G_{V_i}| \leq \beta^2/(16wk)$.

If V_i and V_{i+1} are adjacent strips such that neither contain corners of endpoints, then it follows that $H(x_1, y) = H(x_2, y) = \tilde{H}_{V_i}(y) = \tilde{H}_{V_{i+1}}(y)$, for all $(x_1, y), (x_2, y) \in V_i \cup V_{i+1}$. Thus, the function $G_{V_i \cup V_{i+1}} = c/\zeta$ will be within L^1 distance $2\beta^2/16wk$ from H on $V_i \cup V_{i+1}$. See Figure 7.1b.

Using induction, it can be shown that H can be approximated by a constant function on m adjacent partition rectangles that do not contain corners with error at most $m\beta^2/16kw$.

We now show how we can partition R into $8k + 1$ vertical rectangles and assign a constant value on each rectangle, such that the induced function on R is within β in L^1 distance of H : **(a)** H on each of the at most $2k$ V_i s that contains a corner can be estimated arbitrarily by 0. The total weight of H in each V_i is at most $w\zeta \leq \beta/8k$. Since there are at most $2k$ of these V_i s, the total error is at most $2k \cdot \beta/8k = \beta/2$.

(b) The remaining kw/β V_i s can be partitioned into a set of at most $4k + 1$ larger rectangles, such that H on each large rectangle T that contains t of the V_i s can be estimated by a constant with error at most $t\beta^2/16wk$. Since there are at most $8kw/\beta$ of the V_i s, the total error is at most $\beta/2$. \square

The algorithm and proof of Theorem 7.2 are a relatively straightforward generalization of the algorithm and proof of Theorem 4.1, and therefore we only provide a sketch.

Proof sketch of Theorem 7.2.

We will design our algorithm to accept if all \tilde{H}_{V_i} are constant, and reject if at least one \tilde{H}_{V_i} is not within L^1 distance $\beta^2/16kw$ of constant. If H is vertical on R , then our algorithm will accept since all \tilde{H}_{V_i} will be constant; if H is not within β of a vertical mixture of $8k$ rectangles, then our algorithm will reject.

We now show how to determine whether or not all of the distributions defined by the \tilde{H}_{V_i} s are close to constant. The crucial observation is that since each \tilde{H}_{V_i} is a mixture of k uniform(1) distributions over \mathbb{R} , we can determine if it is close to uniform in a manner similar to CONSTANT. We outline the details below.

Set $\eta = \beta^2/80k^2w$ and let the η -smoothed distribution of \tilde{H}_{V_i} be $\tilde{H}_{V_i\eta}$. Let the intervals defined by the η -smoothed distribution be $(x_0^i, x_1^i), (x_1^i, x_2^i), \dots, (x_{1/\eta-1}^i, x_{1/\eta}^i)$ and h_j^i be the density of $\tilde{H}_{V_i\eta}$ on the interval (x_j^i, x_{j+1}^i) , where $x_j^i = x_0 + j\eta$.

As in Section 4.2.1, it can be shown that if $\alpha_j^i = \left| h_j^i - \zeta \int_{(x_j^i, x_{j+1}^i) \times (b_1, b_2)} H \right| \leq \beta^2/32wk^2$ for all j , then \tilde{H}_{V_i} is within $\beta^2/16wk$ of uniform. Thus, our algorithm determines whether or not $\alpha_j^i \leq \beta^2/32wk^2$ for all i and j . In order to estimate each α_j^i , we define the following random variables:

1. $N_j^i = |X \cap (V^i \cap [(0, a) \times (x_j^i, x_{j+1}^i)])|$.
2. $\hat{\alpha}_j^i = |N_j^i/n\eta - \sum_j N_j^i/n|$. Note that $\mathbf{E}[\hat{\alpha}_j^i] = \alpha_j^i$.

If we require that $|X| = \tilde{\Omega}(w^2 k^4 / (\beta^4 \eta^4 \zeta^2)) = \tilde{\Omega}(w^2 k^{12} / \beta^{12})$, then from the VC bound, for all i, j it follows that $|N_j^i / \eta n - h_j^i| \leq \beta^2 \eta \zeta / 200 w k^2$ and that $|N_j^i / n - \int_{V_i} H| \leq \beta^2 \eta \zeta / 200 w k^2$.

Thus, $|\alpha_j^i - \hat{\alpha}_j^i| \leq \beta^2 \eta \zeta / 100 w k^2$. Let α be the vector of dimension $1/\eta \zeta$ whose components consist of all the α_j^i s. As we argued before, if all $1/\eta \zeta$ of the $\alpha_j^i = 0$, then $|\hat{\alpha}_j^i| \leq \beta^2 \eta \zeta / 100 w k^2$ for all i, j , which implies that $\|\hat{\alpha}\|_1$ is less than $\beta^2 / 100 w k^2$. If some $|\alpha_j^i| \geq \beta^2 / 32 w k^2$, then this implies that some $|\hat{\alpha}_j^i| \geq \beta^2 / 50 w k^2$. This implies that $\|\hat{\alpha}\|_1 \geq \beta^2 / 50 w k^2$. Thus, if our algorithm accepts if $\|\hat{\alpha}\|_1 \leq \beta^2 / 100 w k^2$, rejects if $\|\hat{\alpha}\|_1 \geq \beta^2 / 50 w k^2$, and otherwise acts arbitrarily, then it will accept if H is vertical and will reject if it is not within β of vertical.

We use Indyk's algorithm to approximate $\|\hat{\alpha}\|_1$ to within a factor of $1/4$ in a single pass using space $O((\log 1/\beta + \log k + \log w) \log 1/\delta)$. With this solution, we can solve the above problem. \square

7.2. The Algorithm. We show how to modify SMALLRAM in order to learn a mixture of k uniform(2) distributions. Since the algorithm and its proof of correctness are similar to SMALLRAM and its proof, we only provide a sketch.

THEOREM 7.6. *Let F be the density function of a mixture of k uniform(2) distributions, such that $F(x) \leq w$ for all $x \in \mathbb{R}^2$. Suppose X is a data stream drawn according to F , with a sufficient number of elements. There exists a $4\ell + 2$ -pass algorithm that approximates F to within L^1 distance ϵ^ℓ using at most $\tilde{O}(\ell k^{4+1/\ell} / \epsilon^3 + \ell^2 k / \epsilon)$ bits of memory. *Proof sketch.* The main algorithm can be organized into ℓ recursive*

calls, each of which requires two passes. At a p th level call, the input is a rectangle $R^p = (0, a) \times (0, b) \subseteq R$. We give an outline of the recursive algorithm:

1. In the first pass, we draw a sample of size m from $X|_{R^p}$. If $p = 1$, we set $m = \Theta(\epsilon^2 / k^2)$; for subsequent p , we set $m = \Theta(\epsilon^2)$. We use this sample to partition R^p into a set of subrectangles $\{R_i\}$, such that
 - (a) Each rectangle R_i can be written as $(0, a) \times (x_i^1, x_i^2)$ (i.e. they are "horizontal").
 - (b) Each rectangle has at least 0.8ϵ (or $0.8\epsilon/(4k)$ if $p = 1$) proportion of the weight of F in R^p and at most 0.9ϵ (or $0.9\epsilon/(4k)$ if $p = 1$). With this condition, it can be shown that $0.8^p \epsilon^p / (4k) \leq \int_{R_i} F \leq 0.9^p \epsilon^p / (4k)$.
2. In the second pass of each pair of passes, we check each of the $\Theta(k/\epsilon)$ subrectangles R_i to see if it is within $\epsilon^{\ell+1}/(16k)$ of vertical using the one pass algorithm from Section 7.1. We set the failure probability to be at most $1 - O(\delta \epsilon^{c\ell} k^c)$ for some constant c . With these input parameters, each call to the subroutine will require $O((\log 1/\epsilon + \log k + \log w) \ell^2 \log(k/\epsilon \delta))$ bits of memory. We mark the R_i s that are rejected. Note that since the R_i s are horizontal rectangles, at most $2k$ of them can contain horizontal boundary lines; thus, at most $2k$ of them are marked.
3. If F is within $\epsilon^{\ell+1}/(16k)$ of a mixture of $8k$ vertical uniform(2) distributions in R , then projecting points onto the x axis will form a mixture of at most $8k$ uniform(1) distributions. Call this distribution \tilde{F}_{R_i} . We can learn \tilde{F}_{R_i} to within L^1 distance $\epsilon^{\ell+1}/(16k)$ by calling our $2(\ell + 1)$ pass algorithm SMALLRAM from the previous section. Call this approximation \tilde{F}'_{R_i} and set $G_{R_i}(x, y) = \tilde{F}'_{R_i}(x)$ for all $(x, y) \in R_i$, to be our approximation to F in R_i .

We claim that

$$\int_{(x,y) \in R_i} |G_{R_i}(x,y) - F(x,y)| \leq \frac{\epsilon^{\ell+1}}{4k}.$$

It is straightforward to prove that \tilde{F}'_{R_i} is within L^1 distance $\epsilon^{\ell+1}/8k$ of the distribution constructed in the proof of Lemma 7.5. Since this distribution is within L^1 distance $\epsilon^{\ell+1}/8k$ of F in R , by the triangle inequality we know that F is within L^1 distance $\epsilon^{\ell+1}/4k$ of our approximation \tilde{F}'_{R_i} .

4. In parallel, recursively approximate the density of F in each of the rectangles R_i that were marked in Step 2.

The above algorithm can be implemented in $4\ell + 2$ passes; the $2\ell + 2$ passes used by each call to **SmallRam** can be run in parallel with the passes of the main algorithm. The algorithm needs $\tilde{O}(\ell k^{4+1/\ell}/\epsilon^3 + \ell^2 k^2/\epsilon^2)$ bits of memory, since each call to **SMALLRAM** uses $\tilde{O}(k^{3+1/\ell}/\epsilon^2 + \ell k/\epsilon)$ bits of memory, and there are at most $O(\ell k/\epsilon)$ parallel calls to **SMALLRAM**.

The proof that the total error of the function output by the algorithm does not exceed ϵ^ℓ is similar to the analysis of the error of **SMALLRAM**. \square

We arrive at the following corollary by transforming ϵ .

COROLLARY 7.7. *There exists a $4\ell + 2$ pass algorithm that will learn F to within L^1 distance ϵ using at most $\tilde{O}(k^{4+1/\ell}/\epsilon^{3/\ell})$ bits of memory.*

8. Conclusions and further research. We have presented pass-efficient algorithms for the massive data set problem of approximately learning a mixture of k uniform distributions. An interesting feature of our work is the fact that our multiple pass algorithm has flexible performance guarantees: the number of passes allotted to the algorithm is an input parameter, such that each increase in this parameter will substantially decrease the error of the computation while holding memory requirements constant. Thus, we have demonstrated that allowing a streaming algorithm to make more than one pass over the data can substantially enhance its performance. Furthermore, we have proved that for a suitable generalization of our problem and strengthening of the algorithm this tradeoff is close to optimum.

Direct areas of further work involve designing massive data set algorithms for learning other types of distributions. For example, designing algorithms for learning mixtures of Gaussian distributions in high dimension is a natural and compelling extension of this work. Furthermore, the general application of massive data set algorithms to problems in machine learning is an area with great theoretical promise and of great interest to practitioners.

REFERENCES

- [1] A. Abounaga and S. Chaudhuri. Self-tuning histograms: Building histograms without looking at data. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 181–192, 1999.
- [2] D. Achlioptas and F. McSherry. Fast computation of low rank matrix approximations. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, pages 611–618, 2001.
- [3] 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.
- [4] S. Arora and R. Kannan. Learning mixtures of separated nonspherical Gaussians. *Annals of Applied Probability*, 15(1A):69–92, 2005.

- [5] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.
- [6] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36(1):929–965, 1989.
- [7] K. Chang. Multiple pass streaming algorithms for learning mixtures of distributions in R^d . In *Proceedings of the 18th International Conference on Algorithmic Learning Theory*, pages 211–226, 2007.
- [8] S. Chaudhuri, R. Motwani, and V. R. Narasayya. Random sampling for histogram construction: How much is enough? In *SIGMOD Conference*, pages 436–447, 1998.
- [9] S. Dasgupta. Learning mixtures of Gaussians. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 634–644, 1999.
- [10] P. Drineas and R. Kannan. Pass efficient algorithm for large matrices. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 223–232, 2003.
- [11] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. Graph distances in the streaming model: The value of space. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 745–754, 2005.
- [12] P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. In *Proceedings of 23th International Conference on Very Large Data Bases*, pages 266–275, 1997.
- [13] A. C. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *Proceedings of the 34th Annual ACM Symposium on the Theory of Computing*, pages 389–398, 2002.
- [14] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *Proceedings of 27th International Conference on Very Large Data Bases*, pages 79–88, 2001.
- [15] S. Guha and N. Koudas. Approximating a data stream for querying and estimation: Algorithms and performance evaluation. In *Proceedings of the 18th International Conference on Data Engineering*, pages 567–, 2002.
- [16] S. Guha, N. Koudas, and K. Shim. Data-streams and histograms. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, pages 471–475, 2001.
- [17] S. Guha and A. McGregor. Lower bounds for quantile estimation in random-order and multi-pass streaming. In *Proceedings of the 34th International Colloquium on Automata, Languages, and Programming*, pages 704–715, 2007.
- [18] S. Guha and A. McGregor. Space-efficient sampling. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, 2007.
- [19] M. R. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science: External Memory Algorithms*, 50:107–118, 1999.
- [20] P. Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. In *Proceedings of the 41th IEEE Symposium on Foundations of Computer Science*, pages 189–197, 2000.
- [21] H. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. Optimal histograms with quality guarantees. In *Proceedings of 24th International Conference on Very Large Data Bases*, pages 275–286, 1998.
- [22] Y. Matias, J. S. Vitter, and M. Wang. Dynamic maintenance of wavelet-based histograms. In *Proceedings of 26th International Conference on Very Large Data Bases*, pages 101–110, 2000.
- [23] P. B. Miltersen, N. Nisan, S. Safra, and A. Wigderson. On data structures and asymmetric communication complexity. *Journal of Computer and System Sciences*, 57(1):37–49, 1998.
- [24] J. I. Munro and M. Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12:315–323, 1980.
- [25] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.
- [26] M. Talagrand. Sharper bounds for Gaussian and empirical processes. *Annals of Probability*, 22(1):28–76, 1994.
- [27] V. Vapnik and A. Červonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.
- [28] S. Vempala and G. Wang. A spectral algorithm for learning mixtures of distributions. *Journal of Computer and System Sciences*, 68(4):841–860, 2004.

Input: data stream X , such that $|X| \geq m$ (defined below).

1. In one pass, draw a sample of size

$$m = O\left(\frac{k^2 \log^2(k/\epsilon) + \log^2 \log(1/\delta)}{\epsilon^2} \left(\log \frac{k}{\epsilon} + \log \frac{1}{\delta}\right)\right) \quad (3.1)$$

from X , where m is a power of 2. Sort the samples in memory, and call the set S .

2. Call CLUSTER on S , and let the resulting intervals be $\tilde{J}_1, \dots, \tilde{J}_v$.
3. Output the approximate density function, G , as the step function given by the intervals $\{\tilde{J}_i\}$ with densities $|S \cap \tilde{J}_i|/(m \text{length}(\tilde{J}_i))$.

Subroutine CLUSTER

Input: A set of t sorted points, $T = \{s_1, \dots, s_t\}$.

Base Case: If $t < 3$, then output interval (s_1, s_{t+1}) , where s_{t+1} is the point following s_t in S . Note that $T \subseteq (s_1, s_{t+1})$.

Main Recursion:

1. Set $T_1 = \{s_1, \dots, s_{t/2}\}$ and $T_2 = \{s_{t/2+1}, \dots, s_t\}$
2. Call CLUSTER on T_1 , and let the resulting intervals be I_1, \dots, I_{k_1} .
3. Call CLUSTER on T_2 and let the resulting intervals be J_1, \dots, J_{k_2} .
4. Set $K = I_{k_1} \cup J_1$.
5. If the empirical densities of both I_{k_1} and J_1 are consistent with K on T , then output the set of intervals $\{I_1, \dots, I_{k_1-1}, K, J_2, \dots, J_{k_2}\}$. Otherwise, output the set of intervals $\{I_1, \dots, I_{k_1}, J_1, \dots, J_{k_2}\}$

FIG. 3.1. *Algorithm ONEPASS*

Input: data stream X of samples from distribution with density F , such that

$$n = |X| = \Omega \left(\frac{1.25^\ell k^6}{\epsilon^{6\ell}} \cdot \ell \cdot \log \left(\frac{k\ell}{\delta\epsilon} \right) \right) \quad (4.1)$$

1. Initialize $p \leftarrow 1$, $J \leftarrow \text{Support}(F)$ (Note that J can be found in the first pass of the algorithm).
2. Call ESTIMATE on p, J .

Subroutine ESTIMATE

Input: interval J , and level p .

Main Recursion:

1. In a single pass, draw a sample S of size $m = \Theta \left(\frac{k^2}{\epsilon^2} \log \left(\frac{\ell k}{\epsilon \delta} \right) \right)$ from $X|_J$, and store it in memory.
2. Sort the samples in S in ascending order. If $p = 1$, set $q \leftarrow \frac{9\epsilon}{20k}m$; otherwise set $q \leftarrow (9/10) \cdot \epsilon m$.
3. Partition J into m/q disjoint open intervals $\{J_i^p\}$, such that $|S \cap J_i^p| = q$ for all i .
4. In a second pass, count the exact number of points of X , the entire data stream, that fall in each of the J_i^p s.
5. Also in the second pass, check if F is near uniform on interval J_i^p by calling CONSTANT on J_i^p with error parameter $\epsilon^\ell/2k$, for each J_i^p in parallel. Mark those intervals J_i^p that CONSTANT rejects. Set M^p be the set of marked intervals.
6. If interval J_i^p is not marked in the previous step, output it as a step of G with density $|X \cap J_i^p|/(n \cdot \text{length}(J_i^p))$.
7. If $p < \ell$, for each $J_i^p \in M^p$, run ESTIMATE on $J \leftarrow J_i^p$, $p \leftarrow p + 1$, in parallel with all other level $p + 1$ copies of ESTIMATE.
If $p = \ell$, output each interval $J_i^p \in M^p$ as a step of G with density 0.

FIG. 4.1. Algorithm SMALLRAM

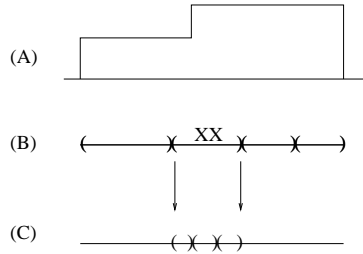


FIG. 4.2. Zooming. (A) A step distribution. (B) Intervals created in Step 3 of some call to ESTIMATE. One of them contains a jump and has been marked. (C) ESTIMATE is recursively called on the marked interval.

Input: data stream X such that

$$|X| = \tilde{\Omega} \left(\left(\frac{k}{\beta} \right)^2 \frac{1}{\eta^3} \log \frac{1}{\delta} \right) = \tilde{\Omega} \left(\frac{k^5}{\beta^5} \log \frac{1}{\delta} \right).$$

1. Set $\eta \leftarrow \beta/(5k)$.
2. Simulate Indyk's algorithm on X , as described in Lemma 4.13. Let ζ be the estimate of $\|\hat{\alpha}\|_1$ output by the algorithm.
3. **accept** if $\zeta \leq \beta/5k$. Otherwise, **reject**.

FIG. 4.3. *Algorithm CONSTANT*

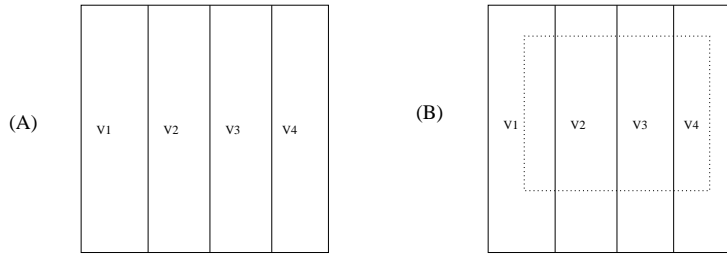


FIG. 7.1. (A) The ζ -area partition of a rectangle. (B) A rectangle of the mixture given by dotted lines, superimposed on the partition. Note that since V_2 and V_3 are adjacent rectangles that do not contain corners, their densities are the same.