

領域効率の良い頻出データアイテム発見アルゴリズム

川副 真治[†] 有村 博紀[†]

[†]九州大学大学院システム情報科学研究科・研究院

〒812-8581 福岡市東区箱崎 6-10-1

E-mail: †{s-kawa,arim}@i.kyushu-u.ac.jp

あらまし 頻出データアイテム発見問題は、 N 個のデータアイテムの列から、与えられたしきい値 $0 \leq \sigma \leq 1$ 以上の出現頻度を持つデータアイテムをすべて発見する問題であり、ネットワーク監視やテキストマイニング等に広い応用をもつ。本稿では、本稿では、この問題に対して最近 Manku と Motowani (VLDB'02) が与えた 1 パス近似アルゴリズム LossyCounting を元に、2 回のデータ走査と $O(\frac{1}{\sigma} \log N)$ の記憶領域で、すべての頻出アイテムを発見する 2 パスアルゴリズム DoubleScan を与える。アルゴリズムは LossyCounting をそのまま停止させずに 2 回データ上を走らせるだけの単純なものだが、その正当性は、データ列の巡回に関する自明でない組み合わせ的性質に基づいている。さらに Savasere 等 (VLDB'95) の 2 パスアルゴリズム Partition の理論的解析を与えることで、Partition と比べて DoubleScan は著しく領域計算量が小さいことを示す。計算機実験では、合成データと実データの両方で、DoubleScan は Partition に比べて計算時間が数倍程度大きいことが分かった。

キーワード データマイニング, 知識発見, 結合規則発見, 外部記憶アルゴリズム

A Space-efficient Algorithm for Finding Frequent Item Lists from Large Disk-resident Data

Shinji KAWASOE[†] and Hiroki ARIMURA[†]

[†] Department of Informatics, Kyushu University

6-10-1 Hakozaki, Fukuoka 812-8581, Japan

E-mail: †{s-kawa,arim}@i.kyushu-u.ac.jp

Abstract The *frequent item list problem* is the problem of, given a stream of N data items and a threshold $0 < \sigma < 1$ for the minimum frequency, finding all *frequent data items* that have frequency above σ , and it has many useful applications such as network monitoring, text mining, and iceberg query processing. In this paper, we propose a space-efficient two-pass algorithm called *DoubleScan* that exactly solves the frequent item list problem in $O((1/\sigma) \log N)$ space by simply running the approximation online algorithm called LossyCounting recently developed by Manku and Motowani (VLDB'02). Then, we present theoretical and empirical comparison of DoubleScan algorithm and the well known two-pass exact algorithm, called Partition, by Savasere *et al.* (VLDB'95), and show that DoubleScan is superior to Partition in space order of magnitude.

Key words Data Mining, Knowledge Discovery from Databases, External memory algorithm.

1. はじめに

多くの大規模データ解析事例において、潜在的に非常に大きな数のデータアイテム全体 \mathcal{I} に対して、データアイテムの集合であるデータベース \mathcal{D} から、事前に与えられたしきい値 $0 \leq \sigma \leq 1$ 以上の頻度を持つごく少数のデータアイテムを発見する問題が重要になる。このような問題を、頻出アイテム発見問題 (The frequent item list problem) [6] とよぶ。

例えば、ネットワークログ解析では、データベースは、ネッ

トワーク通信データを tcpdump 等のパケット解析ツールで解析して得られたログデータであり、ここからあらかじめ専門家が指定した複数の属性の値を組み合わせることで得られた属性の組がデータアイテムとなる。具体的には、送信ホストと、通信タイプ、サービス (ポート) の組 (216.239.53.101, tcp, http) や、受信ホストと送信ホスト、ポートの組 (133.5.24.1, 216.239.53.101) 等がデータアイテムの例であり、分析者は、一定の頻度以上出現する組に注目し、不正侵入検出やネットワーク障害の兆候を発見する。

この頻出アイテム発見問題は、代表的なデータマイニングである結合規則発見問題と比べて、基本となる少数の属性があらかじめ固定されているため、一見自明な問題にみえる。実際、次の単純なアルゴリズム（以後、Naive アルゴリズムとよぶ）で計算可能である。ここで、データベース $\mathcal{D} = (d_1, \dots, d_N) \in (\mathcal{I})^*$ は、長さ N のデータアイテム列とする。

Algorithm Naive

```
foreach  $i = 1, \dots, |\mathcal{D}|$  do:
  • if  $(d = \mathcal{D}[i]) \in \mathcal{C}$  then  $C[d] = C[d] + 1$ ;
  • else  $C[d] = 1$ ;
```

図 1 Naive アルゴリズム

しかし、この自明なアルゴリズムの領域計算量は $O(N)$ であり、可能なアイテム数 $|\mathcal{I}|$ とデータサイズ $N = |\mathcal{D}|$ が非常に大きいときには、きわめて非効率である。実際に、ネットワークログデータでは、一度だけ出現するようなものも含めるとデータアイテム数は膨大なものとなり、固定された主記憶容量よりもはるかに大きな値になり得る。

そこで本稿では、少ない領域とデータ走査数で、厳密にすべての頻出アイテムを計算する効率良いアルゴリズムを開発することを目標として研究をおこなう。

1.1 定義

形式的には、頻出アイテム発見問題を次のように定義する。アイテム (item) の集合を \mathcal{I} とする。サイズ N のデータベース (database) とは、アイテムの配列 $\mathcal{D} = d_0 \dots d_{N-1}$ ($d_i \in \mathcal{I}$) である。 \mathcal{D} の第 i 番目の要素を $\mathcal{D}[i]$ で表す。 \mathcal{D} におけるアイテム p の出現数を次のように定義する：

$$\text{hit}_{\mathcal{D}}(p) = \sum_{i=0}^{N-1} [d_i = p].$$

ここに、関数 $[E]$ は式 E が真のとき 1 をとり偽のとき 0 をとる。 p の出現頻度は $\text{freq}_{\mathcal{D}}(p) = \text{hit}_{\mathcal{D}}(p)/|\mathcal{D}|$ である。このとき、最小頻度とよばれる正数 $0 < \sigma \leq 1$ に対して、 p が σ 頻出であるとは、 $\text{freq}_{\mathcal{D}}(p) \geq \sigma$ が成立することをいう。

頻出データアイテム発見問題

入力: 長さ N のアイテム配列 \mathcal{D} と最小サポート $0 < \sigma \leq 1$.

問題: すべての σ 頻出なアイテム $p \in \mathcal{I}$ を出力せよ。

1.2 関連研究

この問題を解く自明な方法は、先に示した Naive 法 (図 1) である。Naive 法は、1 回のデータ走査しか行なわれないが、 $O(|\mathcal{I}|)$ の主記憶を必要とし、最悪時の領域計算量は $O(N)$ である。

これに対して、Savasere 等 [7] は、二回のデータ走査でこの問題を厳密に解く分割型アルゴリズム Partition を与えた。これは、データを一定サイズのブロックに分割し、ブロックごとに計算した頻出アイテムを併合するアイデアを用いている。彼らは領域量の解析を与えていないが、本稿の 2 節では、Partition の領域計算量を解析し、これが $O(\sqrt{N/\sigma})$ となることを示す。

最近、Manku と Motowani (VLDB'02) は、すべての頻出アイテムを厳密に求めるのではなく、1 回のデータ走査で近似

的に頻出アイテムを求める近似計数技法を提案した。与えられた誤差パラメータ ε に対して、彼らの近似アルゴリズム Lossy Counting は、すべての頻出アイテムを発見し、それらの頻度を真の頻度から高々 ε 程度高い程度で推測可能である。反対に、非頻出アイテムを見つける可能性もあるが、 $\sigma - \varepsilon$ 以下の頻度をもつアイテムは決して発見されないことが保証される。

1.3 主結果

本稿では、Manku と Motowani (VLDB'02) のオンラインアルゴリズム Lossy Counting をデータ上で 2 回走らせることで、きわめて小さい記憶領域で頻出データ発見問題を厳密に解くことができることを示す。

[定理 1] 図 3 のアルゴリズム DoubleScan は、ちょうど 2 回のデータ走査と領域計算量 $O(\frac{1}{\sigma} \log N)$ を用いて、頻出データアイテム発見問題を厳密に計算する。

主結果の鍵は、今回示した頻度列の巡回 (rotation) に関する次の組み合わせの結果である：

[定理 2] 要素の総和が N となる長さ N の任意の整数配列 A に対して、次の性質をみたく巡回 B が存在する： B のすべての接頭列 $B[0 : i]$ ($0 \leq i < N$) に対して、その要素の総和 $\sum_{j=0}^i B[j]$ が長さ i 以上になる。

この定理により、Manku と Motowani のアルゴリズムが 2 回のデータ走査の間に、すべての頻出アイテムの出現を正しく検出することが保証される。

さらに、合成データと実データを用いた計算機実験で、提案のアルゴリズムを Naive と Partition の 2 つのアルゴリズムと比較する。結果として、提案の DoubleScan アルゴリズムは計算速度は遅いが、著しく小さな領域計算量で頻出データアイテム発見問題を解くことが分かった。また、逐次的な入出力特性により、外部記憶アルゴリズムとして用いることができる。このことから、DoubleScan アルゴリズムは限られた記憶領域の元で、非常に大きなデータ集合に対して頻出アイテムを発見する場合に有効である。

1.4 本稿の構成

はじめに 2 章で、Savasere 等 [7] の Partition アルゴリズムを与え、その領域計算量を解析する。3 章で、Manku と Motowani (VLDB'02) のオンラインアルゴリズム Lossy Counting を簡単に紹介し、われわれが提案する DoubleScan アルゴリズムを与える。ここで、上の定理 2 を証明し、これを用いて主結果を示す。4 章で、実験結果について述べ、5 章でまとめを述べる。

2. 分割型 2 パスアルゴリズム

2.1 アルゴリズム

図 2 に、Savasere et al [7] の分割型 2 パスアルゴリズム Partition を示す。Partition は、入力パラメータとしてブロックサイズ $0 < B \leq N$ をとり、1 パス目で \mathcal{D} をサイズ B のブロック $B_1 + \dots + B_{\lceil N/B \rceil}$ に分割し、各ブロックごとに局所的な候補アイテム集合を計算する。2 パス目で、それらを併合して全体の候補アイテム集合とし、正確な出現数を計算する。アルゴリズム Partition は、次の観察に基づいている。

[補題 1] (Savasere et al. [7]) \mathcal{D} 上でアイテム $p \in \mathcal{I}$ が σ

Algorithm Partition

入力: 長さ N のアイテム配列 D および, 最小サポート値 $0 < \sigma \leq 1$,
ブロックサイズ $0 < B \leq N$.

ステージ 1:

- アイテム配列 D を, $n = \lceil N/B \rceil$ 個のサイズ B のブロック $D = B_1 \cdots B_n$ に分割する.
- 各 $1 \leq i \leq n$ に対して, ブロック B_i を主記憶に読み込み, B_i 内で出現回数 $\text{freq}_{B_i}(p) \geq B\sigma$ を満たすすべてのアイテム $p \in B_i$ を求め, 局所的候補集合 C_i に格納する (パス 1)

ステージ 2

- 大域的候補集合 $C := C_1 \cup \cdots \cup C_n$ を計算する.
- アイテム配列 D を左から右へ走査しながら, 各候補アイテム $p \in C$ の出現回数 $g(p)$ を計算する (パス 2)
- $g(p)/N \geq \sigma$ を満たすすべてのアイテム $p \in C$ を頻出アイテムとして出力する.

図 2 Partition アルゴリズム

頻出ならば, p は少なくとも 1 つのブロック D_i ($1 \leq i \leq N/B$) 上で σ 頻出である.

[補題 2] (Savasere 等 [7]) 図 2 のアルゴリズム Partition は, 頻出データアイテム問題を 2 回の走査で厳密に解く.

2.2 領域計算量の解析

アルゴリズム Partition の領域計算量はブロックサイズ B の選択に大きく依存するが, Navasere 等 [7] は, 領域計算量の理論的解析は行っていない. そこで本節では, Partition の領域計算量を簡単に解析する.

[定理 3] Partition は, 頻出アイテム配列問題を $\sqrt{N/\sigma}$ の領域量で解くよう実装可能である.

[証明] ステージ 1 では, 各ブロック B_i を 1 回ずつ順に処理するので, 領域計算量の総和は高々 $N = \sum_i |B_i|$ である. 各ブロック B_i で, 頻出アイテムは $B\sigma$ 回以上出現し, また各エントリは 1 個のアイテムしか含まないので, B_i は高々 $B/B\sigma = 1/\sigma$ 種類の頻出アイテムを含む. これを併合すると, 大域的な候補集合の総数は高々 $\max\{B, N/(B\sigma)\}$ である. これを解くと, B の最小値は $B = \sqrt{N/\sigma}$ となる. 後は, 入力からこの B を計算するように, アルゴリズムを構成すればよい. \square

領域計算量はアイテムの分布にも依存する. 例えば一様分布では, Partition の定理 3 の上界よりも小さな領域計算量をもつ. しかし, 任意の分布を許した場合には, 定理 3 がタイトな理論的上界を与える.

[定理 4] Partition が使用する領域量が少なくとも $\Omega(\sqrt{N/\sigma})$ となるようなデータベース D が存在する.

[証明] 任意の N と B に対して, $1/(N\sigma)$ 個のアイテムを考え, 各ブロックに $1/(B\sigma)$ 個の異なるアイテムがそれぞれ $B\sigma$ 回ずつ出現すると仮定するようなアイテム配列を考える. このとき, 上の補題の証明と同様の議論で下界を示せる. \square

定理 3 の理論的見積もりは, 領域量の大まかな上限に過ぎない. 例えば $N = 10^6, B = 10^4, \sigma = 0.75\%$ のとき, 補題 3 に

よる領域量の理論的上限は $|C| = 1333$ と大きい, Navasere 等 [7] による実測値では $|C| = 170$ である.

3. 提案アルゴリズム

本節では, 頻出データアイテム発見問題を 2 回のデータ走査と $O(\frac{1}{\sigma} \log N)$ 領域で解く走査型 2 パスアルゴリズム DoubleScan を提案する. さらに, その理論的解析を行なう.

3.1 アルゴリズム

図 3 に, 提案のアルゴリズム DoubleScan を示す. このアルゴリズムは, Manku と Motowani [6] の 1 パス近似アルゴリズム LossyCounting を流用して構成したもので, 2 回繰り返してアイテム配列を走査する点以外は, LossyCounting とまったく同一のアルゴリズムである. 領域計算量の解析は, パラメータの意味が変わるが, Manku と Motowani [6] の議論がそのまま成立する.

しかし, 2 回同じ走査をくりかえすことで, 1 パス近似アルゴリズムが, 2 パスの厳密計算アルゴリズムとして働くことの証明は自明ではない. この正当性の証明を本節で行なう.

Algorithm DoubleScan

入力: 長さ N のアイテム配列 D と最小サポート値 $0 \leq \sigma \leq 1$.

- $B := \lceil 1/\sigma \rceil$; /* ブロック長の設定 */
- D を, $n = \lceil N/B \rceil$ 個のサイズ B のブロック $D = D_0 \cdots D_{n-1}$ に分割する.
- $C := \emptyset; \mathcal{F} := \emptyset$ /* 初期化 */
- 以下の操作を, $current = 0, \dots, 2n - 1$ に対して繰り返す:
 - $k := current \bmod n$ ($0 \leq k \leq n - 1$).
 - k 番目のブロック D_k を主記憶に読み込み, 各アイテム $p \in D_k$ に対して, もし $p \in C$ ならば, $p.f := p.f + 1$ とカウンタを増やす. それ以外のとき, p を C に加えて, $p.f := 1$ と $p.s := current$ で初期化する.
 - 条件 $p_j.f < current - p_j.s + 1$ を満たすすべての候補 $p \in C$ を C から削除する.
 - 条件 $current - p.s + 1 = n$ を満たすすべての候補 $p \in C$ を頻出アイテムとして A に加え, 同時に C から削除する.
- Return \mathcal{F} ;

図 3 DoubleScan アルゴリズム

アルゴリズム DoubleScan は, 初めにブロック長 $= \lceil 1/\sigma \rceil$ を計算し, 入力アイテム配列 D を n 個のサイズ B のブロック $D = D_1 \cdots D_n$ に区切る. このブロック長 B の選択がアルゴリズムの鍵である. その後, アイテム配列 D 全体を左から右にちょうど 2 回走査しながら, 各ブロックを順に主記憶に読み込み, 出現したアイテムを候補集合 C に追加し, その後, 出現するたびにその計数を更新する.

候補集合 C 中の各候補アイテム $p \in C$ は, 次の情報を属性としてもつ:

- 候補集合に挿入されたときの時刻 $p.s \geq 0$.
- 出現回数のカウンタ $p.f \geq 0$.

この候補集合 C の管理の方針は, つぎのとおりである.

- 初めて出現したアイテムは, C に追加する .
- C に追加されたアイテムは, 条件 $p.s + p.f - 1 \geq \text{current}$ が成立する間は C に保持しつづける .

3.2 解 析

アルゴリズム DoubleScan は, 次のようなアイデアに基づく . 今, ある頻出アイテムがデータ系列 D に $M \geq \sigma$ 回出現していると仮定する . すると, 今 D を長さ $B = \lceil 1/\sigma \rceil$ の等長ブロックに分割したとする . すると, p は頻出であるから, 平均して各ブロックに 1 回以上出現すると期待できる . そこで, アイテム p に対して, それが隣接したブロックに連続して 1 回以上出現する限り, p を候補集合 C に保持する戦略をとる . こうすると, 頻度が σ 以上のパターンは長期にわたって C にとどまりつづけ, 頻度が σ 未満のパターンは C から除去されると期待できる .

ただし, 実際には, アイテムの出現は任意であるから, もちろんアイテムが疎に出現しているところや, 密に出現するところが存在する . そこで, 候補 $p \in C$ が候補として C に追加された後の累積出現数 $p.f$ を計数し, この累積出現数を用いて p の管理を行なうという戦略を採用する .

[定義 1] データ系列を 2 回走査して得られるブロック列を $S = (D_i)_{i=0, \dots, 2n-1}$ とし, 任意のアイテムを $p \in \mathcal{I}$ とおく . このとき, p に関する S 上の長さ ℓ の成功連 (successful run) とは, 連続した S の添え字の列 $0 \leq i_0, \dots, i_{\ell-1} \leq 2n-1$ で次の条件をみたすものをいう :

- 添え字は連続している . すなわち, 任意の j に対して, $i_{j+1} = i_j + 1$ が成立する .
- p が候補となってから現在までの累積出現数から, 候補となってから現在までのブロック数 (時間) を引いたものが非負である . すなわち, 任意の j に対して, $p.j.f \geq j - p.j.s + 1$ が成立する .

この戦略の巧妙な点は, ある時刻 t にアイテム p が C にとどまっているためには, p は自分の生存期間に十分多く D に出現する必要があることである . 一方で, 一つの時刻には一つのアイテムしか出現できないという制約があるので, 固定した正数 ℓ 以下の生存期間をもつようなアイテムは高々 $B\ell$ 個しか存在し得ない . したがって, ブロック長任意の時刻 t で C のサイズは調和級数

$$\frac{B}{1} + \frac{B}{2} + \frac{B}{3} + \dots + \frac{B}{n} = B \cdot H(n) = O\left(\frac{1}{\sigma} \log \frac{N}{B}\right)$$

で抑えられることがわかる . ここに, $B = 1/\sigma$ であり, $n = N/B$ である .

この議論は, Manku と Motowani [6] による LossyCounting の領域計算量の解析で, 誤差パラメータ ε を最小サポート σ で置きかえたものと同じであり, DoubleScan の領域計算量は, LossyCounting のもので ε を σ で置きかえたものになる . よって, 次の補題が成立する .

[補題 3] (Manku と Motowani [6]) 図 3 のアルゴリズム DoubleScan は, ちょうど 2 回のデータ走査と領域計算量 $O\left(\frac{1}{\sigma} \log N\right)$ を用いて, 頻出データアイテム発見問題を厳密に計算する .

以下では, アルゴリズム DoubleScan を D 上のブロック系列 $(D_i)_{i=1, \dots, 2n-1}$ 上で走らせたと仮定する . 時刻 $0 \leq t \leq 2n-1$ におけるアイテム $p \in C$ の生存時間とは, $t - p.s$ である .

候補管理戦略の定義から, アイテム p に関する成功連があれば, DoubleScan がその期間は p を候補として C に保持しつづけることは明らかである . これに関して, 次の補題が成立する . [補題 4] 任意のアイテム p と時刻 $0 \leq t \leq 2n-1$ に対して, 時刻 t で終わる p に関する長さ $\ell \geq 0$ の成功連が存在することと, 時刻 t における p の生存時間が ℓ 以上であることは同値である .

次の補題は, p が十分長い時間 C で生存しつづけたならば, D における p の頻度が $p.f$ で与えられることを示す .

[補題 5] 任意のアイテム p に対して, ある時刻 $0 \leq t \leq 2n-1$ において, 時刻 t における p の生存時間がちょうど n であるならば, 時刻 t において $\text{hit}_D(p) = p.f$ が成立する .

DoubleScan の正当性は, 任意の頻出アイテム p が長さ n の成功連をもつかどうかにかかっている . この鍵は, 今回示した頻度列の巡回 (rotation) に関する次の組み合わせ的性質である . 詳細については, 文献 [5] を参照されたい .

[定理 2 (再)] 要素の総和が N となる長さ N の任意の整数配列 A に対して, 次の性質をみたす巡回 B が存在する : B のすべての接頭列 $B[0 : i]$ ($0 \leq i < N$) に対して, その要素の総和 $\sum_{j=0}^i B[j]$ が長さ i 以上になる .

上の定理から, 次の補題がただちに成り立つ .

[補題 6] アルゴリズム DoubleScan を D 上のブロック系列 $(D_i)_{i=1, \dots, 2n-1}$ 上で走らせたと仮定する . このとき, アイテム p が D 上で σ 頻出であることと, p に関する長さ n の成功連が存在することは同値である .

補題 5 と補題 6 から, 本稿の主結果が導かれる .

[定理 1 (再)] 図 3 のアルゴリズム DoubleScan は, ちょうど 2 回のデータ走査と領域計算量 $O\left(\frac{1}{\sigma} \log N\right)$ を用いて, 頻出データアイテム発見問題を厳密に計算する .

4. 実 験

開発したアルゴリズム DoubleScan の性能を評価するために, 人工データと実データを用いて, 頻出アイテム発見の計算機実験を行った .

4.1 デ ー タ

表 1 実験に用いたデータ

名称	サイズ N	異なりアイテム数 I	生成方法
Ohsumed	12MB	740,000 個	英文テキスト
Uniq	1MB	10,000 個	一様分布乱数
Zipf	1MB	10,000 個	Zipf 分布乱数

実験には, 表 1 に示すデータを用いた . Ohsumed は, 英文医療文献データ MEDLINE の記事 86MB からなるデータ OHSUMED^(注1) から, 頻出英単語を発見する問題である . 記事中のすべての単語に一意な id 番号を与えて, 約 740,000 個

(注1): <http://ftp.ics.uci.edu/pub/machine-learning-databases/ohsumed/>

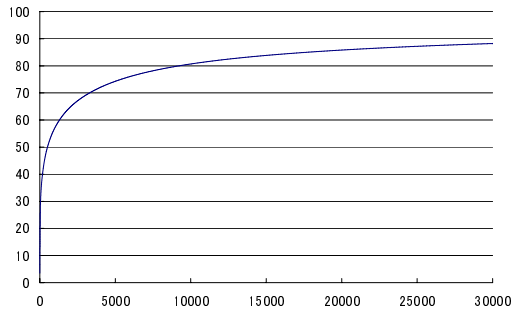


図 4 Ohsumed のアイテム累積頻度分布

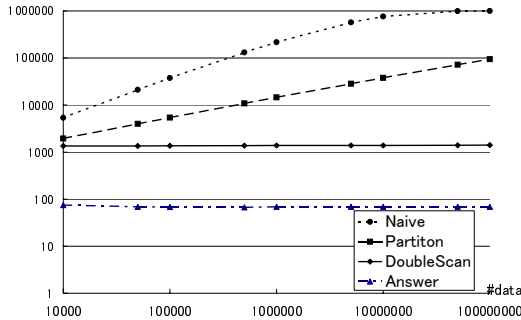


図 5 実験 1 . データサイズと領域計算量

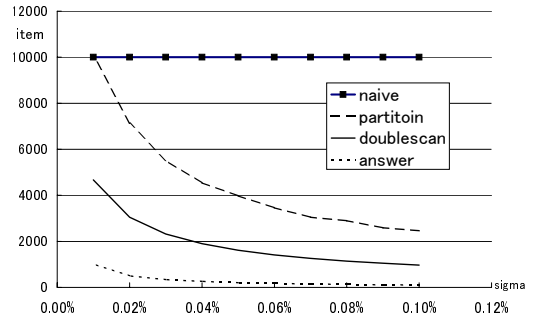


図 6 実験 2 . 最小サポート値と領域計算量

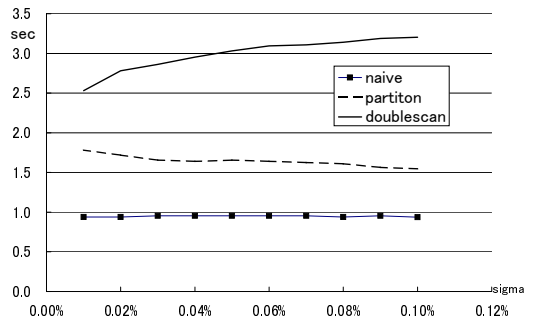


図 7 実験 2 . 最小サポート値と計算時間

の異なりアイテムに変換した。

Unif と Zipf は、それぞれ、一様分布乱数 $P(x) = 1/I$ と Zipf 分布 $P(x) = x^{-\alpha}$ を用いて生成した合成データである。総アイテム数は $|I| = 10,000$ 個である。Zipf 分布は、上の Ohsumed データを近似しており、Ohsumed の推定パラメータ $\alpha = 1.188$ に対して、Zipf は $\alpha = 1$ としている。なお、出現頻度の上位 5000 個で全体の約 70% を占めている。図 4 に Ohsumed データの頻度分布図を示す。

4.2 実験方法

次の 3 つのアルゴリズムを、C++ 言語と STL (C++ Standard Template Library) を用いて実装した。

- * Naive: 1.2 節のアルゴリズム Naive を、候補集合 C に STL の拡張可能なハッシュ辞書を用いて実装したもの。
- * Partiton: 2. 節で示した Savasere 等の分割型 2 パスアルゴリズム Partiton (図 2) を実装したもの。定理 3 にしたがって、最小化ブロックサイズを $B = \sqrt{N/\sigma}$ とした。
- * DoubleScan: 今回提案した操作型 2 パスアルゴリズム DoubleScan (図 3) を実装したもの。候補集合の更新には、 C の線形探索を用いた (優先度付き待ち行列を用いた版も実装したが、予備実験では、有為な差が認められなかった。)

すべての実験は PC (Pentium4, 2.5GHz, RAM 1GB, WindowsXP) 上で行った。

4.3 結果

4.3.1 実験 1 : データサイズと領域計算量

合成データ Zipf において、最小サポート値 $\sigma = 0.1(\%)$ とし、データサイズ $N = 10\text{KB}$ から 100MB まで変化させて、3

つのアルゴリズムが使用する領域量を比較した。実データと異なり、解の異なり数は一定であることに注意されたい。

図 5 に結果のグラフを示す。図の上から Naive, Partiton, DoubleScan アルゴリズム, 正解数 (最終的な頻出パターン数) の順で使用領域の対数をプロットしている。これより、例えばデータサイズ 1M 個前後では、正解数 $\alpha = 100$ に対して、DoubleScan の領域量は、その 10α 程度であり、Partiton の 100α や Naive の 1000α に対して、圧倒的に領域効率が良い。参考として表 2 に、データサイズ $N = 5 \times 10^6$ (item) のときの各アルゴリズムの領域量を示す。

表 2 アルゴリズムによる領域量の比較。データサイズ 5×10^6 。

	Naive	Partiton	DoubleScan	Answer
領域量	572,069	28,460	1,388	69
相対比	$\times 412$	$\times 21$	$\times 1$	$\times 0.04$

4.3.2 実験 2 : 最小サポート値と領域計算量

実験 1 の全データを用いて、最小サポート値を $\sigma = 1\% \sim 0.1\%$ に変化させ、3 つのアルゴリズムの領域量を比較した (注 2)。

図 6 に結果を示す。一般に最小サポート σ が小さいほど (図の左側)、正解数と領域量が急速に大きくなる。図から、DoubleScan と Partiton が、サポート値に自動的に適応して、領域量を節約していることと、DoubleScan の領域量が特に小さいことがわかる。注目すべき点は、サポート値が非常に低い値 ($\sigma = 0.01\%$) で、Partiton は Naive と同程度に領域効率が悪

(注 2): 公平を期すために、Partiton のパラメータ探索を行い、最適なブロックサイズ B を選択した。

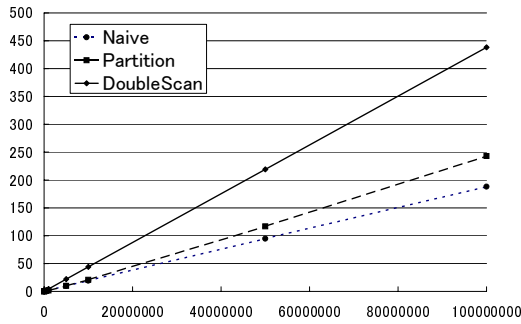


図 8 実験 3 . データサイズと計算時間

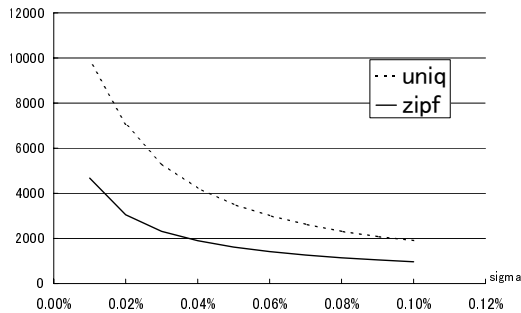


図 9 実験 4 . データによる違い : 計算領域

いものに対して, DoubleScan はその 2 倍程度領域効率が良い点である. 図 7 に, 最小サポート値の変化に対する計算時間 (sec) を示した. これらの結果から, 広い範囲の最小サポート値において, DoubleScan は, 計算時間では遅いが, 領域効率が著しくよいことがわかる.

4.4 実験 3 : 計算時間

図 8 にデータサイズを変化させた場合の, 計算時間を示す. 図の上から, DoubleScan, Partition, Naive の順でプロットしており, DoubleScan は Naive の 2 倍程度の計算時間を必要とすることがわかる. また, どのアルゴリズムもデータサイズに関して線形時間であり, 規模耐性は良い.

しかし, 図 5 の領域量の結果から, 主記憶サイズが, 例えば 10K 個のアイテム分に制限されたとき, Naive は, $N = 10^4$ 個超で, Partition は $N = 10^6$ 個弱で主記憶の限界に達する. これに対して, この実験では, 正解数は定数であり, DoubleScan は長期に動きつづけることが予想される. この意味で限られた量の主記憶しか使えない状況で, DoubleScan は規模耐性が良いといえる.

4.5 実験 4 : データの違いと領域計算量

最後に, 頻度分布のことなる 2 種の合成データで DoubleScan の性能比較を行なった. 図 9 に, 最小頻度 σ を変化させたときの領域計算量を示す. これより, 現実によくみられる Zipf 分布において, DoubleScan はより領域計算量が小さいという良い性質をもつことがわかる. 計算時間に関しても同様の結果が得られた. また, 実データ Ohsumed での実験でもこの観察が裏付けられた.

5. ま と め

本稿では, 頻出データアイテム発見問題に対する領域効率の良いアルゴリズム DoubleScan を提案した. これは, データ走査に DoubleScan は, 長さ N のデータに対して, 2 回の走査と, N の線形時間および領域 $O(\frac{1}{\sigma} \log N)$ 領域で, すべての頻出アイテムを厳密にすべてを見つけることを示した. さらに, 人工データとテキストデータ上で領域計算量を, 自明なアルゴリズム Naive および分割型 2 パスアルゴリズム Partition と比較した.

実験結果をまとめると, DoubleScan は, Partition に比べて計算速度では 2 倍程度劣るが, 領域効率は 10 倍程度と非常によく, また適応性に優れている. したがって, 非常に大量のデータを対象とする応用に, きわめて有効であると期待される. ネットワークデータの解析や, テキストマイニングなど, 実際のデータマイニング問題への適用が今後の課題である.

6. 追 記

最近になって, Karp, Papadimitriou, Shenker 等 [4] が, 頻出アイテム問題を領域計算量 $O(1/\sigma)$ で厳密に解くアルゴリズムを提案した^(注3). この結果は, σ に関して最適な領域計算量 $O(1/\sigma)$ を達成しており, 究極の結果だといえる.

彼等のアルゴリズムは, 出現したアイテムを主記憶上の候補集合に入れるところまでは, Lossy Counting と同じであるが, 主記憶に空きがあるあいだは何もせず, 主記憶がいっぱいになると, 主記憶に空きができるまで, すべての候補アイテムのカウントを同時に減らしていき, 最初にカウントが 0 になったアイテムと新たなアイテムを入れ替える. この戦略の正当性は非自明であり, エレガントな戦略である.

同じ σ の値に対しても, 実際の正解 (頻出アイテム数) が $1/\sigma$ より小さい場合もある. このような場合に, 最小限の領域しか使わないような適応的なアルゴリズムの開発は, 依然として現実の問題では重要であり, 今後の研究課題である.

文 献

- [1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules, In Proc. VLDB'94, 487-499, 1994.
- [2] S. Brin, R. Motwani, C. Silverstein. Beyond Market Baskets: Generalizing Association Rules to Correlations, In Proc. SIGMOD Conference 1997, 265-276, 1997.
- [3] C. Estan and G. Verghese. New directions in traffic measurement and accounting, In Proc. ACM SIGCOMM Internet Measurement Workshop, November 2001.
- [4] R. M. Karp, C. H. Papadimitriou, S. Shenker, A simple algorithm for finding frequent elements in streams and bags, Manuscript, 2002. (Available at the authors home page)
- [5] 川副真治, データベースから頻出アイテムを求める領域効率の良い 2 パスアルゴリズム, 九州大学大学院システム情報科学府, 情報理学専攻, 修士論文, 平成 15 年 2 月.
- [6] G. S. Manku, R. Motwani, Approximate Frequency Counts over Data Streams, In Proc. VLDB'02, 2002.
- [7] A. Savasere, E. Omiecinski, S. Navathe, An Efficient Algorithm for Mining Association Rules in Large Databases, In Proc. VLDB'95, 1995.

(注3): この結果の存在は, Manku と Motwani 等 [6] に個人的な消息として触れられていた. 手稿の内容がはっきりしたのは最近である.