二つ組データのストリームに対する
ストリームアルゴリズムの提案

任 熙宰

2013年2月

電気情報工学科

目 次

第1章

序論

1.1 研究背景

近年，インターネット環境の普及と情報技術の発達と共にインターネットから発生するデータの量は年々伸びている．急速かつ大量に発生したデータに従来のデータ処理技術を適用することは難しい．したがって，多くの研究者や企業は大量のデータから有用な情報を得る新しい技術に注目している．本研究ではその一つであるストリームアルゴリズム [1][2][3][4] について考察する．

ストリームアルゴリズムは，過去のデータを記憶するためのメモリが十分にないという状況下で時系列データを処理するアルゴリズムである．言い換えると，処理するデータの量に比べて，データの記憶及び計算に用いるメモリ空間が小さいとき，データから必要な情報だけを記録して処理する方法である．ストリームアルゴリズムの代表例として，頻出アイテム検出 [1] とデータ種類のカウント [2] という二つのアルゴリズムが存在する．前者はデータの中から多く出現したアイテムを検出するアルゴリズムである．後者はデータの中から異なるアイテムの種類数を求めるアルゴリズムである．両アルゴリズムは入力データとして，アイテム集合 X の要素からなるストリームを想定している．ところが，現実にはアイテム集合 X の要素とアイテム集合 Y の要素の組からなるストリームが多く存在する．例えば，インターネットのトラフィックデータは送信 IP と受信 IP の組からな

るストリームである．しかし，発表者の知る限り，このような二つ組のストリームに対するアルゴリズムは研究されていない．

1.2 本研究の目的

本研究では，二つ組 (x, y) のストリームを入力データとし，多種類の y と対で出現した x を検出するアルゴリズムについて考察する．初めに，頻出アイテム検出アルゴリズムの一つである Lossy Counting Algorithm[1] とデータ種類カウントアルゴリズムの一つ [2] を組み合わせて新しいストリームアルゴリズムを提案する．次にアルゴリズムの妥当性について理論的考察を行う．また，人工的に生成されたストリームデータ及び実際のダークネットトラフィックデータを用いた実験を行い，提案アルゴリズムが目標の要素を検出できることを確認する．

1.3 提案アルゴリズムの適用例

二つ組 (x, y) のストームデータは，通信，金融，流通など様々な分野で存在する．y を基地局，x を携帯端末機とすれば，(x, y) は携帯端末機と基地局の接続情報である．また，x を出発空港，y を到着空港をすれば，(x, y) は飛行機の移動経路を表わす．これらのデータに対して多数の y と対で出現した x を検出することは，多数の携帯端末機と接続している基地局，多数の空港と接続されている空港を求めることである．一方，ストリームアルゴリズムは時間的に発生した大量のデータから情報を抽出することができる．特にインターネット上では急速かつ大量に発生したデータが多く存在する．以下ではダークネットトラフィックデータを例にとり，提案アルゴリズムの適用法を説明する．

インターネット上であるサイトに接続するときは，そのサイトに割り当てられている IP アドレスを通じて接続する．ただし，IP アドレスの中にはどこのサイトにも接続されていないダークネットというアドレスが存在する．ダークネットは普段使われていないので，一般使用者がこの IP アドレスに接続する場合はほとんどない．一方，ネット環境に

は不正目的でネットに接続している攻撃者がいる．これらはセキュリティーが弱いサイトやサーバを探すことを目的にして多数の感染 PC からなるネットワーク（ボットネット）を用いて任意の目的地を検索している．これらのアドレス接続記録は一般利用者に比べて短期間に訪問頻度が倍になったりする傾向がある．特に，短期間に集中したダークネットへの接続は不明の攻撃者の行動である可能性がある．

ダークネットトラフィックデータから，接続者の IP アドレスを x，ダークネット内の IP アドレスを y とすれば，二つ組 (x, y) のストリームデータが得られる．これに提案アルゴリズムを適用すれば，アルゴリズムは多数のダークネット IP アドレスに接続する IP アドレスを出力する．x と y を入れ替えれば，多数の IP アドレスから接続されるダークネット内の IP アドレスが分かる．

急速，大量に発生するダークネットトラフィックデータに提案アルゴリズムを適用することによって，不正目的で活動しているボットネットの候補検出に応用できる．

第2章

代表的なストリームアルゴリズム

本章では代表的な二つのストリームアルゴリズムについて述べる．初めに，頻出アイテム検出アルゴリズムの一つである Lossy Counting Algorithm を紹介する．次にデータ種類カウントアルゴリズムの一つを説明する．

2.1 Lossy Counting Algorithm

Manku ら [1] が提案した Lossy Counting Algorithm について説明する．Lossy Counting Algorithm は頻出アイテム検出アルゴリズムの一つで，N 個のデータが流れてきたときに，γN 回以上出現した要素を探すことを目的とする．ただし，γ は 1 より小さい正の定数であり，サポート値とよばれる．詳しく述べると，ストリームデータ $S = \{x_i\}_{i=1}^{N}\ (x_i \in X)$ があって，x_1 から x_Nまで データが一個ずつプログラムに入力される．プログラムは限られたメモリを使って出現した要素 $x \in X$ の出現頻度を記録していく．最終的に出力を要請されたら，出現頻度が $(\gamma - \epsilon)N$ 以上である x をすべて出力する．ただし，ϵ はサポート値 γ より小さい正の定数であり，誤り値とよばれる．サポート値 γと誤り値ϵ は利用者が設定するものである．

2.1.1 アルゴリズムの定義と手順

まず，頻出情報を記録する方法について説明する．要素 $x \in X$，見積もり頻度値 f，最大許容誤差 Δ として，三つ組 (x, f, Δ) に頻出情報を記録する．三つ組 (x, f, Δ) の集合をシノプス $\mathcal{D}$ とする．流れてくるデータは $w = \lceil \epsilon^{-1} \rceil$ 個ずつに区切り，w 個のデータのまとまりをバケットとよぶ．バケットには 1 番から順に番号が付いていて，i 番目のデータはバケット $\lceil \frac{i}{w} \rceil$ に入ることになる．現在のバケット番号は b_{current} とする．アルゴリズムの詳細を以下に示す.

アルゴリズム 1 *Lossy Counting Algorithm*

1. $\mathcal{D}$ を初期化し，$i = 1$ とする.
2. S からデータ x_i を読んで，x_i が $\mathcal{D}$ になければ，$(x_i, 1, b_{\text{current}})$ を $\mathcal{D}$ に追加する.
3. x_i が $\mathcal{D}$ に既に存在したら，その三つ組の f の値を 1 つ増やす.
4. $i \neq N$ かつ $i \equiv 0 \pmod{w}$ であったら（バケットが終わったら)，$f \leq b_{current} - \Delta$ であるすべての (x, f, Δ) を $\mathcal{D}$ から削除する.
5. $i = N$ ならば，$\mathcal{D}$ から $f \geq (\gamma - \epsilon)N$ であるすべての (x, f, Δ) を出力する．そうでなければ，i の値を 1 増やしてステップ 2 に戻る.

2.1.2 出力の結果と信頼性

Lossy Counting Algorithm に対して次のような定理と補題が成り立つ．以下では議論を簡単にするために，$w = \lceil \frac{1}{\epsilon} \rceil$ を整数とする.

補題 1 [1] 三つ組 (x, f, Δ) が除去されるとき，x の実際の出現回数を f_x とすると

$$f_x \leq b_{\text{current}} \tag{2.1}$$

が成り立つ.

(証明) 数学的帰納法で証明する．まず，最初の除去の場合，除去条件 $f \leq b_{current} - \Delta$ と $f_x = f$ 及び $\Delta \geq 0$ より $f_x \leq b_{\text{current}}$ である．2 回目以降の除去の場合，過去に削除された時点で真の頻度とバケット番号をそれぞれ f'_x, b' とすると，帰納法の仮定から $f'_x \leq b'$ である．一方，f は三つ組が現れた後の頻出数なので，$f_x = f'_x + f$ が成立する．また，$\Delta + 1$ は三つ組が現れたときのバケット番号なので $\Delta \geq b'$ である．これらと除去条件 $f \leq b_{\text{current}} - \Delta$ より

$$\begin{aligned} f_x &= f'_x + f \\ &\leq b' + f \\ &\leq \Delta + f \\ &\leq b_{\text{current}} \end{aligned}$$

が得られる． □

定理 1 [1] 実際の出現回数 f_x が $f_x \geq \gamma N$ を満足する任意の x に対して，アルゴリズムが終わって時点で $\mathcal{D}$ は三つ組 (x, f, Δ) を含んでおり，かつ出力条件 $f \geq (\gamma - \epsilon)N$ が成立している．

(証明)　まず，$f_x \geq \gamma N$ を満足する任意の x に対して，アルゴリズムが終わって時点で $\mathcal{D}$ は三つ組 (x, f, Δ) を含んでいることを背理法で示す．$\mathcal{D}$ が三つ組を含んでいないと仮定する．これは，最後のバケットに入る前のどこかの時点で三つ組が $\mathcal{D}$ から除去され，その後 x が一度も出現しなかったことを意味する．三つ組が最後に除去されたときの真の頻度数とバケット番号をそれぞれ f'_x, b' とする．条件 $f_x \geq \gamma N$ より $f'_x \geq \gamma N > \epsilon N$ であり，一方，$b' \leq \lceil \epsilon N \rceil - 1 < \epsilon N$ が成立するので，$f'_x > b'$ となり (2.1) と矛盾する．

次は後半を示す．上と同様に，三つ組が最後に除去される瞬間，真の頻度数とバケット番号をそれぞれ f'_x, b' とする．除去されるまでの真の頻度数は (2.1) より b' 以下である．

$b' \leq \lceil \epsilon N \rceil - 1 < \epsilon N$, $f_x = f'_x + f$ より

$$\begin{aligned} f &= f_x - f'_e \\ &\geq f_x - b' \\ &> f_x - \epsilon N \\ &\geq (\gamma - \epsilon)N \end{aligned}$$

が得られる. □

以上の結果より，Lossy Counting Algorithm は出力として $f_x \geq (\gamma - \epsilon)N$ である要素 x をすべて出力する．詳しく述べると，Lossy Counting Algorithm は誤り値 ϵ を導入することで次のようなことが分かる.

1. $f_x \geq \gamma N$ である要素はすべて出力される
2. $f_x < (\gamma - \epsilon)N$ である要素は出力されない
3. 見積もり頻度は $f_x \geq f \geq (f_x - \epsilon N)$ を満たす

Manku らは上記の性質を ϵ 劣シノプスと定義し， 次の定理を証明した.

定理 2 [1] Lossy Counting は多くても $\epsilon^{-1}\log(\epsilon N)$ 個の三つ組を使う．すなわち，シノプス $\mathcal{D}$ が $\epsilon^{-1}\log(\epsilon N)$ 個の三つ組を記録できるようにメモリ空間を確保しておけば，Lossy Counting は ϵ 劣シノプスを満たす.

(証明)　現在の b_{current} を B とする．$i \in [1, B]$ に対して，シノプスの中から $\Delta = B - i$ である三つ組の数を b_i とする．バケット $B - i + 1$ に出現して B まで残っている三つ組は除去条件より $f \geq i$ である．一つのバケットには w 個のデータがあり，$j \in [1, B]$ に対して，バケット j までのデータ数は jw である．以上より,

$$jw \geq \sum_{i=1}^{j} \sum_{\Delta=B-i} f \geq \sum_{i=1}^{j} \sum_{\Delta=B-i} i = \sum_{i=1}^{j} id_i$$

が成り立つ．したがって，

$$\sum_{i=1}^{j} id_i \leq jw \tag{2.2}$$

次に帰納法と式 (2.2) を使い，$j \in [1, B]$ に対して

$$\sum_{i=1}^{j} d_i \leq \sum_{i=1}^{j} \frac{w}{i} \tag{2.3}$$

を証明する．

まず，$j = 1$ のときは式 (2.2) より成り立つ．$j = 1, 2, ..., p-1$ まで式 (2.3) が成立すると仮定する．仮定と式 (2.2) より，$\sum_{i=1}^{p} id_i + \sum_{k=1}^{p-1} \sum_{i=1}^{k} d_i$ に対して次の不等式が成り立つ．

$$\sum_{i=1}^{p} id_i + \sum_{k=1}^{p-1} \sum_{i=1}^{k} d_i \leq pw + \sum_{k=1}^{p-1} \sum_{i=1}^{k} \frac{w}{i}$$

左辺をまとめたら，$p \sum_{i=1}^{p} d_i$ になる．右辺は $pw + \sum_{i=1}^{p-1} (p-i)\frac{w}{i}$ となり，両辺を p で割って整理したら式 (2.3) が得られる．

シノプスのサイズは $|\mathcal{D}| = \sum_{i=1}^{B} b_i$ なので，式 (2.3) より

$$|\mathcal{D}| \leq \sum_{i=1}^{B} \frac{w}{i} \leq \frac{1}{\epsilon} \log B = \frac{1}{\epsilon} \log(\epsilon N)$$

が得られる． □

2.2 データの種類カウント

Gibbons[2] が提案したデータの種類カウントのアルゴリズムについて説明する．このアルゴリズムの目的は，ストリームデータに出現する異なる要素の数を求めることである．入力データとして $S = \left\{x_i\right\}_{i=1}^{N}$, $x_i \in X = \{1, ..., M\}$ が与えられているとする．データを記録するメモリ空間はすべての要素を記録するには足りないとする．従って，限られたメモリ空間で種類数を求めるために一部の要素のみ記録し，それを基に全体の種類数を推定する方法を使う．どの要素を記録するかの判定基準は次のような廃棄ハッシュ関数に従って行う．

2.2.1 廃棄ハッシュ関数の設定

議論を簡単にするために M を素数として, [1,M] に入る整数の組 (α,β) を考える．m を $2^{m-1} \leq M < 2^m$ を満たす整数とする．$x \in \{1, 2, ..., M\}$ に対して廃棄ハッシュ関数 $dh(x)$ は

$$f(x) = f_{\alpha,\beta}(x) = \alpha x + \beta \bmod M$$

$$dh(x) = dh_{\alpha,\beta}(x) = m - \lfloor log_2 f(x) \rfloor \tag{2.4}$$

とする．廃棄ハッシュ関数の族 $\mathcal{H} = \{h_{\alpha,\beta} | (\alpha,\beta) \in [1, M] \times [1, M]\}$ を考える．αとβ を $[1, M]$ からランダム選ぶと，任意の x に対して次のような確率を満たす.

$$\Pr[dh(x) \geq l] = \begin{cases} 1 & (l = 1) \\ 2^{m+1-l}/M & (l \geq 2) \end{cases} \tag{2.5}$$

さらに，ハッシュ関数の族 $\mathcal{H}$ はペアワイズ独立性を満たす.

定理 3 [3] 廃棄ハッシュ関数の族 $\mathcal{H}$ はペアワイズ独立性を満たし，任意の $x \neq y \in U = \{1, 2, ..., M\}$ と $i, j \in S = \{1, 2, ..., m\}$ に対して，$dh \in \mathcal{H}$ をランダムに取ると,

$$\Pr[dh(x) = i | dh(y) = j] = \Pr[dh(x) = i]$$

を満たす.

左辺は $dh(y) = j$ という条件の下で $dh(x) = i$ となる確率を表す．証明は参考文献 [3] を参照してほしい.

2.2.2 アルゴリズムの手順

上で定義した廃棄ハッシュ関数を用いるアルゴリズム [2] の手順を説明する．要素を記録するメモリ空間を Y，セレクトレベルを l とする.

アルゴリズム 2 データの種類カウント

1. l の初期値を 1，$Y = \emptyset$，$i = 1$ とする．
2. S から x_i のデータを読んで，x_i を廃棄ハッシュ関数 (2.4) に代入し，その値が l 以上および Y に x が存在しなかったら，x を Y に記録する．
3. Y が一杯になったら，廃棄ハッシュ関数値が l である x をすべて削除し，l を 1 つ増やす．
4. $i = N$ ならば，種類数の推定値 $v = \begin{cases} |Y| & (l = 1) \\ |Y|M2^{l-m-1} & (l \geq 2) \end{cases}$ を出力する．それでなければ，i を 1 つ増やしてステップ 2 に戻る．

2.2.3 アルゴリズムの結果と信頼性

アルゴリズム 2 はハッシュ関数を使って，限られたメモリ空間に一部の要素のみ記録する．全体の種類数の推定値は，メモリにある要素の数に，サンプルされる確率 (2.5) の逆数を掛けて求める．言い換えれば，種類数の推定値 v は

$$v = \frac{(Y \text{に記録された要素数})}{(\text{Y に記録する条件})} = \frac{|Y|}{\Pr[dh(x) \geq l]} = \begin{cases} |Y| & (l = 1) \\ |Y|M2^{l-m-1} & (l \geq 2) \end{cases} \tag{2.6}$$

である．ハッシュ関数の確率より，$l = 1$ なら推定値と真の種類数が一致するが，l が 2 以上なら確率を使ったので真の種類数と推定値の差は存在する．$l \geq 2$ のとき，真の種類数と推定値は Chebyshev の不等式を用いて評価できる．

定理 4 (Chebyshev の定理) [3]X を期待値 μ，標準偏差 σ である確率変数とする．任意の $\delta > 0$ に対して

$$\Pr[|X - \mu| \geq \delta\sigma] \leq \delta^{-2} \tag{2.7}$$

が成り立つ．

補題 2 [3] アルゴリズム 2 の結果に対して，真の種類数と種類数の推定値をそれぞれ n, v とする．任意の $t > 0$ に対して，

$$\Pr[|v - n| \geq tn] < \frac{M}{t^2 n 2^{m+1-l}} \tag{2.8}$$

が成り立つ．

(証明) ストリームデータ S には実際 n 種類のデータがあるとする．X_j を，ストリームデータに存在するデータ j に対して，Y に記録されていたら 1，そうでなければ 0 という確率変数とする．データ j が Y に含まれている確率，つまり，サンプル率 p_j は確率 (2.5) より j によらず，$p = 2^{m+1-l}/M$ である．ただし，ここではメモリ容量より y の種類が多くて，$l \geq 2$ の仮定で考える．$X = \sum_{j=1}^{n} X_j$ とすると，X の期待値 μ は $\sum_{j=1}^{n}(p \cdot 1 + (1-p) \cdot 0) = np$ である．X の分散は，廃棄ハッシュ関数のペアワイズ独立性よる分散の加法性で，$\sigma^2 = \sum_{i=1}^{n} \sigma_i^2 = \sum_{i=1}^{n} p(1-p) = np(1-p) < np$ である．一方，種類数の推定値 v は $|Y|/p$ である．X と $|Y|$ を同様に考えることができ，これらを Chebyshev の定理に代入すると，

$$\begin{aligned}
\Pr[||Y| - \mu| \geq \delta\sigma] &= \Pr[||Y| - np| \geq \delta\sqrt{np(1-p)}] \\
&= \Pr[|\frac{|Y|}{p} - n| \geq \delta\sqrt{\frac{n(1-p)}{p}}] \\
&= \Pr[|v - n| \geq \delta\sqrt{\frac{n(1-p)}{p}}] \leq \delta^{-2}
\end{aligned}$$

$\delta = t\sqrt{\frac{np}{1-p}}$，$t > 0$ を代入すると，

$$\Pr[|v - n| \geq tn] \leq \frac{1-p}{npt^2} < \frac{1}{npt^2}$$

が得られて，確率 (2.5) を代入して，最終的に真の種類数 n と種類数の推定値 v の差は確率で表現できる．

$$\Pr[|v - n| \geq tn] < \frac{M}{t^2 n 2^{m+1-l}} \tag{2.9}$$

□

第3章

提案アルゴリズム

前章では頻度と種類を求める二つのアルゴリズムを紹介した．本章ではそれらのアルゴリズムを組み合わせた新しいストリームアルゴリズムを提案する．

3.1 問題設定

提案アルゴリズムを説明する前に，改めて問題を設定する．二つ組データの列 $S = \{(x_i, y_i)\}_{i=1}^{N}$ が与えられているとする．ただし，任意の $i \in \{1, 2, \ldots, N\}$ に対して x_i, y_i はそれぞれ有限集合 $X = \{1, 2, \ldots, A\}$, $Y = \{1, 2, \ldots, B\}$ の元とする．このとき，S の 1 番目から i 番目までのデータにおいて $x \in X$ と対で現れる Y の元の種類数を $v_i(x)$ $(1 \leq i \leq N)$ とおく．すなわち $v_i(x)$ を

$$v_i(x) = |Y_i(x)|, \quad Y_i(x) = \{y \in Y \mid \exists j \in \{1, 2, \ldots, i\}, (x_j, y_j) = (x, y)\} \tag{3.1}$$

で定義する．例えば，x をメールの送信者，y をメールの受信者とすれば，ストリームデータ S は（送信者，受信者）の情報が数字化されて連続に並んだデータである．このとき，$v_i(x)$ は，x の人が i 番目のデータまで何人の異なる人にメールを送ったかを表す．ここで同じ人に何回を送っても，1 とカウントされることに注意する．例えば，x が i 番目のデータまで，ずっと一人に i 回メールを送ったら $v_i(x) = 1$ であり，異なる i 人に 1 回ずつメールを送ったら $v_i(x) = i$ である．

本論文では，列 S の長さ N が非常に大きいために S 中のすべてのデータを計算機のメモリに記憶することができないという状況を想定し，その下で $v_N(x)$ がある値以上となるすべての $x \in X$ を求める問題を考える．より厳密に言えば，γ を正定数として

$$v_N(x) \geq \gamma N \tag{3.2}$$

であるすべての $x \in X$ を求める問題を考える．S をメール送受信者の情報と考えると，$v_N(x) \geq \gamma N$ である x を求めることは γN 以上の人にメールを送った送信者 x を求めることである．ただし，送受信者の情報量は非常に大いため，すべての情報を計算機のメモリに記録できない状況で問題を考える．

3.2 提案アルゴリズム

提案アルゴリズムの目的は S に対して γN 以上の異なる y とペアで出現するすべての x を求めることである．列 $S = \{(x_i, y_i)\}_{i=1}^{N}$ と正整数 $M\,(\leq B)$ が与えられているとする．ここで M はアルゴリズムで y の種類を記録するメモリ空間のサイズである．サポート値 γ と誤り値 $\epsilon\,(0 < \epsilon < \gamma < 1)$ は利用者が設定する．

提案アルゴリズムは Lossy Counting Algorithm とデータ種類カウントを組み合わせたものである．Lossy Counting Algorithm はバケットの概念を使って少ない頻度の要素をメモリから削除する．データの種類カウントは廃棄ハッシュ関数を使って限られたメモリ空間で種類数を求める．提案アルゴリズムは二つのアイデアを使う．まず，要素 x に対して異なる y の種類数を廃棄ハッシュ関数を使って求める．次に，バケットの概念を使って小さい種類数を持っている x をバケットが終わったら削除する．これが提案アルゴリズムの重要な考え方である．

ここからはメモリに記憶する情報について述べる．提案アルゴリズムは，S 中のデータを一つ読み込むたびにシノプスとよばれる五つ組の集合を更新し，データを読み終えたら条件を満足する五つ組を出力する．以下ではシノプスを $\mathcal{D}$ で表し，その元を $(x, \hat{v}(x), \Delta(x), l(x), \hat{Y}(x))$ とする．ただし，$x \in X$ であり，$\hat{v}(x)$ は $v(x)$ の推定値，$\Delta(x)$ は $v(x)$ の最大許容誤差値，$l(x)$ は

廃棄ハッシュ関数の x に関するセレクトレベルを表す．また，$\hat{Y}(x)$ は $Y(x)$ を推定するための集合であり，その要素数は M を超えないとする．$x, v(x), \Delta(x)$ は Lossy Counting Algorithm から，$l(x), Y'(x)$ はデータの種類カウントから引用した．ただし Lossy Counting Algorithm では頻度値 f を記録したが，ここでは種類数の推定値 $\hat{v}$ を記録する．

アルゴリズム 3 提案アルゴリズム

1. 正定数 $\gamma, \epsilon\ (<\gamma)$ を設定する．
2. 廃棄ハッシュ関数のパラメータ $\alpha, \beta \in \{1, 2, \ldots, B\}$ をランダムに選ぶ．
3. シノプス $\mathcal{D}$ を空集合に初期化し，データ番号を表す変数 i とバケット番号を表す変数 b_{current} を 1 に初期化する．
4. S から (x_i, y_i) を読み込む．シノプス $\mathcal{D}$ に x_i を含む元が存在しないならば $\hat{v}(x_i) = 1$, $\Delta(x_i) = b_{\text{current}} - 1$, $l(x_i) = 1$, $\hat{Y}(x_i) = \{y_i\}$ として $\mathcal{D}$ に五つ組 $(x_i, \hat{v}(x_i), \Delta(x_i), l(x_i), \hat{Y}(x_i))$ を挿入し，ステップ 7 に進む．そうでなければステップ 5 に進む．
5. $dh(y_i) \geq l(x_i)$ かつ $y_i \notin \hat{Y}(x_i)$ ならば $\mathcal{D}$ の元 $(x_i, \hat{v}(x_i), \Delta(x_i), l(x_i), \hat{Y}(x_i))$ の $\hat{Y}(x_i)$ に y_i を追加する．
6. $\mathcal{D}$ の元 $(x_i, \hat{v}(x_i), \Delta(x_i), l(x_i), \hat{Y}(x_i))$ において $|\hat{Y}(x_i)| = M$ ならば，$dh(y) = l(x_i)$ であるすべての y を $\hat{Y}(x_i)$ から除去し，$l(x_i)$ の値を 1 増やす．
7. $\mathcal{D}$ の元 $(x_i, \hat{v}(x_i), \Delta(x_i), l(x_i), \hat{Y}(x_i))$ において $\hat{v}(x_i)$ の値を次式によって更新する．

$$\hat{v}(x_i) = \begin{cases} |\hat{Y}(x_i)|, & l(x_i) = 1 \text{ のとき} \\ |\hat{Y}(x_i)| B 2^{l(x_i)-m-1}, & l(x_i) \geq 2 \text{ のとき} \end{cases} \tag{3.3}$$

   ただし，m は $B < 2^m$ を満たす最小の正整数である．
8. $i < N$ かつ i が $\lceil \frac{1}{\epsilon} \rceil$ の倍数（バケットの最後に到達した）ならば $\mathcal{D}$ から

$$\hat{v}(x) \leq b_{\text{current}} - \Delta(x) \tag{3.4}$$

を満たす五つ組 $(x, \hat{v}(x), \Delta(x), l(x), \hat{Y}(x))$ をすべて除去し，b_{current} の値を 1 増やす.

9. $i = N$ ならば，シノプス $\mathcal{D}$ から

$$\hat{v}(x) \geq (\gamma - \epsilon)N \tag{3.5}$$

を見たす五つ組をすべて出力して終了する．そうでなければ i の値を 1 増やしてステップ 4 に戻る.

ここではアルゴリズム 3 の手順を更に詳しく説明する．ステップ 1 では γ と ϵ を設定する．$v_N(x) \geq \gamma N$ を満たす x を求めるのであるが，N は与えられていて利用者は γ を調節する．例えば，500 種類以上の y とペアで出現した x を求めるとき，N が 100000 であれば γ を 0.005 に設定する.

ϵ を設定したら，入力されたデータはバケットに分けて処理される．一つのバケットには $w = \lceil \epsilon^{-1} \rceil$ 個のデータが入っている．バケットには番号が付いていて，i 番目のデータはバケット $\lceil \frac{i}{w} \rceil$ に入っている．例えば，$\epsilon = 0.01$ と設定すると，一つのバケットには 100 個のデータが入っている．バケット 1 には $(x_1, y_1), ..., (x_{100}, y_{100})$ が，バケット 2 には $(x_{101}, y_{101}), ..., (x_{200}, y_{200})$ のデータが入っている．100 個目のデータを読んで次に 101 個目のデータを読むようになったら，バケット 1 が終わったと表現する．現在のバケット番号は b_{current} とおいて，最終のバケットは $\lceil \frac{N}{w} \rceil$ である.

ステップ 2 では廃棄ハッシュ関数のパラメータ α, β を設定する．α, β は [1,B] からランダム値を選ぶ．廃棄ハッシュ関数はデータの種類カウントアルゴリズムの廃棄ハッシュ関数と一致する．廃棄ハッシュ関数を $dh(y)$ とすると，

$$f(y) = f_{\alpha,\beta}(y) = \alpha y + \beta \bmod B$$

$$dh(y) = dh_{\alpha,\beta}(y) = m - \lfloor log_2 f(y) \rfloor \tag{3.6}$$

である．m は $B < 2^m$ を満たす最小の正整数である.

ステップ 1, 2, 3 でデータを受け入れる準備が終わったら，ステップ 4 からデータを読み始める．アルゴリズムは x と対で出現した y を $\hat{Y}(x)$ に記録する．ただし，メモリ空間が

十分ではなく全部を記録できないので，廃棄ハッシュ関数を使って一部の y のみ記録して $v(x)$ を推定する．ステップ 4, 5, 6, 7 は要素 y を $\hat{Y}(x)$ に記録する手順である．任意の y が $\hat{Y}(x)$ に記録されている確率 $p(l) = p(l(x))$ は，廃棄ハッシュ関数の性質より

$$p(l) = \Pr[dh(y) \geq l] = \begin{cases} 1 & (l = 1) \\ 2^{m+1-l}/B & (l \geq 2) \end{cases} \tag{3.7}$$

である．従って $\hat{v}(x)$ は，($\hat{Y}(x)$ にある y の要素数)/($\hat{Y}(x)$ に記録する確率) より求められて，式 (3.3) のように計算する．

ステップ 6 では $|\hat{Y}(x_i)| = M$，すなわち $\hat{Y}(x)$ のメモリ空間が一杯になると，$dh(y) = l(x_i)$ であるすべての y を $\hat{Y}(x_i)$ から除去する．そうしたら，廃棄ハッシュ関数の性質より $\hat{Y}(x)$ にある y の半分くらいが削除されて空き空間ができる．

ステップ 8 では，バケットが終わって削除される条件 (3.4) を満たす五つ組を削除する．最終的に $v(x) \geq \gamma N$ とならないものを削除することで，$\mathcal{D}$ のメモリ空間を節約する．Lossy Counting Algorithm はバケットが終わって削除されるものがあっても最終的に頻出度が γN 以上の要素は必ず出力される保証があった．しかし，提案アルゴリズムはデータ種類カウントと組み合わせたので，出力される保証の有無を検討する必要がある．

最後のステップ 9 では，$i = N$ ならば $\hat{v}(x) \geq (\gamma - \epsilon)N$ である x をすべて出力する．Lossy Counting では定理 1 のように $f_x \geq \gamma N$ である x は $f \geq (\gamma - \epsilon)N$ になって，目的である x を常に出力することが成立した．しかし，提案アルゴリズムでは出力されない可能性があるが，それについて理論解析で詳しく述べる．

3.3 理論解析

提案アルゴリズムは Lossy Counting Algorithm とデータの種類カウントを元で作られた．Lossy Counting Algorithm は γN 以上出現した要素は必ず出力に含まれる保証がある．データの種類カウントは真の種類数と種類数の推定値との差を Chebyshev 不等式より確率的に抑えた．本節では両アルゴリズムの信頼性の特徴が提案アルゴリズムではどうなっているか確認する．その次に，提案アルゴリズムの信頼性を評価する方法について述べる．

3.3.1 提案アルゴリズムの性質

提案アルゴリズムの目的は $v_N(x) \geq \gamma N$ である x をすべて求めることである．しかし，提案アルゴリズムでは $v_N(x) \geq \gamma N$ であるすべての x が必ず出力されるという保証がない．なぜなら，アルゴリズムは $v_N(x)$ の推定値 $\hat{v}_N(x)$ の値でデータ処理を行うからである．データ処理中，$v_N(x)$ と $\hat{v}_N(x)$ の差が発生し誤って処理されたら，$v_N(x) \geq \gamma N$ を満足する x が $\hat{v}_N(x) < (\gamma - \epsilon)N$ で出力に含まないことが起こりうる．真の値と推定値が違う原因は次のような２つがある．

1. x に対する y の種類数を求めるとき，ハッシュ関数を使って一部の y のみ記憶して種類数を求めている (種類数の推定値＝記録している y の数/記録する確率)．確率を使ったので，真の値と推定値の差は存在する．

2. 一つのバケットが終わったとき，除去条件 (3.4) より種類数が基準を超えなかった要素 x を削除する．ここで本来なら削除されない x が推定値と真の値との差で誤って削除されてしまったら，誤差は更に大きくなる．

1,2 が原因で，$v_N(x) \geq \gamma N$ を満足する x が $\hat{v}_N(x) < (\gamma - \epsilon)N$ になったら，その x は出力されない問題が発生してしまう．ところが，$v(x)$ と $\hat{v}(x)$ がどの程度違いがあって x が出力されるかどうかは，同じデータでもアルゴリズムの設定パラメータ，メモリ空間のサイズ，ランダムに作られたハッシュ関数によって異なる．従って，任意のデータを入力して目的に合う x をすべて出力する確率の推定は簡単に求められない．まず，いくつかの特殊な条件の下でアルゴリズム結果の考察を行う．

はじめに簡単のため M が十分に大きい場合を考える．このとき，$\mathcal{D}$ に含まれる任意の五つ組 $(x, \hat{v}(x), \Delta(x), l(x), \hat{Y}(x))$ において，つねに $|\hat{Y}(x)| < M$ が成立するので $l(x)$ の値は 1 のままで増加しないことに注意する．

定理 5 M が十分に大きいとする．$x \in X$ が条件 (3.2) を満たすならば，アルゴリズム 3 は五つ組 $(x, \hat{v}(x), \Delta(x), l(x), \hat{Y}(x))$ を出力する．

（証明）はじめに，b 番目のバケット終了時に五つ組 $(x, \hat{v}(x), \Delta(x), l(x), \hat{Y}(x))$ が $\mathcal{D}$ から除去されるならば，

$$v_i(x) \leq b \tag{3.8}$$

が成立することを数学的帰納法で示す．ただし，i は b 番目のバケットの最後のデータの番号を表す．これが最初の除去の場合，除去条件 (3.4) と $\hat{v}(x) = v_i(x)$ および $\Delta(x) \geq 0$ より $v_i(x) \leq b$ が成立する．2 回目以降の除去の場合，過去最後に除去されたときのバケット番号を b' とし，b 番目のバケットの最後のデータの番号を i' とすると，帰納法の仮定から $v_{i'}(x) \leq b'$ である．一方，$i+1$ 番目のデータを読み込む直前の $\hat{v}(x)$ の値は，五つ組が挿入されてからそれまでの間に x と対で現れた y の種類数なので，$v_i(x) \leq \hat{v}(x) + v_{i'}(x)$ が成り立つ．不等式が等号で成立するのは，$i+1$ 番目のデータを読み込む直前の $\hat{Y}(x)$ が $Y_{i'}(x)$ と共通の元を持たないときかつそのときに限られる．また，$\Delta(x)+1$ は五つ組が最後に挿入されたときのバケット番号なので $\Delta(x) \geq b'$ である．これらと除去条件 (3.4) より

$$\begin{aligned} v_i(x) &\leq \hat{v}(x) + v_{i'}(x) \\ &\leq \hat{v}(x) + b' \\ &\leq \hat{v}(x) + \Delta(x) \\ &\leq b \end{aligned}$$

が得られる．

次に，条件 (3.2) を満たす任意の x に対して，五つ組 $(x, \hat{v}(x), \Delta(x), l(x), \hat{Y}(x))$ はアルゴリズム 1 の終了時に $\mathcal{D}$ に含まれ，かつ出力条件 (3.5) を満たすことを示す．まずは前半を背理法で示す．$\mathcal{D}$ が五つ組を含んでいないと仮定する．このことは，最後のバケットに入る前のどこかの時点で $\mathcal{D}$ から除去され，その後 x が一度も出現しなかったことを意味する．五つ組が最後に除去されたときのバケット番号を b とし，b 番目のバケットの最後のデータの番号を i とすると，条件 (3.2) より $v_i(x) \geq \gamma N > \epsilon N$ であり，一方，$b \leq \lceil \epsilon N \rceil - 1 < \epsilon N$ が成り立つので，$v_i(x) > b$ となり (3.8) と矛盾する．次に後半を示す．上と同様に，五つ組が最後に除去されたときのバケット番号を b とし，b 番目のバケットの最後のデータの

番号を i とすると，(3.8) より $v_i(x) \leq b$ となる．一方，$\hat{v}(x)$ は五つ組が挿入されてからアルゴリズム 3 が終了するまでの間に x と対で現れた y の種類数なので $v_N(x) \leq \hat{v}(x) + v_i(x)$ が成り立つ．これらより

$$\begin{aligned} \hat{v}(x) &\geq v_N(x) - v_i(x) \\ &\geq v_N(x) - b \\ &> v_N(x) - \epsilon N \\ &\geq (\gamma - \epsilon)N \end{aligned}$$

が得られるので出力条件 (3.5) が成り立つ． □

定理 6 M が十分に大きいとする．アルゴリズム 3 において $|\mathcal{D}|$（$\mathcal{D}$ に含まれている五つ組の個数）が $\epsilon^{-1}\log(\epsilon N)$ を超えることはない．

(証明) M が十分に大きければ，定理 3 と同様に証明することができる．現在の b_{currnet} を B とする．$i \in [1, B]$ に対して，シノプスの中から $\Delta = B - i$ である五つ組の数を b_i とする．バケット $B - i + 1$ に出現してバケット B まで残っている（除去されずに）五つ組は除去条件より $\hat{v} \geq i$ である．一つのバケットには w 個のデータがあり，$j \in [1, B]$ に対して，バケット j までのデータ数は jw である．以上より，

$$jw \geq \sum_{i=1}^{j} \sum_{\Delta=B-i} \hat{v} \geq \sum_{i=1}^{j} \sum_{\Delta=B-i} i = \sum_{i=1}^{j} i d_i$$

が成り立つ．以下は定理 3 と同様である． □

定理 7 $M > \epsilon N$ とし，アルゴリズム 3 における $\hat{v}(x_i)$ の更新式 (3.3) を次式で置き換えるとする．

$$\hat{v}(x_i) = \begin{cases} |\hat{Y}(x_i)|, & l(x_i) = 1 \text{ のとき} \\ \max\{M, |\hat{Y}(x_i)|B2^{l(x_i)-m-1}\}, & l(x_i) \geq 2 \text{ のとき} \end{cases} \tag{3.9}$$

$x \in X$ が条件 (3.2) を満たすならば，シノプス $\mathcal{D}$ はアルゴリズム 3 の終了時に五つ組 $(x, \hat{v}(x), \Delta(x), l(x), \hat{Y}(x))$ を含む．

(証明） 背理法で証明する．アルゴリズム 1 の終了時にシノプス $\mathcal{D}$ が x の五つ組 $(x, \hat{v}(x), \Delta(x), l(x), \hat{Y}(x))$ を含まないと仮定する．このとき条件 (3.2) より，x の五つ組は 1 回以上 $\mathcal{D}$ から除去されている．五つ組が除去されたときのバケット番号の集合を考え，その任意の元を b とする．すると除去条件 (3.4) より，b 番目のバケット終了時に

$$\hat{v}(x) \leq b - \Delta(x) < \epsilon N - \Delta(x) \leq \epsilon N$$

が成り立つ．これは $\hat{v}(x)$ が一度も ϵN に達しない，または，$l(x)$ がつねに 1 であることを意味している．したがって，定理 1 よりアルゴリズム 1 は x の五つ組を出力する．これは背理法の仮定と矛盾する． □

M が十分大きければ，つまり M が y の種類を全部記録できるサイズであれば，提案アルゴリズムは Lossy Counting と似ていて，定理 5 と定理 6 が成立する．これは，$v(x) \geq \gamma N$ である x をすべて求められることを意味するが，実際のデータ処理に対して，メモリ空間の限界があって M を大きくできない場合がある．

定理 7 は M のサイズが小さい場合を考えた工夫である．$M \leq v(x)$ であれば $l \geq 2$ になり，条件 (3.2) を満たしても誤って除去され出力に含まれない五つ組が存在する．$\hat{Y}$ が一杯になって除去して種類数の推定値を更新するときに，新しい推定値を少くなとも今までの種類数よりは小さくならないように設定すれば，誤って除去される五つ組がなくなる．したがって，定理 7 より，条件 (3.2) を満たす x は正しく出力に含まれているかは分からないが，少なくともシノプスには残っている保証がある．

第4章

実験

本章では提案アルゴリズムの妥当性や有効性を確認するために，いくつかの二つ組データのストリームを用いて実験を行う．扱うデータは人工データと独立行政法人情報通信研究機構が管理するダークネットトラフィックデータである．実験に用いた計算機のスペックは，CPU が Intel Core i5-2430m 2.4Ghz，RAM が 8GB，OS が Windows 7 64bit である．人工データを生成するアルゴリズムと提案アルゴリズムは Visual Studio 2010 を用いて実装した．

4.1 人工データのよる実験

4.1.1 人工データの生成

入力データを $S = \{(x_i, y_i)\}_{i=1}^{N}$ $x_i \in X = \{1, 2, ..., A\}, y_i \in Y = \{1, 2, ..., B\}$ とする．提案アルゴリズムはデータ中の x_i と y_i の値によって出力結果が異なる．それを考慮して表 4.1 のようにサンプルを生成した．

サンプル 1 では，平均 501，標準偏差 30 のガウス分布に従う乱数列 $x_1, x_2, ..., x_N$ と区間 $[1, 1000]$ の一様分布に従う乱数列 $y_1, y_2, ..., y_N$ からストリーム $S = \{(x_i, y_i)\}_{i=1}^{N}$ を作成する．

表 4.1: 人工データ

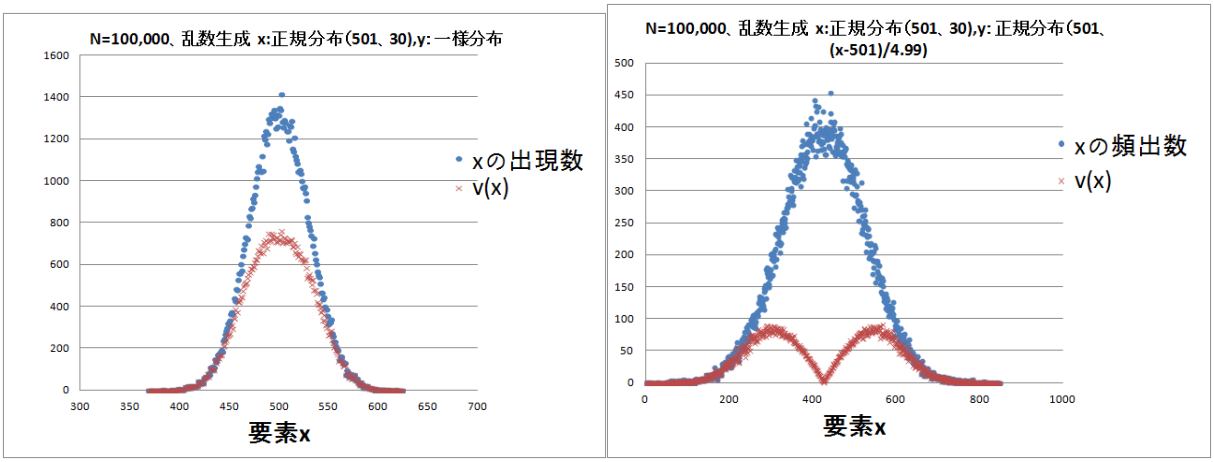
サンプル名	N	x	y
サンプル 1	100,000	Gauss 分布 (501,30)	一様分布 $y \in [1, 1000]$
サンプル 2	100,000	Gauss 分布 (501,30)	Gauss 分布 $(501, \frac{\|x-501\|}{1000-501}100))$
サンプル 3	100,000	一様分布 $x \in [1, 1000]$	一様分布 $y \in [1, 1000]$

サンプル 2 では，平均 501，標準偏差 30 のガウス分布に従う乱数列 $x_1, x_2, ..., x_N$ と平均 501 で標準偏差が x_i の値に依存するガウス分布に従う $y_1, y_2, ..., y_N$ からストリームを作成する．サンプル 3 では，$[1, 1000]$ の一様分布に従う乱数列 $x_1, x_2, ..., x_N$ と同じ一様分布に従う乱数列 $y_1, y_2, ..., y_N$ からストリームを作成する．ただし，ガウス乱数の生成には Box-Muller 法 [5] を利用し，一様乱数の生成には C 言語の rand 関数を利用した．

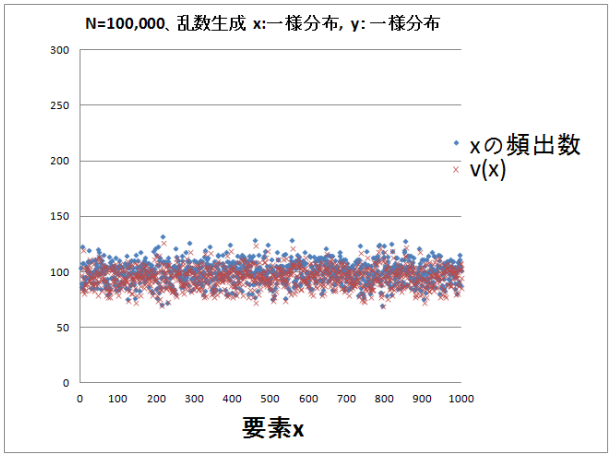
図 4.1 は，$\{x_i\}_{i=1}^N$ の頻度と，x の各値に対する y の種類数 $v(x)$ の関係を表している．サンプル 3 は x と y がともに区間 $[1, 1000]$ の一様分布で生成されたので，頻度と種類数は $N/1000$ の付近に分布している．サンプル 1 は，x がガウス分布，y が一様分布に従って独立に生成されたので，x の頻度と種類数 $v(x)$ は似た形を示している．サンプル 2 は，x をガウス分布に従って生成した後に，y を平均 501，標準偏差を $|x - 501|/4.99$ のガウス分布に従って生成した．その結果 x が平均 501 の近くでは頻出度が大きいが，y の標準偏差が小さくなるので種類数 $v(x)$ は小さくなる．次からはこれらのサンプルを使って γ, ϵ, メモリサイズを変えながら行った実験を説明する．

4.1.2 人工データの実験

アルゴリズムの結果に影響を与えるものは γ, ϵ, メモリサイズ，ハッシュ関数ある．アルゴリズムはシノプス $\mathcal{D}$ の情報をメモリ空間に記録する．最大記録できる $\mathcal{D}$ の五つ組の数を $M_{\mathcal{D}}$ とする．提案アルゴリズムの説明の際に定義した M は y の種類を記録するメモリ空間のサイズである．ハッシュ関数のパラメータ α, β の値は C 言語の rand 関数を用いて決めた．



(a) サンプル 1 (b) サンプル 2



(c) サンプル 3

図 4.1: サンプルの分布

x	$\hat{v}(x)$	$\Delta(x)$	$l(x)$
535	536	0	5
472	520	0	5
505	725	0	5
503	788	0	5
⋮	⋮	⋮	⋮
488	709	0	5
シノプスから$\hat{v}(x) \geq (\gamma - \varepsilon)N = 480$を満たす78個の$x$が出力			

(a) 出力結果

x	$v(x)$	$\hat{v}(x)$
470	554	441
471	566	425

(b) $v(x) \geq \gamma N$ だが出力に含まれなかった x

図 4.2: サンプル 1，$\gamma = 0.005, \epsilon = 0.0002, |\mathcal{D}| = 300, M = 50$ のとき，出力結果と誤差 x

実験 1 サンプル 1 を入力データとし，$\gamma = 0.005$, $M_{\mathcal{D}} = 300$, $\epsilon = 0.0002$, $M = \{30, 50, 100, 200, 400, 600, 800, 1000\}$ の設定で，提案アルゴリズムを実行した．サンプル 1 の N は 10000, $\gamma = 0.005$ なので，実験 1 の目的は $v(x) \geq \gamma N = 500$ である x を求めることである．プログラムはアルゴリズムの手順通り動作して，最終的に $\hat{v}(x) \geq (\gamma - \epsilon)N = 480$ である x を出力する．

図 4.2(a) は，$\gamma = 0.005$，$\epsilon = 0.0002$，$M_{\mathcal{D}} = 300$，$M = 50$ としたときの出力結果である．78 個の x が $\hat{v}(x) \geq 480$ を満たして出力された．アルゴリズムを評価するためには $v(x) \geq \gamma N = 500$ である x が出力に含まれているかどうか調べる必要がある．サンプル 1 には 237 個の x があり，そのうち $v(x) \geq \gamma N = 500$ を満たすものは 70 個であった．出力結果にこれらが入っているか確認したところ，図 4.2(b) に示す 2 つの x が出力されていない．二つの誤差が発生したので，条件を満たすすべての x を探すことはできなかったと判断される．

実験 1 では M を 30 から 1000 まで変化しながら，プログラムを 5 回ずつ実行した．それぞれの出力結果から，誤差数（$v(x) \geq \gamma N = 500$ だが，出力に含まれていない x の数）を求めて，平均誤差値を計算した．その結果が図 4.3 である．M が十分大きくて y の種類数を記録できるなら，誤差数は 0 に近くなり，目的とした $v(x) \geq \gamma N = 500$ である x をすべて探すことに成功したと見られる．ただし，$M = 30$ にして，y の種類数に比べてかなり

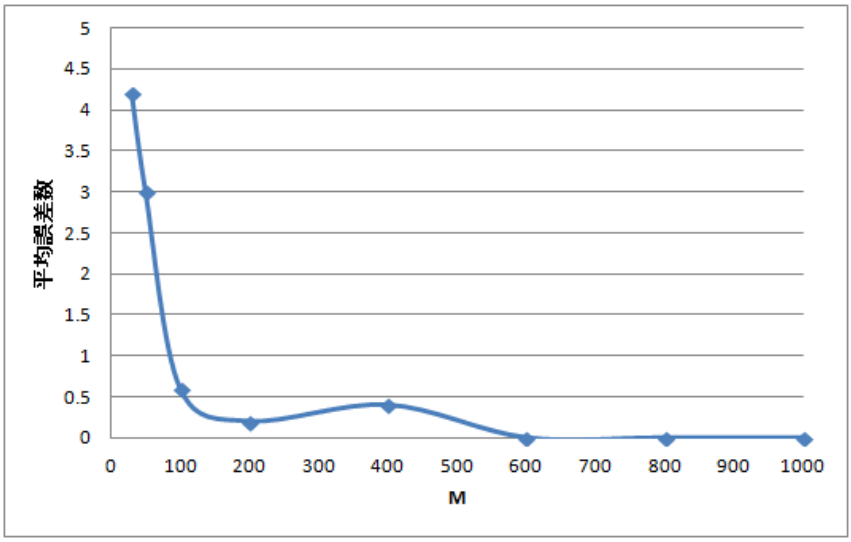


図 4.3: 実験 1 の結果 : M と平均誤差数の関係

限られたメモリ空間を利用すると，正しく出力されない x が多くなる．

実験 2 サンプル 2 を入力データとし，$\gamma = 0.0005$，$\epsilon = 0.0001$，$M_{\mathcal{D}} = 500$，$M = \{30, 50, 100, 200, 400, 600, 800, 1000\}$ の設定で，提案アルゴリズムの元で作られたプログラムを実行した．サンプル 1 は y を一様分布により生成したので，x の頻出数に比べて $v(x)$ が増加する傾向がある．実験 2 ではサンプル 2 を用いて，x の頻出数と種類数が比例しない場合を考える．つまり，多く出現したが種類数は少ない x が存在する場合について，アルゴリズムの妥当性を評価する．図 4.4 は，実験 1 と同様に上記の設定でプログラムを 5 回ずつ実行して平均誤差数を求めたものである．サンプル 2 は全部で 720 個の x が出現し，このうち $v(x) \geq \gamma N = 50$ であるものは 287 個である．図 4.4 から M が 100 以上なら，条件を満たすすべての x がアルゴリズムの出力結果に含まれていることが分かる．

実験 3 サンプル 3 を入力データとし，$\gamma = 0.001$，$\epsilon = 0.0001$，$M_{\mathcal{D}} = 500$，$M = \{30, 50, 100, 200, 400, 600, 800, 1000\}$ の設定で，提案アルゴリズムの元で作られたプログラムを実行した．サンプル 3 は x, y が [1,1000] から一様分布に従って生成されたものである．一様分布なので，$v(x) \geq 100$ である x は 338 個，$100 > v(x) \geq 90$ である x が 380 個，かなり種類数が 100 付近に存在する．実験 3 は $v(x) \geq \gamma N = 100$ である x をすべて出力することを目的

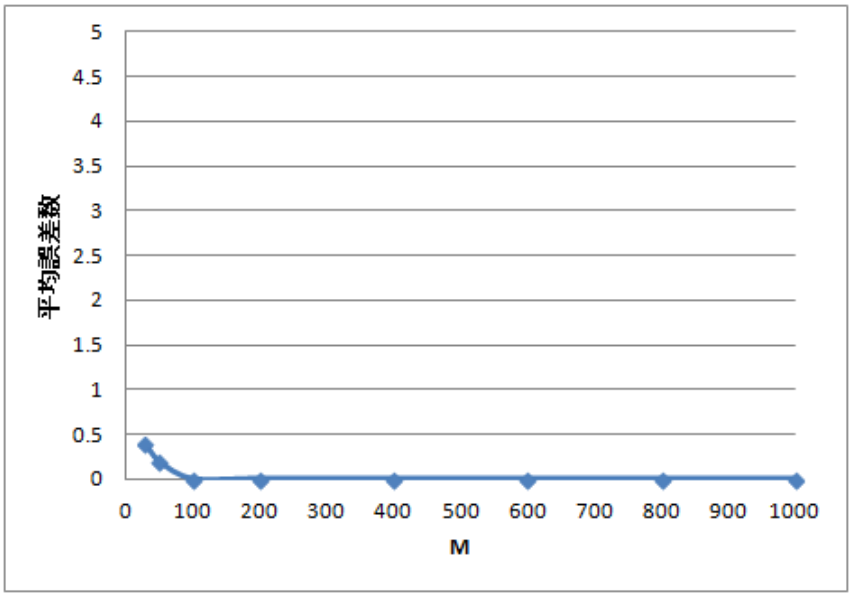


図 4.4: 実験 2 の結果 : M と平均誤差数の関係

とする．図 4.5 は M を 30 から 1000 まで変化しながら，それぞれ 5 回ずつ実行し，結果から誤差数を調べたものである．M が十分大きければ，実験 1，2 と同様にアルゴリズムは正しく x を出力する．M が小さいときは，50 個以上の x が出てこない．380 個の中から 50 個の x が出て来なかったことはかなり問題である．

実験 4 サンプル 1 を入力データとし，$\gamma = 0.005$, $M_{\mathcal{D}} = 300$, $M = \{30, 50, 100, 200, 400, 600, 800, 1000\}$, $\epsilon = \{0.0001, 0.0002, 0.0005\}$ の設定で，提案アルゴリズムの元で作られたプログラムを実行し，平均誤差数を求めた．実験 4 は，実験 1 と似ている条件であるが，ϵ を変化した結果である．実験結果から誤差数を求めた図 4.6 をみると，M が十分大きいときは ϵ の値によらずいい結果が得られる．M が小さいときは，ϵ が小さいほど平均誤差数は小さい．アルゴリズムは最後に $\hat{v}(x) \geq (\gamma - \epsilon)N$ である x を出力するので，ϵ が大きいほど，出力基準が小さくなり誤差が小さくなる．

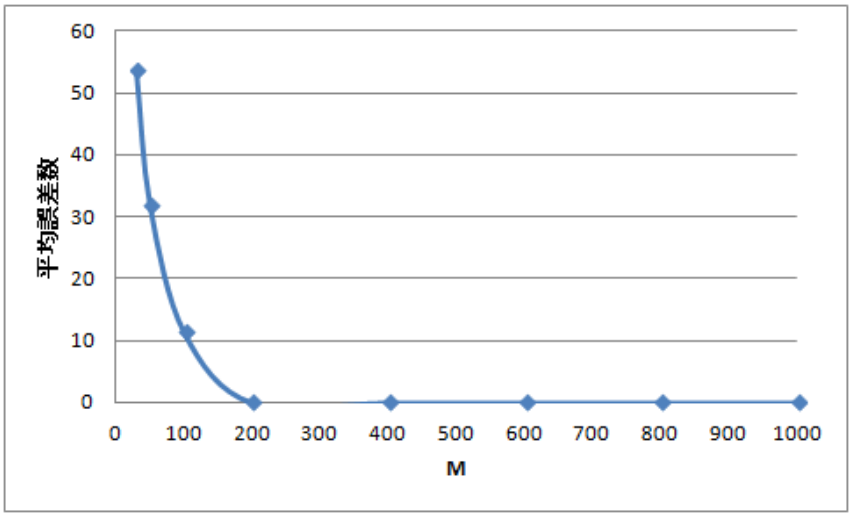


図 4.5: 実験 3 の結果 : M と平均誤差数の関係

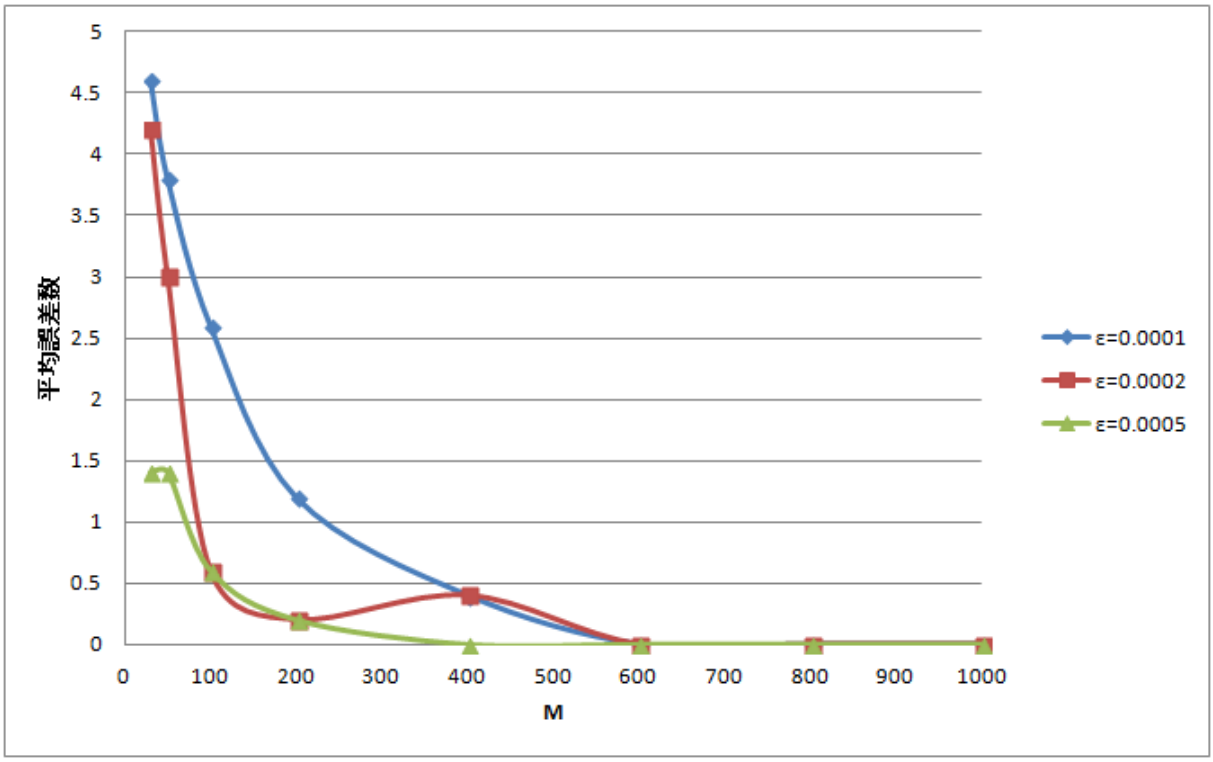


図 4.6: 実験 4 の結果 : M と平均誤差数の関係

4.2 ダークネットトラフィックデータ

本実験では独立行政法人情報通信研究機構（NICT）が管理するダークネットのトラフィックデータを用いる．このデータは，2009 年 3 月 29 日 17 時 30 分から 21 時まで 1 時間半の間にダークネット IP アドレスに接続した IP アドレスを表したもので，$N = 344891$ 個の接続記録から構成される．パケットを送信する IP アドレスを x とおき，パケットを受信する IP アドレス（ダークネット）を y とおくと，二つ組のストリームデータとして扱える．IP アドレスは 0 から 255.255.255.255 の間にあるので，一つの IP アドレスは簡単に考えて 0 から 255255255255 の間の整数と考えることができる．従ってアルゴリズムの入力データは

$$S = \left\{\left(x_i, y_i\right)\right\}_{i=1}^{344891} x_i \in X = \{0, 1, ..., 255255255255\}, y_i \in Y = \{0, 1, 2, ..., 255255255255\}$$

である．実験には多くの y と対で出現した x を求めることと，多くの x と対で出現した y を求める二つの目的がある．前者は多数のダークネット IP へ接続する IP アドレスを探し，後者は多数の IP アドレスから接続されるダークネット内の IP アドレスを求める実験である．

図 4.7(a) は，$v(x) \geq k$ を満たす x の数，すなわち k 種類以上の y と対で出現した x の数を表す．図 4.7(b) は y の種類数を表し，$v(y) \geq 1000$ である y は 1 個存在する．これは，1 時間半の間に 1000 か所以上のところから接続されたダークネット内の IP アドレスが一つあることを意味する．ダークネットは一般に接続されることは少ないので，短期間に 1000 以上のホストから接続されていることは何からの異常を示している．実験では提案アルゴリズムを使って，このようなデータを探すことを目的とする．

実験 5 ダークネットトラフィックデータ S を入力データとし，多数の x と対で出現した y を求める．具体的に実験 5 は，$v(y) \geq 35$ である y を求める．$N = 344891, B = 255255255257$（255255255255 より大きく一番近い素数）が与えられている．$\gamma = 0.0001, \epsilon = 0.00005$ と設定し，$\gamma N = 34.49, (\gamma - \epsilon)N = 17.24$ とする．$M_{\mathcal{D}} = 20000, M = 50$ として実験を行った．

条件	xの数
v(x) ≥ 1	85804
v(x) ≥ 2	5582
v(x) ≥ 10	421
v(x) ≥ 260	47
v(x) ≥ 1000	25

(a) $v(x) \geq k$ を満たす x の個数 (k=1,2,10,260,1000)

条件	yの数
v(y) ≥ 1	58713
v(y) ≥ 2	43334
v(y) ≥ 10	140
v(y) ≥ 260	2
v(y) ≥ 1000	1

(b) $v(y) \geq k$ を満たす y の個数 (k=1,2,10,260,1000)

図 4.7: ダークネットトラフィックデータの統計的性質

出力結果は表 4.2 のように $\hat{v}(y) \geq (\gamma - \epsilon)N = 17.24$ である 81 個の y が出力された．ただし，情報保護のため y の最初部分は 1111111 で表現している．推定値 $\hat{v}(y)$ の横に真の種類数 $v(y)$ を書いて比較すると，推定値と真値の間に差があることが分かる．ただし，ストリームデータ中は $v(y) \geq 35$ である y が 43 個存在しており，そのすべてが出力に含まれている．従ってアルゴリズムは目的を達成したと見られる．一方，出力に含まれている 81 個の y は，$v(y) \geq 35$ の y が 43 個，$35 > v(y) \geq 18$ の y が 38 個ある．

表 4.2: 実験 5 の結果

y	$\hat{v}(y)$	$v(y)$
111111227121	260	239
111111065120	40	40
111111054068	40	40
111111074118	520	436
111111042015	15832	17591
⋮	⋮	⋮

第5章

結論

本論文では既存の二つのストリームアルゴリズムを組み合わせて二つ組データのストリームに対する新しいアルゴリズムを提案した．提案アルゴリズムの妥当性について理論的考察を行うとともに，人工データおよびダークネットトラフィックデータを用いて実験を行い，提案アルゴリズムの有効性を検証した．提案アルゴリズムは扱うデータと設定パラメータによって誤差が発生する場合もあるが，多くの y とペアで出現する x を小さいメモリ空間を使用して検出することができる．

今後の課題は，提案アルゴリズムの信頼性の検討である．本論文では，ある条件の下で妥当性やメモリ容量について議論したが，一般の場合の妥当性の検証はできなかった．Chebyshev 不等式または Chernoff Bounds[6][7] などを利用して $\nu(x) \geq \gamma N$ を満たすすべての x を求める確率を検討したが，理論的に不十分なので本論文には記載しなかった．提案アルゴリズムの妥当性と効率的なメモリ空間の使用が次の課題として残っている．

謝辞

本研究を行うにあたって，卒業論文指導教員の九州大学大学院システム情報学科研究院情報学部門高橋規一准教授には，熱心かつ丁寧にご指導を戴いた．ここに深謝の意を表する．またゼミや中間発表でお世話になった同部門の竹内純一教授，並びに，同部門川喜田雅則助教に感謝の意を表する．回路研究室の皆様には様々な助言や協力をして頂き謝意を表する．最後に，本研究第 4 章の実験で資料を提供して戴いた独立行政法人情報通信研究機構のサイバーセキュリティ研究室に感謝する．

参考文献

[1] G.S.Manku and R.Motwani, “Approximate Frequent Counts over Data Streams,” Proceedings of the 28th International Conference on Very Large Data Bases, pp.346-357, 2002.

[2] P.B.Gibbons, ”Distinct Sampling for Highly-Accurate Answers to Distinct Values Queries and Event Reports,”Proceedings of the 27th International Conference Very Large Data Bases, pp. 541-550, 2001.

[3] 徳永豪, オンラインアルゴリズムとストリームアルゴリズム, 共立出版, 2007

[4] 伊加田恵志, 濱口佳孝, “効率的な頻出データ計数アルゴリズム Lossy Counting の拡張,” 電子情報通信学会技術研究報告, vol.107, no.524, pp.43-47, 2008.

[5] W.H.Press, 丹慶勝市訳, Numerical Recipes in C, 技術評論社, 1993

[6] D.P.Dubhashi and A.Panocnesi, Concentration of Measure for the Analysis of Randomized Algorithms, Cambridge University Press, 2009

[7] M.Mitzenmacher and E.Upfal, Probability and Computing, Cambridge University Press, 2005