CS5234 Algorithms at Scale

Mini Project Phase 2 Report
Identifying Correlation in Stream of Samples

Gu Zhenhao # A0236600R, Zhang Hao # A0210996X

NUS National University of Singapore

School of Computing

To avoid confusion, we summarize the notation used in the report as follows:

Notation	Meaning
X, Y	Two random variables whose correlation we are trying to identify
n	Range of X and Y should be $[1, n]$.
p_i	Probability of X taking value i, $\Pr[X = i]$
q_j	Probability of Y taking value j, $\Pr[Y = j]$
$s_{i,j}$	Probability of (i, j) appearing in the stream, $\Pr[X = i, Y = j]$
$r_{i,j}$	Product of marginal probabilities, $\Pr[X = i] \Pr[Y = j]$
$\Delta_{i,j}$	Difference between joint probability and product probability, $s_{i,j} - r_{i,j}$
A	Size of counter matrix
$\hat{p}, \hat{q}, \hat{s}, \hat{\Delta}$	Estimator of p, q, s, Δ calculated by the full $n \times n$ counter matrix
$\tilde{p}, \tilde{q}, \tilde{s}, \tilde{\Delta}$	Estimator of $\hat{p}, \hat{q}, \hat{s}, \hat{\Delta}$ calculated by the truncated $A \times A$ counter matrix
$n_{i,j}$	The number of time each pair (i, j) appears in the stream
$C_{i,j}$	The value in the counter matrix on the i^{th} row and j^{th} column
$\sum C_{i,\cdot}, \sum C_{\cdot,j}$	The sum of values in the i^{th} row, j^{th} column in C, respectively
N	Length of the stream

Table 1: Notations

1 **Purpose** (same as Phase 1 Report)

In this mini-project, we try to check whether two (or more) random variables are independent based on their samples in a data stream. More specifically, given a stream of pairs (i, j), where $i, j \in [n]^2$, and (i, j) follows a joint distribution (X, Y), we want to check if X and Y are independent, i.e. $\Pr[X = i, Y = j] = \Pr[X = i] \Pr[Y = j]$ for all $i, j \in [n]^2$.

This is a fundamental problem with a lot of applications in network monitoring, sensor networks, economic/stock market analysis, and communication applications, etc.

For simplicity, we define matrix r and s where $r_{ij} = \Pr[X = i, Y = j]$ and $s_{ij} = \Pr[X = i] \Pr[Y = j]$. Now, to measure how close X and Y are to independence, we have three metrics [3]:

1. ℓ_1 **difference**: The sum of absolute difference of the joint distribution and the product distribu-

tion, similar to the ℓ_1-norm of vectors,

$$|\Delta| = |r - s| = \sum_{i,j} |r_{ij} - s_{ij}|$$

2. ℓ_2 **difference**: The sum of squared difference of the joint distribution and the product distribution, taken square root, similar to the ℓ_2-norm of vectors,

$$\|\Delta\| = \|r - s\| = \sqrt{\sum_{i,j} (r_{ij} - s_{ij})^2}$$

3. **Mutual information**: defined by $I(X;Y) = H(X) - H(X \mid Y)$, where $H(X) = -\sum_{i=1}^{n} \Pr[X = 1] \log \Pr[X = i]$ is the *entropy* of distribution X and

$$H(X \mid Y) = \sum_{i,j} \Pr[X = i, Y = j] \log \frac{\Pr[Y = j]}{\Pr[X = i, Y = j]}$$

is the *conditional entropy*.

All metrics should be zero if X and Y are independent, and further away from 0 if X and Y are further away from independence (higher correlation). Our goal then reduces to **give an estimation of the value of these metrics, and check if it is small enough to conclude independence between X and Y**.

2 Current Progress and New Information

2.1 Implementation of Sketching of Sketches

We tried to implement[1] the Sketching of sketches algorithm to estimate the ℓ_1 and ℓ_2 difference as proposed by Indyk and McGregor [3].

2.1.1 ℓ_2 Difference

The paper proposed an algorithm that uses the result that the weighted sum of 4-wise independent vectors $x, y \in \{-1, 1\}^n$ and any $n \times 2$ matrix v could be used to estimate the norm of v [1]. More specifically,

Lemma 1. *Consider $x, y \in \{-1, 1\}^n$ where each vector is 4-wise independent. Let $v \in \mathbb{R}^{n^2}$ and $z_i = x_{i_1} y_{i_2}$. Define $\Upsilon = (\sum_{i \in [n]^2} z_i v_i)^2$. Then $\mathrm{E}[\Upsilon] = \sum_{i \in [n]^2} v_i^2$ and $\mathrm{Var}[\Upsilon] \leq 3(\mathrm{E}[\Upsilon])^2$.*

[1] Stage of python code: https://github.com/GZHoffie/CS5234-mini-project/blob/master/mini_project/algorithms/sketching_sketches.py.

for the following algorithm,

Algorithm 1: ℓ_2 difference, $\|r - s\|$ Approximation

Input : The error bound ϵ
Output: An estimate of number of minors $|M|$

$x, y \leftarrow$ random vector of length n containing 1 or -1; // 4-wise independent vectors
$t_1, t_2, t_3 \leftarrow 0$;
for *each pair* (i, j) *in stream* **do**
    $t_1 \leftarrow t_1 + x_i y_j$; // sums to $m \sum_{i \in [n]^2} z_i r_i$
    $t_2 \leftarrow t_1 + x_i$;
    $t_3 \leftarrow t_1 + y_j$; // t_2 * t_3 sums to $m^2 \sum_{i \in [n]^2} z_i s_i$
$\Upsilon \leftarrow (t_1/m - t_2 t_3/m^2)^2$; // expectation being $\|r - s\|$
return Υ;

The algorithm looks generally easy, as we only need to compute three numbers for each run. However, when it comes to implementation, the drawback of the algorithm appear.

1. It is actually **memory-inefficient**, since for each run, we need to generate random 4-wise independent vectors x, y and store it until we read the whole stream. And since we need to run $\mathcal{O}(\epsilon^{-2} \log \delta^{-1})$ copies of algorithm 1, storing those vectors will cost $\mathcal{O}(\epsilon^{-2} n \log \delta^{-1})$ space, which is disastrous when n goes large.

2. We can solve the above problem by choosing a large p and use a polynomial of degree 3,

$$h(x) = ((h_3 x^3 + h_2 x^2 + h_1 x + h_0) \mod p \mod 2) \times 2 - 1$$

   where the coefficients $h_0, h_1, h_2, h_3 < p$ are randomly chosen, to map x to either -1 or 1, creating a 4-wise independent vector [4].

   This implementation is good since we don't need to care about n anymore, and we only need to store 8 numbers (4 coefficients h_0, h_1, h_2, h_3 for each of x and y) for each run instead of the two vectors of length n before. However, calculating this hash function is **computationally expensive**, making the algorithm slow. And although we get rid of the n factor in space complexity, the constant factor is at least 8, so again a lot of space is used.

In the end, we were able to verify the correctness of algorithm. With 100 groups of 100 runs of algorithm 1, we are able to obtain a multiplicative error of only 0.0067 with a randomly generated dataset (described in section 2.3) with $n = 1,000$ and $N = 100,000$.

2.1.2 ℓ_1 Difference

We haven't entirely completed the implementation of the ℓ_1 difference estimation. But again, the algorithm will take a linear space since we will need to generate and store

- A vector x of length n where each element follows a Cauchy distribution, and
- a vector y of length n where each element follows a T-truncated-Cauchy distribution [2]

for each $\mathcal{O}(\log \delta^{-1})$ runs of experiment. This time, we cannot use a hash function anymore, so the n factor would be inevitable for a single-pass algorithm, making it very memory-inefficient.

2.2 Analysis and Implementation of Counter Matrix Algorithm

In light of the problems we encountered for sketching of sketches algorithm, we want to see if a new algorithm that is based on the χ^2-test of independence and the counter matrix algorithm (from problem set 5) would be a good substitute.

We can prove that our resulting algorithm

- is correct if X and Y are actually independent (theorem 2),
- can give a lower bound for ℓ_1 difference (theorem 3), and
- can give an unbiased estimate for ℓ_2 difference (theorem 6).

2.2.1 Preliminaries

Given two discrete random variables X and Y, we say that they are independent if and only if for all i, j,

$$\underbrace{\Pr[X = i, Y = j]}_{=s_{i,j}} = \underbrace{\Pr[X = i]}_{=p_i} \underbrace{\Pr[Y = j]}_{=q_j}$$

or $s_{i,j} = p_i q_j$. This means that ideally, $\Delta_{i,j} = s_{i,j} - p_i q_j = 0$ for all i, j and the ℓ_1 difference $|\Delta| = 0$, ℓ_2 difference $\|\Delta\| = 0$. If we have enough space, we can store an $n \times n$ counter matrix,

	1	2	...	j	...	$n-1$	n	$\sum$
1								
2								
$\vdots$								
i				$n_{i,j}$				$\sum n_{i,\cdot}$
$\vdots$								
$n-1$								
n								
$\sum$				$\sum n_{\cdot,j}$				N

Table 2: $n \times n$ counter matrix

where each counter $n_{i,j}$ stores the number of times the pair (i, j) appears in the stream. We can estimate the joint probability $\hat{s}_{i,j} = n_{i,j}/N$. The sum of each row and each column are calculated to estimate the marginal probabilities,

$$\hat{p}_i = \frac{\sum n_{i,\cdot}}{N}, \qquad \hat{q}_j = \frac{\sum n_{\cdot,j}}{N}$$

Ideally, if X and Y are independent, we should have

$$\hat{\Delta}_{i,j} := \hat{s}_{i,j} - \hat{p}_i \hat{q}_j = 0$$

for all i, j. The ℓ_1 difference can be estimated by

$$|\widehat{s-r}| = |\hat{\Delta}| = \sum_{i,j} |\hat{s}_{i,j} - \hat{p}_i \hat{q}_j| \tag{1}$$

and the ℓ_2 difference can be estimated by

$$\|\widehat{s-r}\| = \|\hat{\Delta}\| = \sqrt{\sum_{i,j} (\hat{s}_{i,j} - \hat{p}_i \hat{q}_j)^2} \tag{2}$$

We can also calculate the χ^2 statistics for independence test,

$$X^2_{(n-1)^2} = \sum_{i,j} \frac{(\hat{s}_{i,j} - \hat{p}_i \hat{q}_j)^2}{\hat{p}_i \hat{q}_j} \tag{3}$$

follows a chi-squared distribution with degree of freedom $(n-1)^2$. This statistics can be compared with the critical value $\chi^2_{0.05,(n-1)^2}$ to see if we can reject a null hypothesis: X and Y are independent with $p < 0.05$ [5].

2.2.2 Method

The counter matrix above is nice, but it will cost $\mathcal{O}(n^2)$ space, which is bad. Our idea is **to use an $A \times A$ ($A \ll n$) matrix to store the information in $n \times n$ counter matrix by randomly summing rows and columns together**.

More specifically, we will choose two random hash functions $h_1, h_2 : [n] \to [A]$, $x \mapsto i$ where $1 \le i \le A$. Upon seeing a pair (x, y) in the stream, we increment the counter $C_{h_1(x),h_2(y)}$.

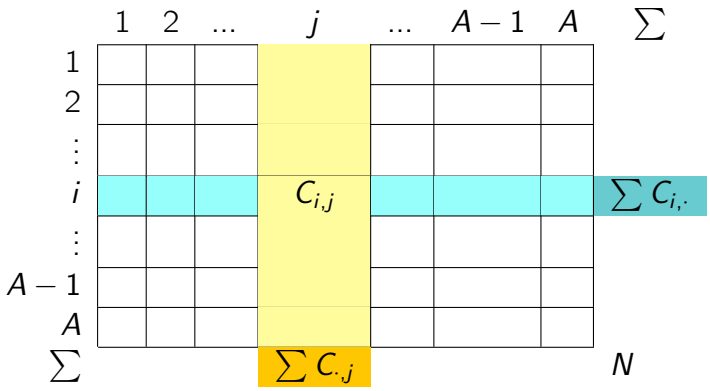

Table 3: $A \times A$ Counter Matrix

After reading the stream we can calculate the estimators for joint and marginal probabilities,

$$\tilde{s}_{i,j} = \frac{C_{i,j}}{N}, \qquad \tilde{p}_i = \frac{\sum C_{i,\cdot}}{N}, \qquad \tilde{q}_j = \frac{\sum C_{\cdot,j}}{N}$$

and use them to estimate the ℓ_1, ℓ_2 and χ^2 statistics using a similar fashion as equation 1, 2 and 3.

2.2.3 Analysis

We want to prove the correctness of our method. First we show that **the estimator is correct when X and Y are actually independent**.

Theorem 2 (Independent Case)**.** *If X and Y are indeed independent, with sufficiently large N, we will have the estimated ℓ_1 and ℓ_2 difference be 0, i.e. $|\tilde{\Delta}| = \|\tilde{\Delta}\| = 0$, where $\tilde{\Delta}_{i,j} = \tilde{s}_{i,j} - \tilde{p}_i\tilde{q}_j$.*

Proof. Let us denote $h_1^{-1}(i) := \{x : h_1(x) = i\}$ the set of all x that maps to i, and similarly $h_2^{-1}(j) := \{y : h_2(y) = j\}$. Let $h^{-1}(i,j) := h_1^{-1}(i) \times h_2^{-1}(j)$ be all the pairs (x,y) that maps to (i,j). Then for each $(i,j) \in [A]^2$, we will have the value of the counter be

$$C_{i,j} = \sum_{(x,y)\in h^{-1}(i,j)} n_{x,y}$$

and the sum of the corresponding row and column be,

$$\sum C_{i,\cdot} = \sum_{x\in h_1^{-1}(i)} n_{x,\cdot}, \qquad \sum C_{\cdot,j} = \sum_{y\in h_2^{-1}(y)} n_{\cdot,y}$$

Therefore we will have $\tilde{p}_i\tilde{q}_j = \sum_{(x,y)\in h^{-1}(i,j)} n_{x,\cdot}n_{\cdot,y}/N^2$. Now, for sufficiently large N, we will have $n_{x,y} \to Ns_{x,y}$, $\sum n_{x,\cdot} \to Np_x$ and $\sum n_{\cdot,y} \to Nq_y$. Therefore,

$$\tilde{\Delta}_{i,j} = \tilde{s}_{i,j} - \tilde{p}_i\tilde{q}_j = \sum_{(x,y)\in h^{-1}(i,j)} \frac{n_{x,y}}{N} - \frac{n_{x,\cdot}n_{\cdot,y}}{N^2} \to \sum_{(x,y)\in h^{-1}(i,j)} \underbrace{s_{x,y} - p_xq_y}_{=0} = 0$$

for all i,j. Therefore the ℓ_1 and ℓ_2 norm will be 0. □

If X and Y are correlated, however, things will get complicated.

1. For ℓ_1 difference case, we claim that **the resulting estimate of ℓ_1 difference given by the $A \times A$ matrix will be a lower bound for the full $n \times n$ counter matrix**.

   Theorem 3 (Lower Bound for ℓ_1 Difference)**.** *We must have* $|\tilde{\Delta}| \leq |\hat{\Delta}| = \sum_{i,j\in[n]} |\hat{s}_{i,j} - \hat{p}_i\hat{q}_j|$.

   Proof. Using triangle inequality, we can see that for each $i, j \in [A]$,

   $$|\tilde{\Delta}_{i,j}| = \left| \sum_{(x,y)\in h^{-1}(i,j)} \hat{s}_{x,y} - \hat{p}_x\hat{q}_j \right| \leq \sum_{(x,y)\in h^{-1}(i,j)} \left|\hat{s}_{x,y} - \hat{p}_x\hat{q}_j\right|$$

   The term on the left sums up to $|\tilde{\Delta}|$, while the term on the right sums up to $|\hat{\Delta}|$. Thus we can prove the result. □

   This theorem indicate that we may want to create multiple $A \times A$ counter matrices and take the maximum of our results as an estimate for ℓ_1 difference. The variance of this estimate have not yet been studied.

2. For ℓ_2 difference case, let us consider the ℓ_2 difference squared, i.e.

   $$\|\hat{\Delta}\|^2 = \sum_{i,j\in[n]} (\hat{\Delta}_{i,j})^2 = \sum_{i,j\in[n]} (\hat{s}_{i,j} - \hat{p}_i\hat{q}_j)^2$$

   In the $A \times A$ counter matrix case, some of the $\hat{s}_{i,j} - \hat{p}_i\hat{q}_j$ will be added together then squared,

   $$\begin{aligned} \|\tilde{\Delta}\|^2 &= \sum_{i,j\in[A]} (\tilde{s}_{i,j} - \tilde{p}_i\tilde{q}_j)^2 = \sum_{i,j\in[A]} \left(\sum_{(x,y)\in h^{-1}(i,j)} \hat{s}_{x,y} - \hat{p}_x\hat{q}_j \right)^2 = \sum_{i,j\in[A]} \left(\sum_{(x,y)\in h^{-1}(i,j)} \hat{\Delta}_{x,y} \right)^2 \\ &= \underbrace{\sum_{i,j\in[n]} (\hat{\Delta}_{i,j})^2}_{=\|\hat{\Delta}\|^2} + \underbrace{\sum_{\substack{h(i,j)=h(k,l) \\ (i,j)\neq(k,l)}} \hat{\Delta}_{i,j}\hat{\Delta}_{k,l}}_{=:X} = \|\hat{\Delta}\|^2 + X \end{aligned}$$

   And we can see that the ℓ_2 estimate given by our $A \times A$ matrix can be expressed by $\|\hat{\Delta}\|^2$ plus a term X, where X is just the sum of product of $\hat{\Delta}$ of all the colliding pairs in our hash function. We thus want to study the distribution of X.

   Before going into this, we want to show a property of $\hat{\Delta}$.

   Lemma 4 (Property of $\hat{\Delta}$)**.** *The sum of each row or each column of* $\hat{\Delta}$ *must be 0.*

   Proof. For row i, for example, the sum of all elements on row i would be

   $$\begin{aligned} \sum \hat{\Delta}_{i,\cdot} &= \sum_{y\in[A]} \sum_{x\in[A]} (\hat{s}_{x,y} - \hat{p}_x\hat{q}_y) = \sum_{x\in[A]} \underbrace{\sum_{y\in[A]} \hat{s}_{x,y}}_{=\hat{p}_x} - \sum_{x\in[A]} \hat{p}_x \underbrace{\sum_{y\in[A]} \hat{q}_y}_{=1} \\ &= \sum_{x\in[A]} \hat{p}_x - \sum_{x\in[A]} \hat{p}_x = 0 \end{aligned}$$

The column case can be proved by a similar fashion. □

The following result immediately follows from the above lemma.

Corollary 5. *All elements in $\hat{\Delta}$ sums to 0.*

Now, let us investigate the expectation of X. The probability of the term $\hat{\Delta}_{i,j}\hat{\Delta}_{k,l}$ appear in X is equal to the probability of (k,l) having the same hash values as (i,j) (collide).

$$\Pr[h(k,l) = h(i,j)] = \begin{cases} 1/A^2 & \text{if } i \neq k, j \neq l \\ 1/A & \text{if} i = k \text{ or } j = l \end{cases}$$

since if $i = k$ or $j = l$, we only need to ensure the other dimension to have the same hash values, while if $i \neq k, j \neq l$, we need both pairs to have the same hash values.

	1	2	...	j	...	$n-1$	n
1							
2		$\frac{1}{A^2}$		$\frac{1}{A}$		$\frac{1}{A^2}$	
⋮							
i		$\frac{1}{A}$				$\frac{1}{A}$	
⋮							
$n-1$		$\frac{1}{A^2}$		$\frac{1}{A}$		$\frac{1}{A^2}$	
n							

Table 4: Illustration of probability that a pair collides with (has the same hash values as) (i,j).

Using lemma 4 and corollary 5, the expectation of the sum of all terms containing $\hat{\Delta}_{i,j}$ in X is then

$$\begin{aligned} & \hat{\Delta}_{i,j}\left[\frac{1}{A^2}\underbrace{\sum_{(i,j)\neq(k,l)}\hat{\Delta}_{k,l}}_{=-\hat{\Delta}_{i,j}} + \left(\frac{1}{A}-\frac{1}{A^2}\right)\underbrace{\sum_{i\neq k}\hat{\Delta}_{k,j}}_{=-\hat{\Delta}_{i,j}} + \left(\frac{1}{A}-\frac{1}{A^2}\right)\underbrace{\sum_{j\neq l}\hat{\Delta}_{i,l}}_{=-\hat{\Delta}_{i,j}}\right] \\ = & -\left(\frac{2}{A}-\frac{1}{A^2}\right)\hat{\Delta}_{i,j}^2 \end{aligned}$$

Therefore, the expectation of X is then

$$\mathrm{E}[X] = -\left(\frac{2}{A}-\frac{1}{A^2}\right)\sum_{i,j\in[n]}\hat{\Delta}_{i,j}^2 = -\left(\frac{2}{A}-\frac{1}{A^2}\right)\|\hat{\Delta}\|^2$$

Meaning that

$$\mathrm{E}[\|\tilde{\Delta}\|^2] = \left(1-\frac{2}{A}+\frac{1}{A^2}\right)\|\hat{\Delta}\|^2 = \left(1-\frac{1}{A}\right)^2\|\hat{\Delta}\|^2$$

This gives rise to our theorem,

Theorem 6 (Unbiased Estimator)**.** $\|\tilde{\Delta}\|^2/(1-1/A)^2$ *is going to be an unbiased estimator for* $\|\hat{\Delta}\|^2$*.*

This is a neat result, as it is not dependent on n or N.

This theorem indicates that we can, again, create multiple $A \times A$ counter matrices, divide the resulting squared ℓ_2 difference with $(1-1/A)^2$, and take the mean to get an unbiased estimation. Again, the variance of this estimate is yet to be studied.

Remark. *We should have* $A > 1$ *for this algorithm to be meaningful. If we set* $A = 1$*, the expectation* $\mathsf{E}[\|\tilde{\Delta}\|^2] = 0$*. In fact, if we only use 1 number, then* $s_{1,1} = p_1 = q_1 = 1$*, so the resulting norm will always be zero.*

2.2.4 Implementation and Results

For now, we have completed the implementation of this algorithm estimating the ℓ_1 and ℓ_2 difference[2]. For both implementation, we choose the hash functions to be

$$h_1(x) = (ax + b) \mod p \mod A, \qquad h_2(x) = (cx + d) \mod p \mod A$$

where p is a random prime that is significantly larger than A and $a, b, c, d < p$ are randomly chosen. Since we only need to compute two hash functions and store the 4 parameters for each counter matrix, our algorithm is quite fast and memory-saving.

1. For ℓ_1 difference estimator, we tried to have multiple counter matrices and return the largest $|\tilde{s} - \tilde{p}\tilde{q}|$. The result is indeed always smaller than the true ℓ_1 difference, but the error seems to mostly dependent on A. If A is large ($A \approx n$), then the error would be small. If $A \ll n$, then no matter how many counter matrices we run, the error will remain large.

2. For ℓ_2 difference estimator, we also run B counter matrices, and calculate the mean of $\|\tilde{\Delta}\|^2/(1-1/A)^2$. We have verified that the algorithm work and the $/(1-1/A)^2$ term indeed make the error smaller.

   With a random dataset with $n = 1,000$ **and** $N = 100,000$**, we are able to achieve a multiplicative error of only 0.00213 with a hundred** 10×10 **counter matrices, which is better than the sketching of sketches algorithm (0.0067).**

2.3 Experiments with Simulated Data

2.3.1 Data Set Generation

We need to generate two types of distribution for sample stream:

[2]Python code for Counter Matrix Algorithm: https://github.com/GZHoffie/CS5234-mini-project/blob/master/mini_project/algorithms/counter_matrix.py

1. X **and** Y **are independent**: To make sure every element pairs in the stream that satisfies $\forall i, j \in [n]^2 : \Pr[X = i, Y = j] = \Pr[X = i] \cdot Pr[Y = j]$, we first randomly generate 2 marginal probability arrays p_x and q_y of size n with each array sums to 1. Then we generated each element pairs using the probability arrays obtained.
2. X **and** Y **are correlated**: Generate the probability matrix $s_{x,y}$ with shape $n \times n$ and normalize it such that all values sums up to 1. Then we generated each element pairs using the probability matrix obtained.

   We first generate x based on the marginal probability p_x, then generate y based on the x^{th} row of the probability matrix, which represents $\Pr[y \mid x]$.

Our script[3] is designed to allow the user to set the following parameters to meet different experiment requirements:

1. `N`: Length of stream.
2. `n`: Maximum value of each element in X and Y.
3. `independent`: Specify whether X and Y is independent or not.

We also use a $n \times n$ counter matrix to calculate the values for $|\hat{\Delta}|$, $\|\hat{\Delta}\|$ and the χ^2 statistics, and store them for comparison.

2.3.2 Metrics

We feed the 2 implemented estimator using the same generated data set to get estimated results, then we can compare the estimated result with the exact solution. In particular, we used the multiplicative error,

$$\epsilon = \left| 1 - \frac{\Upsilon - \|r - s\|}{\|r - s\|} \right|$$

to check how far our estimator Υ is from the actual ℓ_2 difference.

3 Pending Tasks

1. Test the algorithms with various parameters, check if it agrees with theoretical analysis, and study the space-performance trade off.
2. Compare the performance of two algorithms given same space.
3. Study the variance in the ℓ_1 or ℓ_2 estimation and propose a good choice of parameters.
4. Explore and find possible solution to approximate non-discrete data stream using the 2 algorithms.

[3]Python script for dataset generation: https://github.com/GZHoffie/CS5234-mini-project/blob/master/mini_project/data/generate_dataset.py

References

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

[2] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006.

[3] Piotr Indyk and Andrew McGregor. Declaring independence via the sketching of sketches. pages 737–745, 01 2008.

[4] Mark N. Wegman and J.Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265–279, 1981.

[5] Minhaz Fahim Zibran. Chi-squared test of independence. *Department of Computer Science, University of Calgary, Alberta, Canada*, pages 1–7, 2007.