1

Stable Distributions in Streaming Computations

Graham Cormode[1] and Piotr Indyk[2]

[1] AT&T Labs – Research,
180 Park Avenue, Florham Park NJ, graham@research.att.com

[2] Laboratory for Computer Science,
Massachusetts Institute of Technology, Cambridge MA, indyk@mit.edu

1.1 Introduction

In many streaming scenarios, we need to measure and quantify the data that is seen. For example, we may want to measure the number of distinct IP addresses seen over the course of a day, compute the difference between incoming and outgoing transactions in a database system or measure the overall activity in a sensor network. More generally, we may want to cluster readings taken over periods of time or in different places to find patterns, or find the most similar signal from those previously observed to a new observation. For these measurements and comparisons to be meaningful, they must be well-defined. Here, we will use the well-known and widely used L_p norms. These encompass the familiar Euclidean (root of sum of squares) and Manhattan (sum of absolute values) norms.

In the examples mentioned above—IP traffic, database relations and so on—the data can be modeled as a vector. For example, a vector representing IP traffic grouped by destination address can be thought of as a vector of length 2^{32}, where the ith entry in the vector corresponds to the amount of traffic to address i. For traffic between (source, destination) pairs, then a vector of length 2^{64} is defined. The number of distinct addresses seen in a stream corresponds to the number of non-zero entries in a vector of counts; the difference in traffic between two time-periods, grouped by address, corresponds to an appropriate computation on the vector formed by subtracting two vectors, and so on. As is usual in streaming, we assume that the domain of the data and the size of the data are too massive to permit the direct computation of the functions of interest—which are otherwise mostly straightforward—and instead, we must use an amount of storage that is much smaller than the size of the data. For the remainder of this chapter, we put our description in terms of vectors, with the understanding that this is an abstraction of problems coming from a wide variety of sources.

Throughout, we shall use $\boldsymbol{a}$ and $\boldsymbol{b}$ to denote vectors. The dimension of a vector (number of entries) is denoted as $|\boldsymbol{a}|$.

Definition 1. *The L_p* norm *(for $0 < p \leq 2$) of a vector $\boldsymbol{a}$ of dimension n is*

$$||\boldsymbol{a}||_p = (\sum_{i=1}^{n} |\boldsymbol{a}[i]|^p)^{1/p}$$

The L_0 norm *is defined as*

$$||\boldsymbol{a}||_0 = (\sum_{i=1}^{n} |\boldsymbol{a}[i]|^0) = |\{i|\boldsymbol{a}[i] \neq 0|\}|$$

where 0^0 is taken to be 0.

These are vector norms in the standard sense: the result is non-negative, and zero only when the vector is zero; and the norm of the sum of vectors is less than the sum of their norms. For $p > 0$, the L_p norm guarantees that $\|k\boldsymbol{a}\|_p = k\|\boldsymbol{a}\|_p$ for any scalar k. This does not hold for $p = 0$, so L_0 is not a norm in the strict sense. These norms immediately allows the measurement of the difference between vectors, by finding the norm of the (component-wise) difference between them. To be precise,

Definition 2. *The L_p* distance *between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ of dimension n is the L_p norm of their difference,*

$$||\boldsymbol{a} - \boldsymbol{b}||_p = (\sum_{i=1}^{n} |\boldsymbol{a}[i] - \boldsymbol{b}[i]|^p)^{1/p}$$

The L_p distance encompasses three very commonly used distance measures:

- Euclidean distance, given by L_2 distance, is the root of the sum of the squares of the differences of corresponding entries.
- The Manhattan distance, given by the L_1 distance, is the sum of the absolute differences.
- The Hamming distance, given by the L_0 distance, is the number of non-zero differences.

In this chapter, we will show how all three of the distances can be estimated for massive vectors presented in the streaming model. This is achieved by making succinct *sketches* of the data, which can be used as synopses of the vectors they summarize. This is described in Section 1.2. In Section 1.3 we discuss some applications of these results, to the distinct elements problem, and to computing with objects that can't be modeled as simple vectors. Lastly, we discuss related work and new directions in Sections 1.4 and 1.5.

1.2 Building Sketches using Stable Distributions

1.2.1 Data Stream Model

We assume a very general, abstracted model of data streams where our input arrives as a stream of updates to process. We consider vectors $\boldsymbol{a}, \boldsymbol{b}, \ldots$, which are presented in an implicit, incremental fashion. Each vector has dimension n, and its current state at time t is $\boldsymbol{a}(t) = [\boldsymbol{a}(t)[1], \boldsymbol{a}(t)[2], \ldots \boldsymbol{a}(t)[n]]$. For convenience, we shall usually drop t and refer only to the current state of the vector. Initially, $\boldsymbol{a}$ is the zero vector, $\mathbf{0}$, so $\boldsymbol{a}(0)[i]$ is 0 for all i. Updates to individual entries of the vector are presented as a stream of pairs. The tth update is (i_t, c_t), meaning that

$$\begin{aligned} \boldsymbol{a}(t)[i_t] &= \boldsymbol{a}(t-1)[i_t] + c_t \\ \boldsymbol{a}(t)[j] &= \boldsymbol{a}(t-1)[j] \qquad j \neq i_t \end{aligned}$$

For the most part, we expect the data to arrive in no particular order, since it is unrealistic to expect it to be sorted on any attribute. We also assume that each index can appear many times over in the stream of updates. In some cases, c_ts will be strictly positive, meaning that entries only increase; in other cases, c_ts are allowed to be negative also. The former is known as the *cash register* case and the latter the *turn-stile* case [35]. Here, we assume the more general case, that the data arrives unordered and each index can be updated multiple times within the stream.

1.2.2 Stable Distributions

Stable Distributions are a class of statistical distributions with properties that allow them to be used in finding L_p norms. This allows us to solve many problems of interest on data streams. A stable distribution is characterized by four parameters (following [38]), as follows:

- The stability parameter $\alpha \in (0, 2]$.
- The skewness parameter $\beta \in [-1, 1]$.
- The scale parameter $\gamma > 0$.
- The shift parameter δ.

Although there are different parameterizations of stable distributions, we shall fix values of β, γ and δ for this discussion. This has the effect that the different parameterization systems all coincide. We set $\beta = 0$, which makes the distribution symmetric about its mode. Setting $\gamma = 1$ and $\delta = 0$ puts the mode of the distribution at 0 and gives a canonical distribution. Formally then, the distributions we consider are *symmetric and strictly stable*, but we shall simply refer to them as stable.

Definition 3. *A* (strictly) stable distribution *is a statistical distribution with parameter α in the range $(0, 2]$. For any three independent random variables X, Y, Z drawn from such a distribution, for scalars a, b, $aX + bY$ is distributed as $(|a|^\alpha + |b|^\alpha)^{1/\alpha} Z$.*

These are called stable distributions because the underlying distribution remains stable as instances are summed: the sum of stable distributions (with the same α) remains stable. This can be thought of a generalization of the central limit theorem, which states that the sum of distributions (with finite variance) will tend to a Gaussian distribution. Note that, apart from $\alpha = 2$, stable distributions have unbounded variance.

Several well-known distributions are known to be stable. The Gaussian (normal) distribution, with density $f(x) = \frac{e^{-x^2/2}}{\sqrt{2\pi}}$, is strictly stable with $\alpha = 2$. The Cauchy distribution, with density $f(x) = \frac{1}{\pi(1+x^2)}$, is strictly stable with $\alpha = 1$. For all values of $\alpha \leq 2$, stable distributions can be simulated by using appropriate transformations from uniform distributions, as we will show later.

1.2.3 Sketch Construction

By applying the above definition iteratively, we find that

Corollary 1. *Given random variables $X_1, X_2, \ldots X_n$ independently and identically distributed as X, a strictly stable distribution with stability parameter $\alpha = p$, and a vector $\boldsymbol{a}$, then $S = \boldsymbol{a}[1]X_1 + \boldsymbol{a}[2]X_2 + \ldots + \boldsymbol{a}[n]X_n$ is distributed as $||\boldsymbol{a}||_p X$.*

From this corollary, we get the intuition for why stable distributions are helpful in computing L_p norms and L_p distances: by maintaining the inner product of variables from α-stable distributions with a vector being presented in the stream, we get a variable S which is distributed as a stable distribution scaled by the L_p norm of the stream, where $p = \alpha$. Maintaining this inner product as the vector undergoes updates is straightforward: given an update (i, c), we simply add $X_i \cdot c$ to S. However, what we have so far is an equality *in distribution*; what we are aiming for is an equality *in value*. To solve this problem, we proceed as follows.

Let $\text{med}(X)$ denote the *median* of X, i.e., a value M such that $\Pr[X > M] = 1/2$. Then, for any $s > 0$, we have $\text{med}(s \cdot |X|) = s \cdot \text{med}(|X|)$. In our case, $\text{med}(|S|) = \text{med}(\|\boldsymbol{a}\|_p \cdot |X|) = \|\boldsymbol{a}\|_p \text{med}(|X|)$. Moreover, $\text{med}(|X|)$ depends only on p and can be precomputed in advance. For $\alpha = 1$ and $\alpha = 2$ then $\text{med}(|X|) = 1$. For other values of α, $\text{med}(|X|)$ can be found numerically: Figure 1.1 shows a plot of the median values found by simulation, where each point represents one experiment of taking the median of 10,000 drawings, raised to the power $\frac{1}{p}$. Thus, in order to estimate $\|\boldsymbol{a}\|_p$, it suffices to compute an estimation z of $\text{med}(|S|)$, and then estimate $\|\boldsymbol{a}\|_p$ by $\frac{z}{\text{med}(|X|)}$.

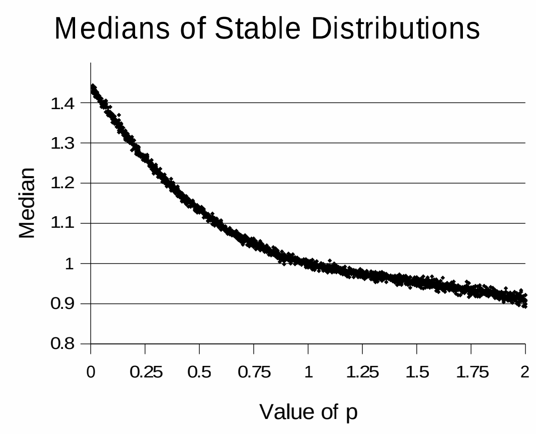

Fig. 1.1. Plot of empirically found median of stable distributions varying stability parameter p in the range 0 to 2

To estimate $M = \text{med}(|S|)$, we take a vector $sk[1] \dots sk[m]$ of independent samples of the random variable S. In other words, for each i, we "generate" m independent samples $x_i^1 \dots x_i^m$ of the X_i, and then compute $sk[j] = \boldsymbol{a}[1]x_1^j + \dots + \boldsymbol{a}[n]x_n^j$. We call this vector sk an *α-stable sketch* of the vector $\boldsymbol{a}$. This can be viewed computationally as maintaining each entry of the sketch as the inner product between the vector and appropriately chosen random vectors, i.e., $sk[j] = \boldsymbol{a} \cdot x^j$. The procedure is presented in more detail in Figure 1.2. Note that the vector sk can be computed in a streaming fashion; in particular, the numbers x_i^j are not actually stored. It is important that we get the same value for x_i^j every time it is accessed: this is done by using a pseudo-random number generator that is initialized with i to give a stream of values x_i^j. Then we use the following lemma (for the random variable $Z = |S|$, the absolute value of S).

Lemma 1. *For any one-dimensional random variable Z with continuous density, let $F(t) = \Pr[Z \le t]$. There is a constant $C > 0$ such that for $m = \frac{C}{\epsilon^2} \log \frac{2}{\delta}$, if we take m independent samples $z_1 \dots z_m$ of Z and set z to be the median element of the sequence $z_1 \dots z_m$, then*

$$\Pr[F(z) \in [\tfrac{1}{2} - \epsilon, \tfrac{1}{2} + \epsilon]] > 1 - \delta$$

Proof. Let t be such that $F(t) = \frac{1}{2} - \epsilon$. If $F(z) < F(t) = \frac{1}{2} - \epsilon$, then $z_i < t$ for most $z_i's$, and therefore $\frac{|\{i : z_i \ge t\}|}{m} < \frac{1}{2}$. However, for each z_i we have $\Pr[z_i \ge t] = \frac{1}{2} + \epsilon$. Therefore, from Chernoff bound [34] we know that there exists a constant $C > 0$ such that

$$P_1 = \Pr[F(z) < \tfrac{1}{2} - \epsilon] \le \exp(-\tfrac{\epsilon^2 m}{C})$$

Using the same argument we obtain

$$P_2 = \Pr[F(z) > \tfrac{1}{2} + \epsilon] \le \exp(-\tfrac{\epsilon^2 m}{C})$$

By setting $m = \frac{C}{\epsilon^2}\log(2/\delta)$ we obtain $P_1 + P_2 \leq \delta$.

Note that our goal is to obtain an approximation to the median M, i.e., the number such that $F(M) = \frac{1}{2}$, while the above provides us (with probability $1-\delta$) with z such that $F(z) \in [\frac{1}{2}-\epsilon, \frac{1}{2}+\epsilon]$. For general functions F, z could be a bad estimate of M; e.g., if F is “flat” around the point $\frac{1}{2}$. However, if the derivative F' of F is bounded by $\frac{1}{B}$ from below around $\frac{1}{2}$, then the above implies that $z \in [M - B\epsilon, M + B\epsilon]$, i.e., that $z = (1 \pm B\epsilon)M$, which is precisely what we want.

It suffices to verify if the derivative of the function F for the random variable $|S|$ is bounded away from 0 around $\frac{1}{2}$. For $\alpha = \{1, 2\}$, this can be verified analytically. For other values of α, this can be verified computationally, e.g., by plotting F. It should be noted that the lower bound for $F'(\frac{1}{2})$ depends on α, and tends to ∞ as α tends to 0.

1.2.4 Simulating Stable Distributions

When implementing this technique, we need to be able to generate values from a stable distribution. These can be generated by using appropriate transformations from uniform random distributions.

- For $\alpha = 1$, we can use the Cauchy distribution, which is easy to draw from. If U is a uniform random distribution returning values in the range $[0, 1]$, then $\tan(\pi(U - \frac{1}{2}))$ is distributed with the Cauchy distribution.
- For $\alpha = 2$, we can use the Normal distribution, which can be drawn from using the Box-Muller transformation: If U and V are independently distributed uniformly over $[0, 1]$, then $\sqrt{-2\ln U}\cos(2\pi V)$ is distributed as a normal distribution.
- For all other values of $\alpha \in (0, 2)$, stable distributions can be simulated using the method of Chambers, Mallows and Stuck [6]. These take uniform distributions U, V onto the range $[0, 1]$ and output a value drawn from a stable distribution with parameter $\alpha \neq 1$. Set $\theta(U) = \pi \cdot (U - \frac{1}{2})$. Then

$$\texttt{stable}(U, V, \alpha) = \frac{\sin \alpha\theta(U)}{\cos^{1/\alpha}\theta(U)} \left(\frac{\cos(\theta(U)\cdot(1-\alpha))}{-\ln V} \right)^{\frac{1-\alpha}{\alpha}}$$

  is distributed as a stable distribution with parameter α.

1.2.5 The Sketch Algorithm

The full algorithm to compute a sketch of a stream is given in Figure 1.2. It works as follows: lines 1–2 intialize the sketch vector to a vector of all zeros. Then for each new tuple (i, c), we initialize a pseudo-random number generator with the index i (line 4), so that when we draw random values (lines 6–7), these are pseudo-random functions of i, the same every time the same value of

Algorithm to compute sketches of a stream

```
1: for $1 \le j \le m$ do
2:    $sk[j] \leftarrow 0.0$
3: for all tuples $(i, c)$ do
4:    initialize-random-seed($i$)
5:    for $1 \le j \le m$ do
6:       $u \leftarrow$ uniformly-random-from$(0, 1)$
7:       $v \leftarrow$ uniformly-random-from$(0, 1)$
8:       $x_i^j \leftarrow$ stable$(u, v, \alpha)$
9:       $sk[j] \leftarrow sk[j] + c \cdot x_i^j$
10: return median$(|sk[1]|, ..., |sk[m]|)$/med$(|X|)$
```

Fig. 1.2. Sketching algorithm

i is seen in the stream. Each successive call to the random number generator yields a new value, but the sequence of values following each re-initialization is the same. Line 8 takes two values in the range 0 to 1, and transforms them to yield a value drawn from a stable distribution with parameter α. The jth entry of the sketch is updated, by adding on the contribution of the update c times the stable value in line 9. This is repeated for all m entries in the sketch. Lastly, to return an estimate of the norm of the vector, we take the median of the (absolute) values of the sketch, and scale this by the median of the stable distribution with parameter α.

We state a theorem that summarizes the properties of this algorithm.

Theorem 1 (From [26]). *In space $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ we can compute an α-stable sketch of a vector $\boldsymbol{a}$ presented in the turn-stile streaming model. Using this sketch we can compute an estimate of $\|\boldsymbol{a}\|_p$ for $p = \alpha$ that is accurate within a factor of $1 \pm \epsilon$ with probability at least $1 - \delta$. Processing each update to the vector $\boldsymbol{a}$ takes time linear in the size of the sketch, $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$.*

This follows from the above Lemma and the preceding discussion.

Note that to complete the proof we must also argue that we can replace truly random samples x_i^j with values drawn using pseudo-random generators. The proof of this relies on the pseudo-random generators of Nisan [37], and we refer the interested reader to the details in [26]. In practice, it suffices to use standard random number generators to generate uniform pseudo-random numbers, and use the transforms given in the previous section.

1.2.6 Other estimators

In the previous sections we used the median of $sk[1] \dots sk[m]$ to estimate the norm of the stream vector. There are alternative estimators that one can use instead. In particular, for the L_1 norm, Li et al [32] proposed the following *bias-corrected geometric mean* estimator:

$$E = \cos^m(\frac{\pi}{2m}) \prod_{j=1}^{m} |sk[j]|^{1/m}$$

This estimator is more accurate than the median estimator when the sample size is small [32].

A similar estimator can be used to estimate the more general L_p norms, $p \in (0, 2]$. Unlike the median estimator (which requires some computation to determine the right parameters of the distribution function F), the geometric mean estimator is computable using a simple analytical formula. See [31] for more details.

1.2.7 Combining Sketches

We now state a number of the properties of this sketching technique, which follow immediately from the method of their construction. These show how the α-stable sketches have application to a variety of circumstances.

Corollary 2.

$$sk(\boldsymbol{a} + \boldsymbol{b}) = sk(\boldsymbol{a}) + sk(\boldsymbol{b})$$
$$sk(\boldsymbol{a} - \boldsymbol{b}) = sk(\boldsymbol{a}) - sk(\boldsymbol{b})$$

These two facts follow immediately from the fact that the sketches are generated as the inner product between the vector $\boldsymbol{a}$ and vectors of values drawn from random distributions, x_i^j. So, the sketch of the sum of two vectors can be computed from the sum of their sketches. This allows the distributed computation of sketches by multiple parties: after agreeing in advance on a random number generator to use, sketches of different data can be computed in parallel, and then the sketches combined to get the sketch of the sum of the data. Similarly, the sketch of the difference of two vectors, and hence the L_p distance between them, can be computed from sketches of the original vectors. This allows large data sets to be compared by only storing the short summarizing sketches of them.

Corollary 3.

$$sk(c \cdot \boldsymbol{a}) = c \cdot sk(\boldsymbol{a})$$

Also by the linearity of construction, the sketch of a vector $\boldsymbol{a}$ scaled by a scalar c can be computed directly from the sketch of the original vector. This allows, for example, a new day's set of data to be compared against the *average* of the previous weeks data: the sketch of the average is computed by summing the sketches of seven days data, and scaling by $\frac{1}{7}$. Similarly, the popular *exponential decay* model where we compute a weighted average of previous vectors $\boldsymbol{a}(0), \boldsymbol{a}(1), \boldsymbol{a}(2) \ldots$ as $(1-\lambda)(\boldsymbol{a}(0)+\lambda\boldsymbol{a}(1)+\lambda^2\boldsymbol{a}(2)+\ldots+\lambda^i\boldsymbol{a}(i)+\ldots)$ $(0 < \lambda < 1)$ is easy to construct iteratively. Suppose we have a sketch of the current vector sk, and wish to include $\boldsymbol{a}$ as the new day's data. Then we can set $sk[j] \leftarrow (1-\lambda)sk(\boldsymbol{a})[j] + \lambda sk[j]$ for all j.

1.3 Application to Streaming Problems

In this section, we outline some of the applications within streaming and beyond that stable distributions have been used to address. These include: estimating the number of distinct items in a stream; as a way to track embeddings in small space; and for geometric problems such as clustering and approximate nearest neighbor searching.

1.3.1 L_0 and counting distinct items

Suppose we are shown a sequence of items, and want to know how many *distinct* items there are in the sequence. This is a fundamental question in data stream analysis, and it has a large number of applications both in this form and for generalizations of this problem. Assume that the each item is an integer in the range $1 \ldots n$. Then we could maintain a vector $\boldsymbol{a}$ where $\boldsymbol{a}[i]$ counts the number of occurrences of item i. Arrivals of new items can be modeled as adding one to the appropriate entry in the vector. In the turnstile streaming model, departures can be modeled as subtracting one from the corresponding entry. The number of distinct items corresponds to the L_0 norm $\boldsymbol{a}$, that is, the number of non-zero counts. The L_0 norm is somewhat more general than this, since it can also incorporate negative counts. Such negative counts arise, for example, when we want to compare two vectors of counts, and find in how many places the counts differ (the Hamming difference). Note that stable distributions do not exist for $\alpha = 0$, so we cannot directly apply the sketching technique. Instead, we observe that for sufficiently small values of p, the L_p norm approximates the L_0 norm:

Theorem 2 (From [11]). *The L_0 norm $||\boldsymbol{a}||_0$ can be approximated by finding the L_p norm of the integer valued vector $\boldsymbol{a}$ for sufficiently small p ($0 < p \leq \frac{\epsilon}{\log U}$) provided we have an upper bound (U) on the size of each entry in the vector, so $\forall i : |\boldsymbol{a}[i]| < U$.*

Proof. We show that the L_0 norm of a vector can be well-approximated by $\sum_i |\boldsymbol{a}[i]|^p = ||\boldsymbol{a}||_p^p$ for a small value of p ($p > 0$). If, for all i we have that $|\boldsymbol{a}[i]| \leq U$ for some upper bound U, then

$$||\boldsymbol{a}||_0 = \sum_i |\boldsymbol{a}[i]|^0 \leq \sum_i |\boldsymbol{a}[i]|^p \leq \sum_i U^p |\boldsymbol{a}[i]|^0$$

$$\leq U^p \sum_i |\boldsymbol{a}[i]|^0 \leq (1+\epsilon) \sum_i |\boldsymbol{a}[i]|^0 = (1+\epsilon)||\boldsymbol{a}||_0$$

We use the fact that $\boldsymbol{a}[i]$ is an integer and $\forall i : |\boldsymbol{a}[i]| \leq U$. The last inequality uses $U^p \leq (1+\epsilon)$ which follows if we set $p \leq \ln(1+\epsilon)/\ln U \approx \frac{\epsilon}{\ln U}$.

From this, it follows that if we set the α of our sketches to be sufficiently small—as small as the value of p indicated by the above analysis—and compute sketches using stable distributions, then this can approximate the number of distinct items, and more generally the L_0 norm and L_0 difference between vectors. Since by definition the stable distributions capture the L_p norm, we have to take no special action when the vectors may contain negative values. Because the sketch is formed by a linear projection of random vectors with the input data, they naturally and smoothly accept updates of negative values.

When implementing this technique there are various technical details to deal with. Values drawn from stable distributions with small stability parameters α tend to grow very large, so even standard floating point formats are insufficient to handle them. However, in practice it usually suffices to set α to be a sufficiently small constant value. The experiments in [10] show that with $\alpha = 0.02$, good approximations to the L_0 norm and the number of distinct items can be found.

1.3.2 Dominance Norms

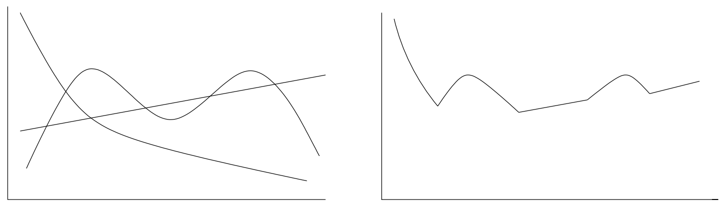

Fig. 1.3. The "dominance norm" of multiple signals (left) computes the "area under the curve" of the upper envelope of multiple signals (right).

The approach of using stable distributions to capture the L_0 norm has been applied to other problems: in [13], the so-called "dominance norm" of data is approximated using stable distributions. The dominance norm is defined as $\sum_i \max_j a_{i,j}$ for a sequence of data items of the form $(i, a_{i,j})$, intuitively giving the "worst-case influence" of a sequence of signal values. This definition is illustrated in Figure 1.3: for the three signals shown on the left, the dominance norm is computed by finding the upper envelope of the signals (shown on the right), and taking the area under this upper envelope.

One can approximate this computation by transforming the input into an instance of computing L_0 norms. Suppose that the signal values $a_{i,j}$ are integers. We can (conceptually) replace each $a_{i,j}$ with a sequence of distinct items, $a_{i,1}, a_{i,2} \ldots a_{i,j}$. Now observe that the number of distinct items in the transformed stream is exactly the dominance norm. This shows that L_0 is at the heart of the dominance norm. However, this approach is not scalable:

naively replacing $a_{i,j}$ from the input with $a_{i,j}$ items means that the algorithm is exponentially slow in the size of the input. Instead, we can make use of the properties of stable distributions to build an estimator whose distribution is correct. The key idea is to round each $a_{i,j}$ to the closest power of $(1+\epsilon)$, $(1+\epsilon)^i$, say, and to add i appropriately scaled values from a stable distribution to build a sketch with the right distribution [13].

1.3.3 Application to Computing Embeddings

Not all objects can be naturally modeled as vectors. In dealing with massive items that consist of text, geometric data, structured data or other objects, new methods are needed to compare and measure them. However, the α-stable sketches for L_1 and L_2 distance are sufficiently flexible that they allow the following "embedding approach". Consider any set of objects X, with a distance functions $D(q,r)$ defined for any $q,r \in X$.

Definition 4. *A mapping $f : X \to L_p$ is called an* embedding with distortion *c, if for any $q,r \in X$, we have*

$$D(q,r) \leq \|f(q) - f(r)\|_p \leq c \cdot D(q,r).$$

Here, we use L_p as shorthand for "a vector space with the vector L_p norm". This definition can be further extended to allow the inequalities to hold with certain probability.

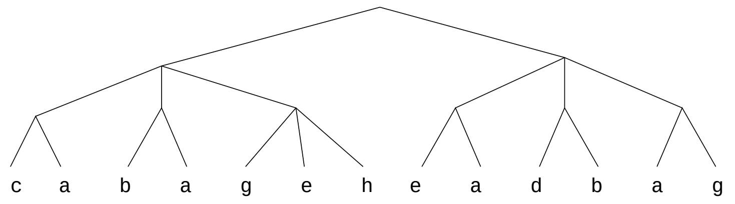

Fig. 1.4. Example parse tree for block edit distance text embedding

If the mapping f works for some $p \in \{0,1,2\}$, and if f can be computed in a streaming fashion, then we can obtain a streaming algorithm for computing short sketches of objects from the space X. That is, for any $q,r \in X$ defined by a stream, we can compute their sketches such that $D(q,r)$ can be approximated given the sketches. See [27, 33] for more on embeddings and their algorithmic applications.

The simplest example of this approach is given in [14], where it is shown that biologically motivated *distances on permutations* can be approximated up to small constant factors by encoding information about adjacent characters in the permutation as appropriate vectors in L_1.

More involved is the method in [12] which shows that a *distance between strings* can also be embedded into L_1. Only local information about the sequence is used in order to build the vector representation. The construction is more complex, since the sequence is parsed into small blocks, which in turn are re-parsed at successive levels in a hierarchy until a single item is left that represents the whole string. An example parsing of a string is shown in Figure 1.4: a tree is built whose leaves are the characters of the string, and whose internal nodes represent selected substrings. The parsing can be computed as successive characters are observed, and the increasingly long substrings given by the internal nodes can be represented compactly with hash values. These substrings can be thought of as defining dimensions of a high-dimensional vector space. In the paper, it is shown that the L_1 distance between two vectors created by this process approximates an editing distance between the corresponding strings. Since the parsing can be computed online, sketches for this distance can be computed in small space using the α-stable approach. In total, $O(\log n \log^* n)$ space is required to process a string of length n, and the embedding has distortion $O(\log n \log^* n)$.

This approach is extended from string based data to tree structures (such as XML documents) in [22]. Using a similar parsing approach, it is shown how an appropriate editing distance on trees can be approximated up to a factor of $O(\log^2 n \log^* n)$ for trees with at most n nodes. Further, with a different kind of sketch based on stable distributions, the join size of a set of trees can be approximated. Here, the join size is the number of pairs that are within a threshold distance of each other.

Another example of this approach is given in [28]. Consider a discrete d-dimensional space $\{1 \ldots \Delta\}^d$, and let P and Q be two subsets from that space. Define $M(P,Q)$ to be the cost of the *matching* between P and Q with minimum cost: the cost of the matching is given by the sum of the distances between the paired-up points. The value of $M(P,Q)$ is a natural measure of a difference between two sets of points. Building on the work of Charikar [8], Indyk [28] showed that $M(\cdot,\cdot)$ can be embedded into L_1 with distortion $O(\log \Delta)$, and that that embedding can be computed in small space. In fact, the embedding is quite simple. Let G_i, $i = 1 \ldots t = \log \Delta$, be square grids over $\Re^d$ with side length 2^{i-1}, shifted by a vector chosen uniformly at random from $[0, \Delta]^d$. For each cell c in G_i, let $n^i_P(c)$ be the number of points in P that fall into c; note that n^i_P can be viewed as a (high-dimensional) vector. The embedding f maps P into (essentially) a concatenation of vectors $2^0 n^0_P, 2^1 n^1_P, \ldots, 2^t n^t_P$. Observe that the embedding can be computed in a streaming fashion: adding a point p to P can be implemented by incrementing t positions in $f(P)$ that correspond to cells containing p; deleting a point from P can be implemented in an analogous way. Thus, the embedding can be naturally combined with the sketching algorithm from the previous section.

It is worth mentioning that the above approximation factor $O(\log \Delta)$ cannot be much improved if one insists on proceeding through the L_1 norm. Specifically, Naor and Schechtman [36] showed that any such embedding must incur $\Omega(\log \Delta)$ distortion. This lower bound may be tight for any approximation, and it will be interesting to resolve this issue.

Finally, we mention that an analogous embedding into L_0 gives a streaming algorithm for estimating the cost of the minimum spanning tree of set of points P, up to a factor of $O(\log \Delta)$. See [28] for details.

1.3.4 Clustering and Nearest Neighbors

The sketch structure can be used as a "distance oracle", giving dependable approximations of the distance between high dimensional vectors while keeping only a constant amount of space for each object. They can therefore be applied to a number of data indexing and data mining questions which rely on such distance computations, replacing exact distance computations with approximations. For example, in order to perform clustering on a set of high dimensional vectors that are defined by data streams, we can keep sketches of the vectors, and then run the clustering algorithm using those sketches. This approach was investigated in [11], where experimental evidence was given that the clusterings found are of similar quality to those using exact distance measurements. The idea of replacing exact distance computations with approximate ones can be analyzed formally. For example, it is easy to show that for the k-center objective function that using approximate distances changes the approximation quality of the result from 2 to $2 + \epsilon$ [9].

A more involved approach was taken in [16]. This showed that sketches using stable distributions could be fitted into the framework of 'Locality Sensitive Hash Functions', and consequently can be used in the construction of Approximate Nearest Neighbor search structures. Although this more generally applies to non-streaming scenarios, the whole algorithm can be run on data presented in a streaming format. The space that is needed is a function of the number of data points, rather than a function of the total size of the input data.

1.4 Related Work

The sketch for L_2, which is formed as the inner product between the vector $\boldsymbol{a}$ and vectors r, each of whose entries is drawn independently from a Gaussian distribution, can be seen as a weaker version of the well-known Johnson-Lindenstrauss Lemma [30]. This states that such there exist embeddings of high dimensional vectors in Euclidean space into a space with dimension $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ which has distortion $1 + \epsilon$ with probability $1 - \delta$. Here, we have shown the result for a space where we use the median operator to compute the distance. It has been shown that by taking the appropriately scaled L_2 difference between such sketch vectors formed in the same way also has this property (see, for example, [29]). What we also have here is a version of this "weak Johnson-Lindenstrauss" lemma for L_1 and L_0. This is about as strong as we may hope for, since it has been shown that it is not possible to create an approximate distance preserving of L_1 into a lower dimensional L_1 space [4]. Here, we use an L_1-like operator: instead of computing the L_1 norm of the sketch as $\sum_j |sk[j]|$, we compute $\text{med}_j(|sk[j]|)$. [3]

The fundamental work of Alon, Matias and Szegedy [2] (described in an earlier chapter) initiated recent focus on computing norms of data streams. An algorithm given therein computes the second frequency moment of a data stream, F_2. As was observed in [18], this directly gives a solution to finding the L_2 norm and L_2 difference between streams in the turn-stile model. For most applications, the fact that updates can be performed very quickly, and that the necessary four-wise independent hash functions can be computed easily [39] means that this approach will be preferable in many situations.

For computing the L_1 difference, [18] shows how to modify the Alon-Matias-Szegedy method using carefully constructed range-summable random variables. However, this is under very strong restrictions on the data: each index can be seen at most once for each vector. The approach here allows a much more general model of the data, and is easier to compute. Similarly, [20] extended the above approach to arbitrary L_p norms for $p \in (0, 2)$, but with the same disadvantages. The main results described in this chapter on constructing sketches using stable distributions (Section 1.2) appeared first in [26].

The distinct elements problem has attracted a great deal of study. In the arrivals only (cash register) model, algorithms are known which are significantly faster than the approach described here. See [19, 24, 23, 3, 17], and the discussions in elsewhere in this book. In the more general problem of computing the L_0 norm and L_0 difference, where entries in the implicit vector defined by the stream can be negative, the method using stable distributions is the only published solution. A detailed empirical study of this approach, and a collection of ways to increase processing speed, are given in [10].

In terms of the application of α-stable sketches to speeding up clustering, see elsewhere in this book for details of much of the other work on clustering data streams. Typically the goal is typically to compute a representation of the optimal clustering of a very large number of points in some arbitrary metric space, when each point has a small representation. Here, we considered a somewhat different scenario, where the number of points to cluster is not too large, but each point is represented by a very high dimensional vector in some L_p normed space. Hence, the two approaches are in some sense complementary and are not directly comparable.

[3] Observe that since we need the median operator, this is not a normed space. This is an important restriction, since it means that one cannot immediately apply well-known techniques which work on specific normed spaces, such as clustering or similarity search. In constrast, since the Johnson-Lindenstrauss lemma does yield points in a lower dimensional metric space, all algorithms for Euclidean space can be applied to the resulting transformed data.

There is a very large body of work on Stable Distributions in Statistics and related areas. For pointers, see the books by Zolotarev [42, 40], and Nolan [38]. It is reasonable to say that the applications of stable distributions to streaming computations are far from exhausted.

1.5 Extensions and New Directions

Finally, we outline some potential areas for future research to extend the applications of stable distributions to streaming computations.

- One obstacle to implementing sketch-based summarization of very high speed data streams is that the time cost of maintaining sketches can be too expensive in some situations. This derives in part from the cost of simulating stable distributions using transforms from uniform distributions. The main cost comes from having to update every entry in the sketch with every update. For L_2 norms alternative methods are known which are asymptotically faster than $\Omega(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ per update; for example, see [7, 39] or the recent work on the "fast Johnson-Lindenstrauss transform" [1]. Likewise, for the problem of approximating the number of distinct items in the arrivals only (cash register) model, then faster updates are possible. It remains an open problem to design algorithms to compute L_1 norms and L_0 norms in the turn-stile model, which have lower per-item update cost. Note that one cannot expect lower *space* costs, since lower bounds of $\Omega(\frac{1}{\epsilon^2})$ have been shown [41].
- It is of interest to address the engineering question of how to incorporate stable sketch computations into high speed data stream systems [15]. Various precomputations may be possible to speed up the computations, using appropriate look-up tables and so on. Techniques such as fixed point arithmetic may also be appropriate for certain fixed L_p values ($p = 1$ or $p = 2$, say), where the generated values do not grow too large. Other approaches may take advantages of skew in the data to, for example, collect together multiple instances of the same vector entry being updated, to further speed up the update time. There is a need to study in detail many implementation issues such as these to make the use of stable sketches within real situations a practical reality.
- The flexibility of this approach means that it is inviting to consider whether there are similar methods to compute other quantities of interest on the stream. For example, the "empirical entropy" of a sequence, given by $\sum_i \boldsymbol{a}_i \log \boldsymbol{a}_i$ has a number of applications, as does the "sum of logs", $\sum_i \log \boldsymbol{a}_i$. Recently, progress has been made on computing the empirical entropy of counts of items in the stream [21, 5], it remains open to determine whether stable distributions or similar techniques can also be applied to these problems.

- It is an intriguing fact that stable distributions exist only in the range $\alpha \in (0, 2]$, which corresponds to the range of L_p norms that can be approximated efficiently (essentially in constant space) on the stream. Meanwhile, there are provable lower bounds on the space required to estimate L_p norms for $p > 2$ that are polynomial in n, the dimension of the vector. The connection between these facts may be more than mere coincidence, and making this connection explicit could lead to the development of stronger lower bounds, or lower bounds for other, related problems.
- Many techniques using stable distributions make use of a natural range summability-like property of these distributions. That is, their defining feature is that the sum of stable distributions is itself distributed stable. This results in careful constructions of random variables such that the range sum of particular sub-ranges of variables can be computed efficiently (exponentially more efficient than directly computing the sum). Such constructions have been shown for $\alpha = 1$ and $\alpha = 2$ [25]. It remains to generalize these techniques to general values of α, and to show new applications.
- Finally, there is a large literature on stable distributions, resulting from their study in statistics, economics and beyond. Applications of stable distributions to streaming computations have only just begun to make use of the wealth of existing knowledge about these distributions, and it is very conceivable that there are many other applications of these distributions to problems of practical interest in streaming computations.

References

1. N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Proceedings of the ACM Symposium on Theory of Computing*, 2006.
2. N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 20–29, 1996. Journal version in *Journal of Computer and System Sciences*, 58:137–147, 1999.
3. Z. Bar-Yossef, T.S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisian. Counting distinct elements in a data stream. In *Proceedings of RANDOM 2002*, pages 1–10, 2002.
4. B. Brinkman and M. Charikar. On the impossibility of dimensionality reduction in L_1. In *IEEE Conference on Foundations of Computer Science*, pages 514–523, 2003.
5. A. Chakrabarti, G. Cormode, and A. McGregor. A near-optimal algorithm for computing the entropy of a stream. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, 2007.
6. J.M. Chambers, C.L. Mallows, and B.W. Stuck. A method for simulating stable random variables. *Journal of the American Statistical Association*, 71(354):340–344, 1976.

7. M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Procedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 693–703, 2002.
8. M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 380–388, 2002.
9. G. Cormode. *Sequence Distance Embeddings*. PhD thesis, University of Warwick, 2003.
10. G. Cormode, M. Datar, P. Indyk, and S. Muthukrishnan. Comparing data streams using Hamming norms. In *Proceedings of the International Conference on Very Large Data Bases*, pages 335–345, 2002. Journal version in *IEEE Transactions on Knowledge and Data Engineering* 15(3):529–541, 2003.
11. G. Cormode, P. Indyk, N. Koudas, and S. Muthukrishnan. Fast mining of tabular data via approximate distance computations. In *Proceedings of the International Conference on Data Engineering*, pages 605–616, 2002.
12. G. Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pages 667–676, 2002.
13. G. Cormode and S. Muthukrishnan. Estimating dominance norms of multiple data streams. In *Proceedings of the European Symposium on Algorithms (ESA)*, volume 2838 of *LNCS*, 2003.
14. G. Cormode, S Muthukrishnan, and S. C. Ṣahinalp. Permutation editing and matching via embeddings. In *Proceedings of 28th International Colloquium on Automata, Languages and Programming*, volume 2076, pages 481–492, 2001.
15. C. Cranor, T. Johnson, O. Spatscheck, and V. Shkapenyuk. Gigascope: A stream database for network applications. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 647–651, 2003.
16. M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Symposium on Computational Geometry*, 2004.
17. C. Estan, G. Varghese, and M. Fisk. Bitmap algorithms for counting active flows on high speed links. In *Proceedings of the Internet Measurement Conference*, pages 153–166, 2003.
18. J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate L_1-difference algorithm for massive data streams. In *IEEE Conference on Foundations of Computer Science*, pages 501–511, 1999.
19. P. Flajolet and G. N. Martin. Probabilistic counting. In *IEEE Conference on Foundations of Computer Science*, pages 76–82, 1983. Journal version in *Journal of Computer and System Sciences*, 31:182–209, 1985.
20. J. Fong and M. Strauss. An approximate L_p-difference algorithm for massive data streams. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 193–204, 2000.
21. S. Ganguly and B. Lakshminath. Estimating entropy over data streams. In *Proceedings of the European Symposium on Algorithms (ESA)*, 2006.
22. M. Garofalakis and A. Kumar. Correlating XML data streams using tree-edit distance embeddings. In *Proceedings of ACM Principles of Database Systems*, pages 143–154, 2003.
23. P. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *Proceedings of the International Conference on Very Large Data Bases*, pages 541–550, 2001.

24. P. Gibbons and S. Tirthapura. Estimating simple functions on the union of data streams. In *ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 281–290, 2001.
25. A. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 389–398, 2002.
26. P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *IEEE Conference on Foundations of Computer Science*, pages 189–197, 2000.
27. P. Indyk. Algorithmic aspects of geometric embeddings (invited tutorial). In *IEEE Conference on Foundations of Computer Science*, pages 10–35, 2001.
28. P. Indyk. Algorithms for dynamic geometric problems over data streams. In *Proceedings of the ACM Symposium on Theory of Computing*, 2004.
29. P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 604–613, 1998.
30. W.B. Johnson and J. Lindenstrauss. Extensions of Lipshitz mapping into Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.
31. P. Li. Very sparse stable random projections, estimators and tail bounds for stable random projections. Technical Report cs.DS/0611114, arxiv, 2006.
32. P. Li, T. Hastie, and K. W. Church. Nonlinear estimators and tail bounds for dimension reduction in L_1 using cauchy random projections. *Journal of Machine Learning Research (JMLR)*, 2007.
33. J. Matoušek. *Lectures on Discrete Geometry.* Springer, 2002.
34. R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.
35. S. Muthukrishnan. Data streams: Algorithms and applications. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, 2003.
36. A. Naor and G. Schechtman. Planar earthmover is not in L_1. In *IEEE Conference on Foundations of Computer Science*, 2006.
37. N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12:449–461, 1992.
38. J. Nolan. Stable distributions. Available from `http://academic2.american.edu/~jpnolan/stable/chap1.ps`.
39. M. Thorup and Y. Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pages 615–624, 2004.
40. V. V. Uchaikin and V. M. Zolotarev. *Chance and Stability: Stable Distributions and their applications.* VSP, 1999.
41. D. Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pages 167–175, 2004.
42. V. M. Zolotarev. *One Dimensional Stable Distributions*, volume 65 of *Translations of Mathematical Monographs.* American Mathematical Society, 1983.