CS5234 Algorithms at Scale

Mini Project Proposal
Identifying Correlation in Data Stream

Gu Zhenhao # A0236600R, Zhang Hao # A0210996X

National University of Singapore
School of Computing

1 Purpose and Related Work

1.1 Objective

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 distribution, similar to the ℓ_1-norm of vectors,

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

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,

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

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.**

1.2 Related Work and Main Technical Ideas

This paper [3] proposes an estimation of all of the above 3 metrics.

1.2.1 ℓ_2 Difference Approximation

For now, we mainly focus on the estimation for ℓ_2 difference, since this is the only given approximation in the paper in single pass with sub-linear space.

1. **Problem**: Estimate the ℓ_2 difference using sub-linear space.
2. **Idea**: Utilize 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 $z_{\mathbf{i}} = x_{i_1} y_{i_2}$. Define $\Upsilon = (\sum_{\mathbf{i} \in [n]^2} z_{\mathbf{i}} v_{\mathbf{i}})^2$. Then $\mathsf{E}[\Upsilon] = \sum_{\mathbf{i} \in [n]^2} v_{\mathbf{i}}^2$ and $\mathsf{Var}[\Upsilon] \leq 3(\mathsf{E}[\Upsilon])^2$.*

   To prove this lemma, we can utilize the idea that z is 2-wise independent to calculate $\mathsf{E}[\Upsilon]$ and $\mathsf{E}[\Upsilon^2]$ respectively, thus concluding that $\mathsf{Var}[\Upsilon] \leq \mathsf{E}[\Upsilon^2] \leq 3(\mathsf{E}[\Upsilon])^2$.

   Using this lemma, we can randomly generate vectors of $x, y \in \{-1, 1\}^n$, and let v in the lemma be $r - s$, and calculate Υ base on the data in the stream. Υ is then an unbiased estimator for the ℓ_2 difference, and the variance is also bounded.

3. **Algorithm**: Using lemma 1, we can use the following algorithm, which is very simple,

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 $m \sum_{\mathbf{i} \in [n]^2} z_{\mathbf{i}} r_{\mathbf{i}}$
    $t_2 \leftarrow t_1 + x_i$;
    $t_3 \leftarrow t_1 + y_j$; // t_2 * t_3 sums to $m^2 \sum_{\mathbf{i} \in [n]^2} z_{\mathbf{i}} s_{\mathbf{i}}$
$\Upsilon \leftarrow (t_1/m - t_2 t_3/m^2)^2$; // expectation being $\|r - s\|$
return Υ;

Intuitively speaking, for a specific pair of (i, j) in the stream, if (i, j) appear more (or less) frequently than the product of frequency of i and frequency of j, then t_1 is supposed to be further away from $t_2 t_3$, making Υ large.

To reduce error, the above algorithm is run $\mathcal{O}(\epsilon^{-2}\log\delta^{-1})$ times, and the returning Υ is grouped into $\mathcal{O}(\log\delta^{-1})$ of $\mathcal{O}(\epsilon^{-2})$ results. The median of mean of each group is then returned. This is similar to the FM++ algorithm we see in the lecture.

4. **Results**: The algorithm gives a $(1+\epsilon,\delta)$ approximation,

   Theorem 2 (Multiplicative error)**.** *This algorithm provides a single-pass, $\mathcal{O}(\epsilon^{-2}\log\delta^{-1})$-space approximation so that the returned result is within a $1\pm\epsilon$ multiplicative bound of $\|r-s\|$, with probability at least $1-\delta$.*

   This can be proved by the lemma 1 along with the Chebychev's inequality.

1.2.2 ℓ_1 Difference Approximation

The strategy to estimate the ℓ_1 difference is generally similar to the ℓ_2 case, but is a bit more complicated. The key lemma here is based on the property of *T-truncated-Cauchy distribution* [2].

Lemma 3 (Property of T-truncated-Cauchy Distribution)**.** *Let $T = 100n$ and $y \in \mathbb{R}^n$ where each coordinate $y_i \sim$ T-truncated-Cauchy. Let $v^{(1)},\ldots,v^{(n)} \in \mathbb{R}^n$, then for sufficiently large n,*

$$\Pr\left[\frac{1}{100} \le \frac{\sum_j |y.v^{(j)}|}{\sum_j |v^{(j)}|} \le 20\ln n\right] \ge \frac{9}{10}.$$

where the "." is element-wise product between vectors.

The proof is mainly based on the T-truncated-Cauchy distribution and Markov's inequality. The idea of the algorithm is basically generate random y based on a $100n$-truncated-Cauchy distribution, and let each $v^{(j)}$ in the lemma be $(r_{\cdot j} - s_{\cdot j})$, so that the sum $\sum_j |v^{(j)}| = |r-s|$, which is just ℓ_1 difference, can be estimated.

However, the result of the algorithm is not satisfactory. For a single-pass algorithm, since we must store the randomly generated y's, it will require $\mathcal{O}(\epsilon^{-2}n\log\delta^{-1})$ space for a $(1+\epsilon)$ multiplicative error w.p. at least $1-\delta$, which is not sub-linear anymore.

1.2.3 Mutual Information Approximation

The paper proved that arbitrary precision multiplicative approximation of the mutual information requires $\Omega(m)$ space, and an additive approximation requires $\Omega(n)$ space [3]. Both are larger than linear.

2 Possible Ways for Extension

2.1 Unanswered Questions

1. The paper above suggested a way to estimate the 3 metrics given a data stream, but failed to reach a conclusion of whether X and Y are independent or not.
2. The idea presented by the paper is only suitable for discrete distribution, and quite hard to generalize to continuous distribution.
3. The paper didn't present an experiment to verify the correctness and effectiveness of the algorithms.

Inspired by problem set 5 and the χ^2-test of independence, we would like propose an algorithm that utilizes hash functions and counter matrices to store the number of times each pair (i, j) appears in the stream and calculate the three metrics along with the χ^2 statistics. We want to check how the algorithm compares to the algorithm described in the paper (performance, space usage, etc.). Below is a detailed sketch of how our algorithm may work.

2.2 Our Proposed Algorithm for Comparison (Sketch)

Recall for any two independent variables X, Y:

$$\forall i, j : \Pr[X = i, Y = j] = \Pr[X = i] \cdot Pr[Y = j]$$

If we want to determine if two variables are independent in a data stream as mentioned above, we can create matrix C with dimension $n \times n$ with initial value of zero for all elements. When a new pair of value (i, j) in the stream arrives, we increment the counter for $C[i][j]$. At the end of the stream(after receiving m value pairs), we can derived the marginal distribution mass function for variable X, Y:

$$f_X(x) = \frac{\sum_{j=0}^{n} C[x][j]}{m}, \qquad f_Y(y) = \frac{\sum_{i=0}^{n} C[i][y]}{m}$$

Then we can apply a χ^2-test with the null hypothesis that X and Y are independent. If the p-value calculated by the chi^2 statistics is small enough, we can reject the hypothesis and claim that X and Y are correlated [4]. The chi^2 statistics is given by

$$\chi^2 = \sum_{i=0}^{n} \sum_{j=0}^{n} \frac{(C[i][j]/m - f_X(i) \cdot f_Y(j))^2}{f_X(i) \cdot f_Y(j)}$$

However, this method will take $\mathcal{O}(n^2)$ space, which is not feasible when n is large. We can optimize space usage by implementing P number of independent hash functions$(h_1, h_2, ...h_p)$, where each hash function takes input in range $[0, N]$ and output in range $[0, M]$. Then we set up a counter T, a Tensor with shape (M, M, P) and every element is initialized to 0. When every values pair (a, b) in stream arrives, for each hash function h_i in $h_1, h_2, ...h_p$, increment $T[h_1(a), h_1(b), i]$. At the end of the stream

we can estimate whether variable X and Y are independent by calculating the estimated marginal probability,

$$g_X(x,p) = \frac{\sum_{j=0}^{M} T[x][j][p]}{m}, \qquad g_Y(y,p) = \frac{\sum_{j=0}^{M} T[i][y][p]}{m}$$

and the estimated χ^2 statistics,

$$\widetilde{\chi}^2 = \frac{1}{P}\left(\sum_{i=0}^{M}\sum_{j=0}^{M}\sum_{p=0}^{P} \frac{(T[i][j][p]/m - g_X(i,p)\cdot g_Y(j,p))^2}{g_X(i,p)\cdot g_Y(j,p)}\right)$$

We hope to reach the conclusion that $\widetilde{\chi}^2$ is a good estimator of χ^2, i.e. $\widetilde{\chi}^2 = (1\pm\epsilon)\chi^2$ w.p. at least $1-\delta$.

Following is the pseudo code for our proposed algorithm (sketch, may modify later):

Algorithm 2: Estimating χ^2 Statistics

Input : Number of hash functions P, output range of each hash function $[0...M]$, total length of stream m
Output: An estimate of χ^2

```
T ← 3D-Tensor with shape (M, M, P) initialized to 0;
sum ← 0;
for each pair (i, j) in stream do
    for each hash function p in 1...P do
        T[h_p(i)][h_p(j)][p] ← T[h_p(i)][h_p(j)][p] + 1
for i ← 1 to M do
    for j ← 1 to M do
        for p ← 1 to P do
            sum ← sum + (T[i, j, m] − g_X(i, p) · g_Y(j, p))^2 / (g_X(i, p) · g_Y(j, p)) ;
return sum/P;
```

The estimated χ^2 statistics can then be used to calculate a p-value, which gives us an idea of how likely X and Y are actually independent.

2.3 Our Planned Tasks

1. Try to prove our proposed algorithm satisfies the claim in section 2.2. Also express P and M in terms of ϵ and δ.
2. Implement the algorithm presented by the paper.
3. Implement our proposed algorithm.
4. Benchmark algorithm from the paper and our proposed algorithm.

5. Explore and find possible solution (if any) to approximate non-discrete data stream using the 2 algorithms.

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