トランザクションストリーム上のオンライン型頻出飽和集合マイニング

An Online Frequent Closed Itemset Mining on a Transaction Stream

福田翔士 [1*]　岩沼宏治 [2]　山本泰生 [2]
Fukuda Shoshi[1]　Iwanuma Koji[2]　Yamamoto Yoshitaka[2]

[1] 山梨大学大学院 医学工学総合教育部 コンピュータ・メディア工学専攻
[1] Computer Science and Media Engineering, Interdisciplinary Graduate School of Medical and Engineering, University of Yamanashi
[2] 山梨大学大学院 医学工学総合研究部 コンピュータ・メディア工学専攻
[2] Interdisciplinary Graduate School of Medical and Engineering, University of Yamanashi

Abstract: In order to avoid a combinatorial explosion in transaction stream pocessing, we propose a new approximation algorithm using the feature of the closed itemsets. The new algorithm LC-CloStream is an online frequent closed itemset mining algorithm obtained by combining CloStream algorithm and Lossy Counting algorithm. We give some theoretical studies and also show several results of its experimental evaluation.

1　はじめに

ストリームマイニングの代表的なアルゴリズムとして, Lossy Counting 法 [2] や Sikp LC-SS 法 [3] などが挙げられる. これらのアルゴリズムはストリームデータから誤差を許容して頻出アイテム集合を抽出するオンライン型の決定性近似アルゴリズムである. しかし, 頻出アイテム集合マイニングでは, アイテムの組み合わせ爆発のため抽出されるアイテム集合の候補数が非常に膨大になるという問題がある.

そこで本稿では, アイテム集合ではなく, その可逆圧縮形式である飽和アイテム集合を抽出の対象し, アイテムの組み合わせ爆発を抑えること目的とした, オンライン型頻出飽和アイテム集合マイニングを提案する.

2　準備

本稿で用いる表記法と用語の定義を以下に示す.

定義 1 アイテムの全体集合を $I = \{i_1, i_2, \cdots, i_n\}$ とするとき, I の部分集合を**アイテム集合**, もしくは**トランザクション**と呼ぶ. トランザクション列 $\langle T_1, T_2, \ldots, T_N \rangle$ を**トランザクションストリーム**と呼び, N を**ストリーム長**と定める.
本稿では, 以下, アイテムは英子文字 $a, b, c, \cdots$ で表し, また, 簡略化のため, アイテム集合 $\{i_1, i_2, \cdots, i_n\}$ を $i_1 i_2 \cdots i_n$ と表記する.

定義 2 $S = \langle T_1, T_2, \ldots, T_N \rangle$ をストリーム, α をアイテム集合とするとき, $S(\alpha)$ を $\{T_i \in S | \alpha \subset T_i\}$ となるトランザクション集合とする. このとき, アイテム集合 α の S 上の**絶対頻度** $s(\alpha)$ を, $s(\alpha) = |S(\alpha)|$ と定める. また, **相対頻度** $s^R(\alpha)$ を $s^R(\alpha) = \frac{S(\alpha)}{N}$ と定める.

*連絡先：山梨大学大学院医学工学総合教育部
コンピュータ・メディア工学専攻
〒 400-8511 山梨県甲府市武田 4-3-11
E-mail: g13mk021@yamanashi.ac.jp

頻出アイテム集合とは, S において, ユーザが与えた閾値（以下, **最少頻度**と呼ぶ）以上の頻度をもつアイテム集合と定める.

定義 3 (飽和アイテム集合) アイテム集合 α に対して, $\alpha \neq \alpha'$ であり $\alpha \subset \alpha'$ かつ $s(\alpha) = s(\alpha')$ であるアイテム集合 α' が存在しないとき, α を**飽和アイテム集合**と呼ぶ.

飽和アイテム集合は, アイテム集合の可逆圧縮形式の 1 種である. 飽和アイテム集合の性質として, 以下の定理が成り立つ.

定理 1 ([4]) すべての飽和アイテム集合 α, β $(\alpha \neq \beta)$ において, $\alpha \cap \beta = \gamma$ かつ $\gamma \neq \emptyset$ であるアイテム集合 γ は必ず飽和アイテム集合である.

本稿では, マイニングの対象として, 飽和アイテム集合を用いる. 定理 1 を利用して, アイテムの組み合わせ爆発を抑えることを目的とする.

通常, k 個のアイテムからなるトランザクション T_i から作られるアイテム集合の候補数は $2^k - 1$ 個となり組み合わせ爆発を起こしやすい. しかし, 飽和アイテム集合で考えた場合, メモリ内部に登録されているアイテム集合の数を L とすると, 作られる飽和アイテム集合の候補数は 定理 1 より $|L| + 1$ となる. なぜなら, メモリに保存されるアイテム集合は, すべて飽和アイテム集合である. したがって, 定理 1 より, メモリに新しく登録される飽和アイテム集合の候補は, メモリ内部に登録されてる飽和アイテム集合 と T_i の積で求められるからである.

図 1 に, 定理 1 の一例を示す. 図中の頻度表 TS とは飽和アイテム集合と頻度の組（エントリー）の集合である. TS に飽和アイテム集合 abc が頻度 1, bde が頻度 1, b が頻度 2 で登録されているとする. ストリームから新たな入力トランザクションとして $T = af$ がきた場合を考える. このとき, T はそれ自身が飽和アイテム集合であるため, 候補集合は, T と TS に登録されている飽和アイテム集合の積で求めることができる.

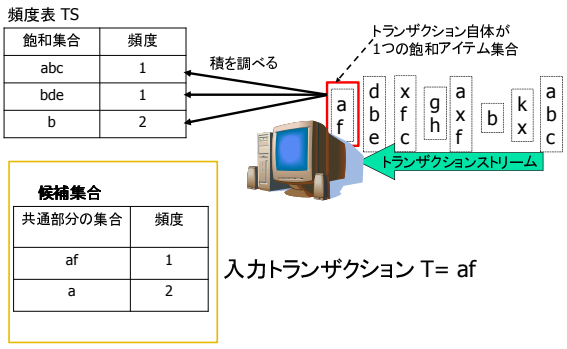

図 1: 定理 1 の一例

$af \cap abc = a$, $af \cap bde = \emptyset$, $af \cap b = \emptyset$ となるため, 候補集合は af, a の 2 つとなる.

3 先行研究

本章では, 先行研究であるストリーム中の頻出飽和アイテム集合マイニング（以下, CloStream）[4], Lossy Counting 法（以下, LC 法）[2] について概説する.

3.1 CloStream

CloStream は, 定理 1 を用いて, 高速にトランザクションストリームから飽和アイテム集合を抽出するオンライン型厳密アルゴリズムである. 頻度表とトランザクションの積は, 各アイテムの頻度表中に出現するエントリーのリストを用いることで, 効率よく求める. 図 2 に頻度表 TS とアイテムの出現エントリーのリスト表 CLT の一例を示す. ここで, CLT とは, アイテムとそのアイテムの出現エントリーの集合の 2 つ組の表である.

頻度表 TS

cid	アイテム集合	頻度
0	null	0
1	abc	2
2	bd	3
3	b	2

アイテムの出現エントリーのリスト表 CLT

アイテム	出現エントリーの集合
a	1
b	1, 2, 3
c	1
d	2

図 2: 頻度表とアイテムの出現エントリーのリスト表

アイテム b に関して, TS 中の飽和アイテム集合で b を含むものは cid 番号 $1, 2, 3$ のそれぞれのアイテム集合である. よって, この cid の集合 $\{1, 2, 3\}$ を b の出現エントリーの集合として CLT に登録する. 以下, CLT において, b に対する cid の集合を $cidset(b)$ と表記する.

3.1.1 CloStream に関する予備的評価実験

CloStream の性能を評価する. 実験には, Frequent Itemset Mining Dataset Repository[5] より疎なデータセット 1 種と密なデータセット 1 種, Machine Learning Repository[6] より密なデータセット 1 種, また, 1992 年から 2013 年までに日本で発生した地震データの 4 つのデータセットを用いる. 各データセットの詳細を以下の表 1 に示す.

表 1: 実験に使用したデータ

データセット	#(item)	#(trans.)	max (trans.)	ave (trans.)
EarthQuake	1,229	16,769	74	2.48
retail	16,470	88,162	76	10.3
mushroom	118	8,124	23	23.0
connect	129	67,557	43	43.0

CloStream に対して, 表 1 のデータセットも用いて実行時間を計る予備実験 [1] を行った. ここで条件として, CPU 時間 12 時間（= 43,200 秒） でタイムアウトとする. その結果を表 2 に示す.

表中の $N.A$ とは, 条件よりタイムアウトになったことを示している. 表 2 より, CloStream は, EarthQuake 以外のデータセットでタイムアウトとなっている. 理由としては, CloStream は厳密アルゴリズムであるため, ストリーム上に出現した全ての飽和アイテム集合の頻度を計算する. そのため, 処理が進むにつれ, 頻度表が巨大になり, 計算速度が著しく低下するのだと考えられる. よって本稿では, CloStream を厳密アルゴリズムから近似アルゴリズムに拡張し, 低頻度のエントリーを頻度表から削除しメモリ節約を行うことを考える. 拡張に用いる手法として Lossy Counting 法を利用する.

3.2 Lossy Counting 法

LC 法 [2] はストリームデータから頻出アイテム集合を抽出する誤差保証型のオンライン型近似アルゴリズムである. 相対最少頻度 σ（$0 < \sigma < 1$）, 許容誤差 ϵ（$0 < \epsilon \leq \sigma$）, ストリーム長 N のストリームデータが与えられたとき, LC 法は $s(\alpha) \geq (\sigma - \epsilon)N$ となる全てのアイテム集合 α を抽出する.

$\Delta(t)$ を時刻 t における誤差とする. アイテム集合 α において, $f(\alpha)$ を頻度表に登録されてからカウントされた実頻度, Δ_α を α が頻度表に登録された時刻 t_α での誤差とする. α が LC 法のアルゴリズムによって与えれる頻度 $c(\alpha)$ は $f(\alpha) + \Delta_\alpha$ となり, α の真の頻度 $s(\alpha)$ に対して, $f(\alpha) \leq s(\alpha) \leq f(\alpha) + \Delta_\alpha$ という関係が成り立つ. 以下, $c(\alpha)$ を**見積もり頻度**と呼ぶ. したがって, 見積もり頻度は必ず真の頻度よりも大きい値となる. 真の頻度が頻出であるならば, 見積もり頻度においても頻出となり, すべての頻出アイテム集合を抽出することができる. また, LC 法のアルゴリズムは現時刻 t において, $c(\alpha) < \Delta(t)$ となる頻度をもつ α を頻度表から削除している（**ϵ-Elimination 処理**）. これにより, メモリ消費を抑えている.

[1] この実験は, 相対最少頻度 σ の値に依存しない

4 LC-CloStream

本稿では, LC 法と CloStream の融合型である頻出飽和アイテム集合のオンライン型近似アルゴリズム LC-CloStream を提案する. LC-CloStream は, メモリ節約を行いながらストリーム上の頻出飽和アイテム集合を高速に抽出する.

LC-CloStream のアルゴリズムは, CloStream のアルゴリズムに LC 法の ϵ-Elimination 処理を導入したものである. また, それに伴い, LC-CloStream では, LC 法で用いられている見積もり頻度を使用する. ここで, 時刻 t で飽和アイテム集合 α を新たに頻度表に登録するときについて説明する. もし, α がトランザクションと, 頻度表のエントリーである飽和アイテム集合 β との共通部分だった場合, α は見積もり頻度 $1+f(\beta)+\Delta_\beta$ として頻度表に登録する. このとき, $f(\beta)$ を β が頻度表に登録されてからカウントされた実頻度, Δ_β を β が頻度表に登録された時刻 t_β での誤差とする. そうでなかった場合, つまり, 新しく出現した飽和アイテム集合 (=トランザクション) を頻度表に登録する場合, α の見積もり頻度は $1+\Delta(t)$ となる. 次に, ϵ-Elimination 処理について説明する. LC 法と同様に, 現時刻 t において, 見積もり頻度が $\Delta(t)$ 未満となる飽和アイテム集合 α を 頻度表中から削除する.

4.1 幾つかの理論的性質

LC-CloStream では 2 つの理論的性質が考えられる. 1 つは抽出した飽和アイテム集合に対する準完全性, もう 1 つは抽出した飽和アイテム集合から復元される頻出アイテム集合に対する ϵ-完全性である. 以下にそれぞれの性質について示す.

4.1.1 飽和アイテム集合に対する準完全性

アルゴリズムの都合上, 全ての頻出飽和アイテム集合は抽出できない場合が存在する. 以下に反例を示す.

例 1 (完全性の反例) 図 *3* は頻出飽和アイテム集合の抽出の完全性の反例である. 相対最少頻度 $\sigma = 0.4$, 許容誤差 $\epsilon = 0.1$ とする. また, ストリームは $T_1 = ab, T_2 = ab$ と到着したのち, T_3 から T_{21} まで xyz のトランザクションが到着する. その後は abc のトランザクションが到着し続けるという状況を想定する. 図 *3* の上図は T_{21} を読み込んだときの真の飽和アイテム集合とアルゴリズムにより計算された飽和アイテム集合を表している. T_{21} を読み込んだ結果, アルゴリズムは飽和アイテム集合 xyz を頻度 19, ab を頻度 2 で保持している. しかし, abc の頻度は ϵN 未満であるため, ϵ-*Elimination* 処理により削除される. 次に図 *3* の下図より, $T_{22} = abc$ を読み込んだとき, アルゴリズムで保持される飽和アイテム集合と, ここまでのストリームから抽出される真の飽和アイテム集合は下図に通りであり, アルゴリズムは ab を抜き出せていない. このまま, ストリームを読み続けた場合, 本来なら頻出飽和アイテム集合として ab を抜き出す必要があるが, アルゴリズムは抜き出すことができない.

しかし, アルゴリズムは以下の性質が成り立つ.

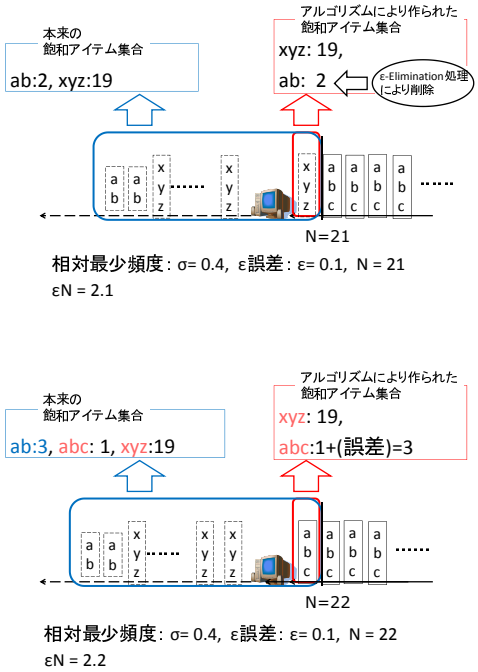

図 3: 頻出飽和アイテム集合の抽出の完全性の反例

定義 4 (ϵ-拡大不可能な飽和アイテム集合) 飽和アイテム集合 α とその絶対頻度を $s(\alpha)$ とし, ストリーム長 N, 許容誤差 ϵ とする. α に対して, $\alpha \supset \alpha'$ かつ $sup(\alpha) - sup(\alpha') < \epsilon N$ である飽和アイテム集合 α' が存在しないとき, α を **ϵ-拡大不可能な飽和アイテム集合**と呼ぶ.

定理 2 (準完全性) *LC-CloStream* は, ϵ-拡大不可能な飽和アイテム集合で頻出なものは全て抽出することができる.

紙面の都合上, 定理 2 の証明は省略する.

4.1.2 頻出アイテム集合に対する ϵ-完全性

時刻 i におけるトランザクションを T_i とする. 任意のアイテム集合 α について, 以下の集合族 $TS(\alpha, n)$ を定義する: $TS(\alpha, n) = \{\beta | \alpha \subseteq \beta, \langle \beta, f(\beta), \Delta(\beta) \rangle \in TS(n)\}$. ただし, $TS(n)$ は, 時刻 n における頻度表とする. $f(\alpha)$ と $\Delta(\alpha)$ を以下のように定義する:

1. $TS(\alpha, n)$ が空のとき, $f(\alpha) = 0, \Delta(\alpha) = \Delta_n$, ただし, Δ_n は時刻 n で処理が終了した時点での頻度誤差.
2. そうでないとき, $f(\alpha) = f(\alpha_{max(n)}), \Delta(\alpha) = \Delta(\alpha_{max(n)})$, ただし, $\alpha_{max(n)} = argmax(\beta \in TS(\alpha, n))$.

α の見積もり頻度 $c(\alpha)$ を $c(\alpha) = f(\alpha) + \Delta(\alpha)$ と定義する. 相対最少頻度を σ ($0 < \sigma \leq 1$) とする.

時刻 n での処理が終了した時点で, 出力要求があったとき, アルゴリズムが抽出するアイテム集合族を $S(n)$ とする. このとき, $S(n) = A(n) \cup B(n)$ とする. ただし,

$$A(n) = \{\alpha | \alpha \in TS(n), c(\alpha) \geq \sigma t\}$$
$$B(n) = \{\beta | \beta \subseteq \alpha, \beta \neq \emptyset, \alpha \in A(n)\}$$

である. 任意のアイテム集合 α について, 時刻 i における真の頻度を $sup(\alpha, i)$ とする.

LC-CloStream は頻出アイテム集合に関して, 以下の完全性が成り立つ.

定理 3 (ϵ 誤差保証) 任意の時刻 n において, 任意のアイテム集合 α について, $f(\alpha) \leq sup(\alpha, n) \leq f(\alpha) + \Delta(\alpha)$ を満たす.

定理 4 (ϵ-完全性) $\sigma n \leq \Delta(n)$ のとき, 任意の頻出アイテム集合 α は $S(n)$ に含まれる.

この 2 つの定理より, アルゴリズムで抽出した頻出飽和アイテム集合から, すべての頻出アイテム集合を, 頻度誤差 ϵt $(= \Delta(t))$ の範囲で求めることができる. 紙面の都合上, これらの定理の証明は省略する.

4.2 LC-CloStream の評価実験

LC-CloStream に対して, 表 1 のデータセットを用いて実行時間を計る実験 [2] を実験 3.1.1 と同様に行った. その結果を表 2 に示す.

ここで, ϵ とは許容誤差である. 表 2 より, LC-CloStream は retail, mushroom, connect で大幅な速度向上がみられる. 12 時間 (=43,200 秒) でタイムアウトであるため, retail と mushroom は 10 倍以上実行時間が早くなったことがわかる. しかし, EarthQuake において, $\epsilon = 0.001$ のとき, 実行時間は LC-CloStream のほうが遅い. これは頻度表と, CLT を操作するオーバーヘッドがかかっているためだと考えられる. そこで, 次は LC-CloStream のデータ構造の改良を考える.

4.3 データ構造の改良案

CloStream, LC-CloStream ともに, CLT を用いて頻度表のエントリーを絞り込み, 候補集合の計算の高速化を図っている. しかし, そのために行う和の計算が非常に難しい. また, LC 法の導入により, 頻度表エントリーの削除に伴い, CLT の更新が必要となる. しかし, 先行研究である CloStream ではこれらの計算を効率化するデータ構造の考慮がなされていない. そこで, 本稿では, データ構造の改良を行う.

まず, CLT の和の計算について考える. 任意 (不特定) 個の集合の和を効率的に求めることはかなり難しいので, ここでは, 和は求めずに直接登録されている cid のエントリーとトランザクションの積を求める. そして, 積を求めた cid は計算済みとしてハッシュに登録する. もし, ハッシュに登録されている cid が出現したら, その cid はスキップする. これにより, 和の計算を省略し, 実行時間の高速化を図る.

次に, エントリーの削除を伴う CLT の更新について考える. 改良案では, エントリーに CLT へのポインタを持たせ, エントリーからダイレクトアクセスで削除できるように改良した. これにより, CLT の更新を定数時間で行い, 処理の高速化を図る.

[2] この実験は, 相対最少頻度 σ の値に依存しない

4.4 改良型アルゴリズムの評価実験

データ構造を改良した改良型 CloStream, 改良型 LC-CloStream に対して, 表 1 のデータセットを用いて幾つかの評価実験 [3] を行った. その結果を以下に示す.

4.4.1 実行速度

まず, これまでと同様に改良型 CloStream, 改良型 LC-CloStream の実行速度を図る実験を行った. その実験結果を表 2 に示す.

まず CloStream と改良型 CloStram を比較する. retail, connect に関しては $N.A$ のままであったが, EarthQuake では約 15 倍ほど速度が早くなった. また, 改良型 CloStream では mushroom での処理が終了するようになり, 約 24 倍以上速度が向上している. LC-CloStream と改良型 LC-CloStream の比較に関しても, 改良型 LC-CloStream は retail, mushroom では約 10 倍, EarthQuake と connect では約 30 倍の速度向上がみられる. このことから, データ構造を改良したことで実行時間を大幅に短縮することができた.

4.4.2 頻度表サイズの推移

次に, 改良型 CloStream と改良型 LC-CloStream での処理中の頻度表サイズの推移を比較する. その結果を図 4 に示す. 尚, 今回は紙面の関係上, mushroom の結果のみを紹介する.

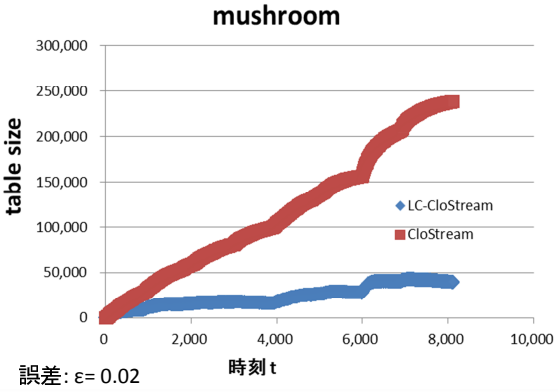

図 4: 処理中の頻度表サイズの推移: mushroom

図 4 において, 横軸が時刻 t, 縦軸が頻度表サイズである.

図 4 より, 改良型 CloStream での頻度表サイズは, 時間ともに上昇し, 最終的なサイズは 25 万である. それに対し, 改良型 LC-CloStream での頻度表サイズの推移は, 非常に緩やかに上昇しており, 最終的なサイズ は 5 万である. これは改良型 CloStream と比較して $\frac{1}{5}$ となっている. したがって, LC-CloStream は, $epsilon$-Elimination 処理が効果的に働いており, 大幅なメモリ節約を行っていることがわかる.

[3] 実験 4.4.1, 4.4.2, 4.4.4 は, 相対最少頻度 σ に依存しない

表 2: 各アルゴリズムの実行時間

データセット	ϵ	CloStream	LC-CloStream	改良型 CloStream	改良型 LC-CloStream
EarhQuake	0.001	1,468.8 s	2,630.9 s	90.0 s	72.8 s
	0.002		1,291.4 s		71.0 s
	0.003		1,199.3 s		70.2 s
retail	0.001	*N.A*	4,709.1 s	*N.A*	489.0 s
	0.002		892.7 s		286.2 s
	0.003		324.2 s		241.1 s
mushroom	0.02	*N.A*	4,780.4 s	1,791.9 s	187.0 s
	0.03		2,368.3 s		122.0 s
	0.04		1,424.3 s		89.8 s
connect	0.80	*N.A*	*N.A*	*N.A*	18,872.4 s
	0.85		*N.A*		8,935.8 s
	0.90		36,418.6 s		4,103.9 s

4.4.3 ϵ を変化させたときの最終頻度表サイズと実行時間

次に, 改良 CloStream-LC に対して, 許容誤差 ϵ を変化させたときの最終的な頻度表サイズと実行時間の関係を確認するための実験を行った. その実験結果の図 5 に示す. 尚, 今回は紙面の関係上, connect の結果のみを紹介する.

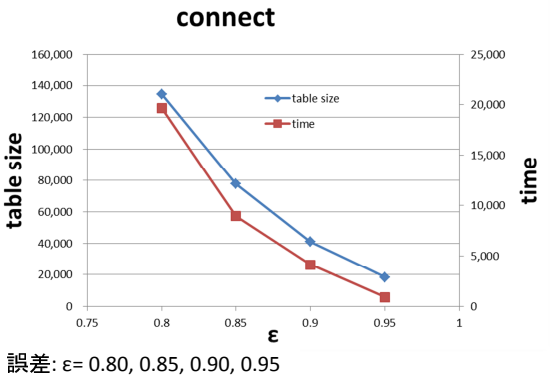

図 5: ϵ を変化させたときの最終頻度表サイズと実行時間

横軸は許容誤差 ϵ である. 左縦軸は各 ϵ での最終的な頻度表を大きさであり, 右縦軸は各 ϵ での実行時間である. 図 5 より, 最終的な頻度表の大きさは, ϵ の値と大きくするにつれ, 大きさは小さくなっている. また実行速度についても同様に, ϵ の値と大きくするにつれ, 速度は早くなっている. 他のデータセットに対しても, 同じ傾向が見られた. このことから, 頻度表の大きさと実行時間は比例関係にあり, 頻度表の大きさが小さくなるほど, 実行時間も早くなるのだと考えられる.

4.4.4 候補集合数

次に, 改良 CloStream と改良 LC-CloStream に対して, 時刻毎の候補集合の数を確認する実験を行った. この結果を以下の表 3 に示す.

max(freq.), ave(freq.) はアイテム集合をマイニングしたときの候補集合数の最大と平均である. max(Clo.), ave(Clo.) は飽和アイテム集合を改良型 CloStream でマイニングしたときの候補集合数の最大と平均である. max(LC-Clo.), ave(LC-Clo.) は飽和アイテム集合を改良型 LC-CloStream でマイニングしたときの候補集合数の最大と平均である.

まず, アイテム集合での候補集合数と提案手法である LC-CloStream での飽和アイテム集合の候補集合数について比較する. 各データセットにおいて, アイテム集合でマイニングした場合の候補集合数と比べて, LC-CloStream でマイニングした場合の候補集合数は非常に小さい数となっている. 特に, retail について, アイテム集合の最大候補集合数は 2^{76} 個であるところ, LC-CloStream での最大候補集合数は $\epsilon = 0.001$ のとき 920 であった. つまり, 飽和アイテム集合にすることで, 755 垓個 の候補集合を 920 個まで大幅に削減している. また, connect についても, アイテム集合での最大候補集合数が 2^{43} 個であるところ, LC-CloStream では 最大候補集合数 が $\epsilon = 0.8$ で 123,554 個であった. 飽和アイテム集合にすることで, 約 9 兆 個の候補集合数を 10 万個まで削減している.

次に, CloStream と LC-CloStream を比較すると, LC-CloStream の候補集合数は CloStream の候補集合よりも小さい値となっているが, ほとんど変わらない. ここで, mushroom の結果に注目する. CloStream と LC-CloStream の候補集合数に大きな差は見られない. しかし, 表 4 より, 頻度表サイズの推移の大きな違いが見られ, 結果として, 表 2 より, LC-CloStream の実行速度は, CloStream よりも 10 倍早くなっている. したがって, 候補集合数はわずかな違いでしかないが, 処理が進むにつれて, その差が積み重なっていき, 最終的な頻度表サイズに大きく影響を及ぼすと考えられる.

以上のことから, 飽和アイテム集合を扱うことで, アイテムの組み合わせ爆発を大幅に抑えていることがわかる. 加えて, 提案手法である LC-CloStream は, CloStream よりも組み合わせ爆発の抑制効果が高いと考えられる.

4.4.5 再現率と適合率

改良型 LC-CloStream に対して, 表 1 のデータセットを用いて, 頻出飽和アイテム集合を抽出する実験を行った. このとき, 各データセット毎で許容誤差 ϵ は固定し, 相対最少頻度 σ を変化させて実験を行っている. この実験結果を以下の表 4 に示す. ここで, U をアルゴリズムが抽出した頻出飽和アイテム集合の集合, X を U 中の真の頻出飽和アイテム集合の集合, A をストリーム中の真の頻出飽和アイテム集合の集合とする. このとき, 再現率, 適合率を以下に定義する.

$$\text{再現率} : \frac{|X|}{|A|}, \quad \text{適合率} : \frac{|X|}{|U|}$$

表 3: 候補集合数

データセット	ϵ	max(freq.)	ave(freq.)	max(Clo.)	ave(Clo.)	max (LC-Clo.)	ave (LC-Clo.)
EarhQuake	0.001	2^{74}	$2^{2.5}$	5,793	12.9	5,718	11.7
	0.002					5,717	11.4
	0.003					5,712	11.2
retail	0.001	2^{76}	$2^{10.3}$	$N.A$	$N.A$	920	32.6
	0.002					654	25.1
	0.003					518	22.2
mushroom	0.02	2^{23}	2^{23}	8,554	1,175.9	5,953	926.9
	0.03					4,994	828.4
	0.04					4,316	749.3
connect	0.80	2^{43}	2^{43}	$N.A$	$N.A$	123,554	48,627.3
	0.85					77,166	31,408.0
	0.90					40,739	16,779.4

表 4: 再現率と適合率

データセット	σ	ϵ	抽出したアイテム	再現率 (%)	適合率 (%)
EarhQuake	0.007	0.001	146	100	88.4
	0.008		112	100	87.5
	0.009		87	100	93.1
retail	0.007	0.001	327	100	96.3
	0.008		248	100	98.0
	0.009		196	100	98.5
mushroom	0.2	0.02	1,185	95.8	96.8
	0.3		423	94.8	95.7
	0.4		140	91.4	91.4
connect	0.90	0.8	3,608	100	96.6
	0.91		2,915	100	96.7
	0.92		2,313	100	95.6

表 4 では, EarthQuake, retail, connect については再現率 100 %であったが, mushroom は, 再現率 100 %にはならなかった. 今回のデータセットの傾向から, 疎なデータセットよりも密なデータセットのほうが, アルゴリズムによる頻出飽和アイテム集合の抽出漏れが起きやすいのではないかと思われる. しかし, connect も密なデータセットであったが, 再現率は 100 %となっており, すべての頻出飽和アイテム集合を抽出できている. ここで, connect と mushroom を比較してみると, 表 1 より, connect のストリーム長は mushroom の 10 倍程度の大きさがある. ストリーム長が長いデータセットでは, ϵ-Elimination 処理で飽和アイテム集合を削除しても, 処理の途中で再び出現する可能性が高くなり, 最終的な結果として, すべての頻出飽和アイテム集合を抽出できるのではないかと推測する.

ここで ϵ-完全性により Y を U から求められる頻出アイテム集合とし, B をストリーム中の頻出アイテム集合としたとき, 頻出アイテム集合における再現率を以下に定義する.

$$\text{頻出アイテム集合における再現率} : \frac{|Y|}{|B|}$$

アルゴリズムの理論的性質から, ストリーム中における頻出アイテム集合についての再現率は 100 %となる. したがって, この結果は圧縮の可能性を示している.

適合率は EarthQuake を除いてすべて 90 %以上であり, 高精度に飽和アイテム集合を抽出している. EarthQuake の適合率が下がってしまっている要因としては, σ と ϵ の値が近かったため, 誤差による影響をうけてしまい, 余分な飽和アイテム集合を多く抽出してしまったと考えられる.

5 まとめと今後の課題

本稿では, アイテムの組み合わせ爆発を抑えることを目的に, 飽和アイテム集合のオンライン型近似アルゴリズムを提案した. 先行研究である CloStream を拡張した LC-CloStream を提案し, 幾つかの理論的性質を示した. また, 先行研究では考慮されていないデータ構造の改良案を提案した. アルゴリズムを実装し評価実験を行った結果, 候補集合数を大幅に削減し, 組み合わせ爆発を抑えることに成功した. また, データ構造の改良により, 実行速度を大幅に向上させた.

今後の課題として, 頻度表のエントリーを固定することで, 高速にマイニングを行うアルゴリズムの検討を行っていく. また, 理論的性質と実験結果から LC-CloStream はある種の圧縮を行っているといえる. そのため, 今後, 圧縮に関しても検討も行っていく.

謝辞

本研究は一部, ISPS 科学研究費補助金 (No.25330256) および JST さきがけの援助を受けている.

参考文献

[1] 有村博紀: 大規模データストリームのためのマイニング技術の動向, 電子情報通信学会論文誌, Vol.J88-D-I, pp.563–575 (2005)

[2] G. S Manku and R. Motwani: Approximate frequency counts over data streams. *Proc VLDM'02*, pp.346–357 (2002)

[3] Y. Yamamoto, K. Iwanuma, S. Fukuda: Resource-oriented Approximation for Frequent Itemset Mining from Bursty Data Streams. *SIGMOD'2014*, 2014, June 22-27, Showbird, UT, USA.

[4] S. Yen, C. Wu, Y. Lee, V.S. Tseng, C. Hsieh: A Fast Algorithm for Mining Frequent Closed Itemsets over Stream Sliding Window. *IEEE International Conference on Fuzzy Systems*, 2011, June 27-30, Taipei.

[5] Frequent Itemset Mining Dataset Repository. http://fimi.ua.ac.be/data/. (最終アクセス日: 2015/1/9).

[6] Machine Learning Repository. http://archive.ics.uci.edu/ml/datasets/Mushroom. (最終アクセス日: 2015/1/9)