Min-hash 法の拡張を用いた ストリームモデルにおける L_0 ノルム計算

数理情報学専攻 48156219　豊岡祥
指導教員　定兼邦彦教授

1　ストリームモデル

ストリームモデルは大規模かつ高速に生じるデータを処理するための計算モデルである．ストリームモデルは形式的には次のように定義される: 入力として $(i_1, a_1), (i_2, a_2), \ldots, (i_m, a_m)$ が与えられるとする．ここで $i_t \in \{i, 2, \ldots, n\}$, $a_t \in \{-r, -r+1, \ldots, r\}$ とし, n は扱うベクトルのサイズ, r は更新値の上限である．この入力ストリーム (i_t, a_t) は「時刻 t においてベクトルの第 i_t 成分が a_t だけ更新された」と解釈される．すなわちストリームモデルにおいては, (初期状態で零ベクトルである) ベクトル $x \in \mathbb{Z}^n$ を考え, ストリーム入力 (i_t, a_t) $(t = 1, 2, \ldots, m)$ が x を

$$x_{i_t} = x_{i_t} + a, \quad x_j = x_j \ (j \neq i_t)$$

として更新するものと定義する (図 1)．このベクトル x を, ストリーム $(i_1, a_1), \ldots, (i_m, a_m)$ が与える**累積ベクトル**と呼ぶ．いま, ストリーム入力としては n, m が非常に大きい場合を扱うので, この累積ベクトル x は明示的に持つことができない．ストリームアルゴリズムの目的は, 限られた計算領域のみを用いて, x に関する情報を計算することである．多くの場合 $O(\log n)$ ビットの計算領域で動作するアルゴリズムが求められることになる．例えば, x の非零成分数やノルムなどを限られた計算領域で計算するアルゴリズムが盛んに研究されている [1, 2, 3, 4]．ストリームモデルはその入力の取りうる値の

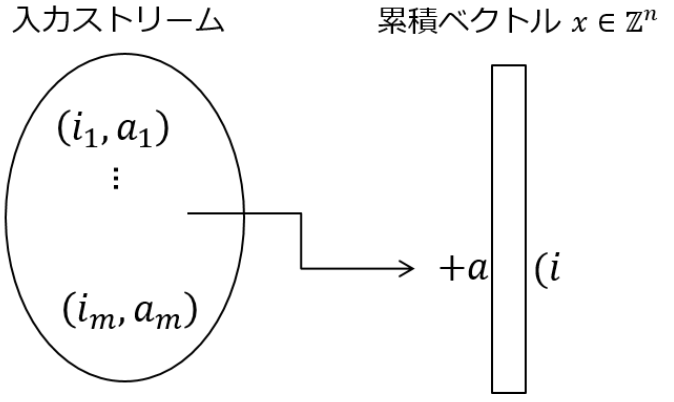

図 1. ストリームモデル

範囲によって 3 種類に分けることができる．

1. Cash register model:　更新値 a が常に正である．
2. Strict turnstile model:　更新値 a が負の値を取ることを許すが, 累積ベクトルの成分 x_i の値は常に非負である．
3. Non-strict turnstile model:　更新値 a も累積ベクトルの成分 x_i も負の値を取ることを許す．

1. はデータが入力される一方のモデルで, 2,3 がデータの削除を許すモデルと解釈できる．後者のモデルほど一般性が高いことに注意されたい．

ストリームモデルの応用例としてネットワーク管理が挙げられる．入力ストリームの (i, a) において i を IP アドレス, a をそのアドレスが行った通信量とすると, 累積ベクトル x の第 i 成分 x_i はアドレス i が行った総通信量ということになる．また, x の非零成分やノルムは通信を行ったアドレスの総数や, その分布に関する情報を与える．

2　研究の流れ

本研究では, ストリームモデルにおける累積ベクトル x の L_0 ノルム (非零成分数) を求める問題を扱う．この問題はモデルを 1. Cash register model に限ったとき, 入力中の異なる i の種類をカウントする問題に帰着する ($a > 0$ ならば一度現れた i に対して必ず $x_i > 0$ となるためである)．そこで本研究では種類カウント問題のアルゴリズムである (次節で述べる)**LogLog counting** [3] というアルゴリズムを 3. Non-strict turnstile model において動作するように改良し, L_0 ノルムを求めるアルゴリズムを提案する．LogLog counting は **Min-hash 法**と呼ばれる方法を用いたアルゴリズムの一種で, これは入力ストリーム上でハッシュ関数の最大値を求めることによって動作する．本研究ではこの Min-hash 法を 3. Non-strict turnstile model で動作するように拡張することによって LogLog counting の改良を行う．

3　先行研究: LogLog counting

[3] は入力ストリーム $i_1, i_2, \ldots, i_m$ の中の異なるアイテムの種類数を推定するアルゴリズムとして, 以下のようなものを与えた:

初期化: ハッシュ関数族 $\mathcal{H} = \{h : [n] \to [\log n]\}$ からハッシュ関数 h をランダムにひとつ選び, 固定する.

更新: ストリーム入力 i に対し $M = \max\{M, h(i)\}$ と更新する.

出力: 上記の計算を $O(\epsilon^{-2})$ 個並行して行い, 得られた M の平均値 $\tilde{M}$ に対して $\alpha 2^{\tilde{M}}$ を返す.

ここで α は適当な比例定数, $\mathcal{H}$ は一様ランダムに h を決めたときに, ハッシュ値が j を取る確率が $1/2^j$ となるようなハッシュ関数族である. 求めたい種類数を x とおくと, このような方法で求められた M は期待値が $\log x$ になり, 分散が定数で抑えられることが示すことができる. LogLog counting はハッシュ関数 h の入力ストリーム上での最大値を求めることによって動作している. このような方法はストリームアルゴリズムの設計においてしばしば用いられる手法であり, **Min-hash 法**と呼ばれる.

4 Min-hash 法の拡張

LogLog counting は Cash register model においてハッシュ関数の最大値を求めることによってアイテムの種類数 ($=L_0$ ノルム) を求めた. したがって Lo g Log counting を拡張してより一般のモデルである Non-strict turnstile model において L_0 ノルムを求めるには, Non-strict turnstile model においてハッシュ関数の最大値を求めることができれば良いことになる. 本研究ではこの問題, すなわち, 入力ストリームが与える累積ベクトルの非零成分を持つインデックス i 上での $h(i)$ の最大値 M を求める問題に対して, 以下の 2 つのアルゴリズムを提案した.

提案アルゴリズム 1 $O(N(\log \delta^{-1} + \log\log mr))$ ビットで確率 $1-\delta$ で M を求める乱択アルゴリズム.

提案アルゴリズム 2 $O(\epsilon^{-2} \log N(\log \epsilon^{-1} + \log\log mr))$ ビットで M の $(1 \pm \epsilon)$ 近似を求める乱択近似アルゴリズム.

ここで N はハッシュ関数の値域の大きさである. アルゴリズムは, 各ハッシュ値を管理するようなサイズ N のベクトルを保持し (1), さらにそのベクトルを $\log N$ サイズになるように線形写像によって圧縮する (2) ことによって構成される.

また, 本研究では Non-strict turnstile model においてハッシュ値の最大値を求めるために必要な空間計算量の下限を与えた. 具体的には以下の 2 つを証明した.

1. 乱択アルゴリズムもしくは ($(1 \pm \epsilon)$ 近似を求める) 近似アルゴリズムは $\Omega(N)$ ビットを必要とする
2. $(1 \pm \epsilon)$ 近似を定数確率で求める乱択近似アルゴリズムは $\log N$ ビットを必要とする.

この結果から提案アルゴリズムの計算量は理論的下界に近い (それぞれ下界の $\log\log mr$ 倍, $\log \epsilon^{-1} + \log\log mr$ 倍) であることが示された. 計算量下界の証明には, 通信複雑度 [5] という道具を用いた.

5 提案アルゴリズム (L_0 ノルム)

LogLog counting の更新ステップを前節で与えた Min-hash 法の拡張アルゴリズム 1 で置き換えることによって, 最終的に得られるアルゴリズムは次のようになる:

初期化: ハッシュ関数族 $\mathcal{H} = \{h : [n] \to [\log n]\}$ からハッシュ関数 h をランダムにひとつ選び, 固定する.

更新: ストリーム入力 (i, a) に対し Min-hash 法の拡張アルゴリズム (前節の 1) を動作させ, ハッシュ関数の最大値 M を得る.

出力: 上記の計算を $O(\epsilon^{-2})$ 個並行して行い, 得られた M の平均値 $\tilde{M}$ に対して $\alpha 2^{\tilde{M}}$ を返す.

前節の提案アルゴリズム 1 において $\delta = O(\epsilon^2)$ とすれば $O(\epsilon^{-2})$ 個全てのアルゴリズムが正しく M を出力する. したがってこのアルゴリズムは全体で $O(\epsilon^{-2} \log n(\log \epsilon^{-1} + \log\log mr))$ ビットで動作する. このアルゴリズムは空間計算量の観点で従来の最良結果 [4] と一致し, よりシンプルなアルゴリズムの構成を与えるものとなっている.

参考文献

[1] N. Alon, Y. Matias and M. Szegedy, The space complexity of approximating the frequency moments, *Journal of Computer and System Science* (1999) **58(1)**, pp. 137–147.

[2] G. Cormode, M. Datar, P. Indyk and S. Muthukrishnan, Comparing data stream using hamming norms (how to zero in), *IEEE Transactions and Knowledge and Data Engineering* (2003) **15(3)**, pp. 529–540.

[3] M. Durand and P. Flajolet, Loglog counting of large cardinalities (extended abstract), *In Proceedings of ESA* (2003), pp. 605–617.

[4] D. M. Kane, J. Nelson and D. P. Woodruff, An optimal algorithm for the distinct elements problem, *In Proceedings of POD* (2010) pp. 41–52.

[5] E. Kushlevitz and N. Nisan, Communication complexity , *Cambridge University Press*(1997)