テーマ２：列挙アルゴリズム

本テーマの目的

- 頻出集合の列挙
- 受けるアルゴリズムとは？さらに受けるアルゴリズムとは？

2.5. 頻出集合列挙問題と Apriori アルゴリズム

大規模なデータから法則や関係を見出すことは，一般にデータマイニンングと総称され，コンピュータネットワーク時代の応用とも関連して，ここ数年，コンピュータサイエンスの活発な研究分野になっている．その発端となったのが，今回紹介する頻出集合列挙問題とそれに対する単純だが効率のよい Apriori アルゴリズムである．

まず，問題を定義するために，データベースに関連する用語をざっと紹介しよう.（今後の議論で使う記号も合わせて導入しておく.）

E: アイテム (item) の全体集合.
X: アイテム集合 (item set)，つまり E の部分集合.
T: トランザクション（データベースに保存されている履歴） (transaction).
（実態はアイテム集合と同じで，E の部分集合.）
$\mathcal{T}$: トランザクションの全体.
$\mathcal{T}(X) \stackrel{\text{def}}{=} \{\, T \in \mathcal{T} | X \subseteq T \,\}$: X を含むトランザクションの集合.
$\mathcal{F}_\sigma \stackrel{\text{def}}{=} \{\, X \,|\, |\mathcal{T}(X)| \geq \sigma \,\}$: 閾値 σ に対する頻出集合（以下では σ は省略することが多い）.

<u>頻出集合列挙問題</u>
入力： σ とデータベース $\mathcal{T}$.
仕事： 頻出集合に $\mathcal{F}_\sigma$ に属する集合の列挙.

Agrawal と Srikant [1] は次のような非常に単純なアルゴリズムを提案した.

Algorithm Apriori
  input σ: integer; $\mathcal{T}$: database;
    $\mathcal{F}_1 = \{\, \{e\} \,|\, |\mathcal{T}(\{e\})| \geq \sigma \,\}$; $\mathcal{F}_1$ を出力; （$\mathcal{F}_1 = \emptyset$ ならば終了）
    for $i = 2$ **to** ∞ {
      for each $X \in \mathcal{F}_{i-1}$ and each $\{e\} \in \mathcal{F}_1$ {
        if $|\mathcal{T}(X) \cap \mathcal{T}(\{e\})| \geq \sigma$ **then** $\mathcal{F}_i$ に $X \cup \{e\}$ を追加;
      }
      $\mathcal{F}_i$ を出力; （ただし $\mathcal{F}_i = \emptyset$ ならば終了）
    }
end.

このアルゴリズムの時間計算量を簡単に評価しよう．とくに凝った工夫をしない場合を考え[1]$\mathcal{F}_1$ と $\mathcal{T}(\mathcal{F}_1)$ $(= \{\, \mathcal{T}(X) \mid X \in \mathcal{F}_1 \,\})$ と，各 i において，$\mathcal{F}_i$ と $\mathcal{T}(\mathcal{F}_i)$ を保持しておいたとすると，

$$\text{Apriori アルゴリズムの計算量} \;=\; O\,(\,|\mathcal{T}|\,) + \sum_{i=2}^{k} O\,(\,|\mathcal{F}_{i-1}| \cdot |\mathcal{F}_1| \cdot \tilde{\sigma}_{i-1},)$$

程度になる．ただし，$\tilde{\sigma}_{i-1}$ は，$\mathcal{F}_{i-1}$ 中のアイテム集合 X に対する $|\mathcal{T}(X)|$ の最大値．また，k は最もアイテムの多い頻出集合のアイテム数である．

さて，最初にも述べたように，このアルゴリズムは非常に有名になり，データマイニングの先駆けの一つとなった．今から[2]考えると，次の二点が重要だったように思う．

- 上記の計算量は見た目にはそう小さくないが，$\mathcal{T}$ を何度も探索することに比べて $\mathcal{F}_1$ や $\mathcal{F}_i$ が非常に小さくなるため，大幅な効率化が得れらた．また，k がそう大きくならなかった点もよかった．
- そういう性質（たち）の良いアルゴリズムで，しかも十分役に立つものを提案できるような「良い問題」を定式化した．

なお，Apriori アルゴリズムは実現の仕方にもよるが，$\mathcal{F}_i$ をとっておかなければならない点（スペースが必要だという点で），我々が今回考えている列挙アルゴリズムとは言えない．また，そもそも $\mathcal{F}$ が非常に大きくなる可能性がある．次ではこれらの問題を考えてみよう．

2.6. よりよい列挙問題＆よりよい列挙アルゴリズム

閾値 σ によっては頻出集合の総数は非常に多くなる．それらをすべて列挙しないでも，ある程度重要な情報は得られないだろうか？たとえば $\mathcal{T}(X)$ が同じならば，もっとも大きい X を代表とみなしてもよいのではないか？その発想から出たのが飽和集合 (closed item set) である．

トランザクション集合 $\mathcal{S} \subseteq \mathcal{T}$ に対して，$I(\mathcal{S}) = \bigcap_{T \in \mathcal{S}} T$ と定義する．飽和集合とは $I(\mathcal{T}(X)) = X$ となる集合である．ここでは，頻出集合の中でも飽和集合になっているものを代表としてみなし，その列挙アルゴリズムについて考えよう．以下では $\mathcal{C}$ で（与えられた閾値 σ に対して）頻出集合であり，しかも飽和集合になっているもの全体を表わすことにする．

この頻出飽和集合を効率的に列挙するアルゴリズムを提案したのが [2] である．このアルゴリズムは逆探索による列挙アルゴリズムのよい例でもある．今回は，逆探索の基本となる探索木の導入とアルゴリズムのあらすじを紹介しよう．

---

[1]実際には，最初の提案論文 [1] でもいくつかの工夫版が提案されている．また，この論文以降，数多くの改良案が提案されてきている．

[2]もっと早く生まれていればなぁ，と思う人もいるだろう．でも高々 20 年弱前の話しである．次の [2] にいたっては，10 年もたっていない．今でもまだまだ先駆けとなれるような話は出尽くしていないはずだ．急げ！

再びいくつか概念と記法を導入する．まず，E の要素間にはある順序があり，すべての要素はその順で番号付けされているものとする．さらに，以下ではその番号と要素を同一視する．

$tail(X) = x_n$ （ただし，$X = \{x_1, ..., x_n\}$ で $x_1 < \cdots < x_n$ とする）．
$X(i) = X \cap \{1, ..., i\}$（ただし，$X(0) = \emptyset$ とする）．
X is a prefix of $Y \iff X = Y(i)$, where $i = tail(X)$.
$i(X) =$ the minimum $i \in X$ such that $\mathcal{T}(X) = \mathcal{T}(X(i))$.
（したがって $\mathcal{T}(X) \neq \mathcal{T}(X(i(X) - 1))$.）

鍵となるのは頻出飽和集合間の親子関係である．

定義 2.3. 頻出飽和集合間 $X \in \mathcal{C}$ に対して，その親 $\mathcal{P}(X)$ を

$$\mathcal{P}(X) \;=\; I(\,\mathcal{T}(X(i(X) - 1))\,)$$

と定義する．また逆に，X を $\mathcal{P}(X)$ の子供と呼ぶ．なお，$I(\mathcal{T}(\emptyset))$ を $\perp$ と表し，これを根と呼ぶことにする．

補題 2.4. (1) 任意の（頻出）アイテム集合 X に対して $I(\mathcal{T}(X))$ は（頻出）飽和集合．
(2) 任意の頻出飽和集合 $X \in \mathcal{C}$（ただし $\perp$ は除く）に対して，$\mathcal{P}(X)$ は一意に定まる．また，$\mathcal{P}(X)$ も頻出飽和集合である．

与えられた頻出飽和集合 X に対し，その子供も比較的簡単に求めることができる．以下の補題がその条件を規定している．以下では，アイテム集合 X と $i \in E$ に対して，

$$\begin{aligned} X[i] \;&=\; X \cup H, \\ &\quad\; \text{where } H = \{\, j \in I(\mathcal{T}(X \cup \{i\})) \mid j \geq i \,\} \end{aligned}$$

と定義する[3]．

補題 2.5. 任意の $X \in \mathcal{C}$ に対して，次の条件は X' が X の子供（でかつ $X' \in \mathcal{C}$）となっているための必要十分条件である．
(c1) $X' = X[i]$ for some i such that $i > i(X)$ and $i \notin X$,
(c2) $X' = I(\mathcal{T}(X'))$，かつ
(c3) X' は頻出集合である．

[3]宇野氏から教えて頂いたのだが，最近の研究で $X[i] = I(\mathcal{T}(X \cup \{i\}))$ としてもよいことがわかったそうだ．その方が以下の議論がすっきりする．ない，その場合には，次の補題の (c2) は「(c2') $X(i-1) = X'(i-1)$」となる．

この条件を利用すると，次ページのような $\mathcal{C}$ の列挙アルゴリズムを設計することができる．これは根から始めて再帰的に子供の頻出飽和集合を ちょうど１度だけ 出力するアルゴリズムになっている．この性質は，各頻出飽和集合に対して，親がただ一人となるように親子関係を整理したために得られるもので，この考え方により重複を避けて列挙する方法は 逆探索 (reverse search) と呼ばれている[4].

procedure LCM(X: frequent closed item set) {
  X を出力;
  for each i such that $i > i(X)$ & $i \notin X$ {
    if $X[i]$ is frequent & $X[i] = I(\mathcal{T}(X[i]))$ **then** LCM($X[i]$);
  }
}

参考文献

[1] R. Agrawal and R. Srikant, Fast algorithms for mining association rules in large databases, in *Proc. 20h Intn'l Conference on Very Large Data Bases*, 487–499, 1994.

[2] T. Uno, T. Asai, Y. Uchida, and H. Arimura, LCM: An efficient algorithm for enumerating frequent closed item sets, in *Proc. Workshop on Frequent Itemset Mining Implementations 2003*, available as RPI CS Dept. Technical Report TR 03-14, 2003.

[3] D. Avis and K. Fukuda, Reverse search for enumeration, *Discrete Applied Mathematics*, Vol.65, 21–46, 1996.

テーマ＃２での課題（〆切：原則として６月１０日（金））

1. 今回紹介した飽和頻出集合列挙アルゴリズム（できれば次回紹介する予定のさらなる改良も）を実際に実現し，ある程度の大きさのデータベースを用いて，その効率を検討してみよう．たとえば， Apriori アルゴリズムと比較する，あるいは，改良を入れた場合の効率向上の度合いを調べる，などがあると思う.（コメント：ソースコードの提出は不要ですが，プログラミングに際して工夫した点や考えなければならなかった点，どういう実験をしたか，その結果，などの説明は必須です.）

2. 逆探索法による凸多面体の端点列挙のアルゴリズムを調べ，その解説を書く．具体例などを入れてわかりやすい解説にして欲しい．計算量についても解説すること.（コメント：非退化の仮定のもとでの列挙で構いません.）

3. 平面上の与えられた点の集合 $P = \{v_1, ..., v_n\}$ に対して，それを含む最小の凸多角形を P の凸包という．その凸包を P の点だけからなる三角形で分割することを P の凸包の三角形分割という．P の凸包の三角形分割の列挙を逆探索によって行うアルゴリズム

[4]それを提唱したのが当時東工大にいた福田公明氏 [3]．今回紹介した論文 [2] の宇野氏は福田氏の弟子筋にあたる．逆探索は東工大の伝統芸なのである．

を調べ，その解説を書く．具体例などを入れてわかりやすい解説にして欲しい．計算量についても解説すること.（コメント：上の端点列挙の課題に比べて少し難しいので（うまく解説されていれば）点数は高くなります.）