1N-06

データストリームに対する効率的な複数連続的集約アルゴリズム

川上 隼† Bou Savong‡ 天笠 俊之‡
†筑波大学情報学群情報科学類 ‡筑波大学計算科学研究センター

1 序論

近年、多くのアプリケーション領域で、リアルタイムなデータの生成が行われている。そこで、連続的なデータストリームに対して効率的かつほぼリアルタイムな分析が求められる。

この分析の手法としてよく使われているのが Sliding Window Aggregation(SWAG)だ。SWAG は任意の Window サイズ:r と Slide サイズ:s が W[r,s]というクエリとして与えられる。そして、データストリームから s 個の新しいデータが来るたびに最新の r 個のデータを集約する。

集約関数には、合計や平均、最大、分散などがある。

SWAG は、株式投資家が生の株価を必要とするのではなく、リアルタイムで独自の分析を必要とする時などに使われる。

しかし、SWAG では一つのストリームデータに対して、複数のクエリを処理するときにスケーラビリティの問題が発生する。

本論文では、SWAG の複数のクエリに対しての効率的な集約アルゴリズムを提案する。クエリの中で、r の最大値と最小値をもとに作る da と qa という二つの配列のみで集約を行い、その結果をすべてのクエリで共有することで、処理時間を大幅に削減する。

2 背景知識

集約の方法として、後方集約と前方集約がある。これは、L-Bix[2]で使われている方法だ。

・後方集約では、新しいレコードから古いレコードの順に集約していく。これを b 値と呼ぶ。window が slide することで期限切れのレコードが必要なくなった時、それを含む b 値は無効となるが、他の b 値は有効なままとなる。

・前方集約では、最初のレコードから新しく到着したレコードまで集約していく。これを f 値と呼ぶ。f 値は新しいレコードが来るたびにそれまでの f 値と集約することで、漸次的に求めることができる。

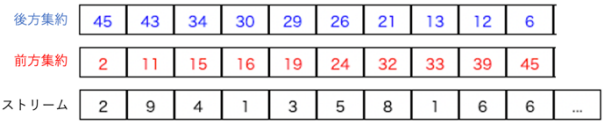


図 1 合計を求める後方集約と前方集約の例

3 提案手法

本論文で提案する手法は、複数のクエリの中で window サイズの最大値と最小値をもとに作る二つの索引付き配列を使って集約する。

一つ目はサイズが最大値となる配列で da と呼ぶ。二つ目は、サイズが最大値と最小値の商を切り上げた値となる配列で qa と呼ぶ。

da は最小値でサイクルが分けられる。例えば、最大値が 38 で最小値が 10 の時は、[10, 10, 10, 8]の 4 つに分けられる。そして、それぞれのサイクルが始まると前方集約が行われ、サイクル内のレコードがすべて更新されたら後方集約が行われる。

qa は da のサイクルに 1 対 1 で対応していて、サイクル内の最大の b 値を後方集約した値を格納する。

この二つの配列から適した値を持ってくることで、すべてのクエリがそれぞれ 3 回以下の集約操作で求めることができる。

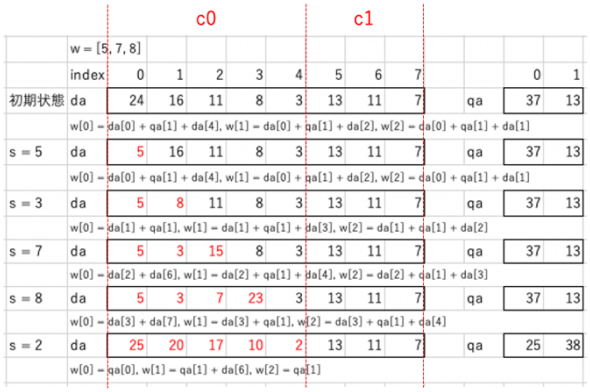
例として、合計を求める W[5,2], W[7,2], W[8,2]と与えられた 3 つのクエリについて提案手法を使って集約していく。

まず、配列を作る。window サイズの最大値より da のサイズは最大値の 8 となる。qa のサイズは 最大値と最小値の商の切り上げより、
[8/5] = [1.6] = 2 と求まる。また、図 2 のように da を最小値より[5, 3]とサイクルに分ける。それぞれの区分を c1, c2 と呼ぶ。

次に集約していく。図 2 を参考にしながら行う。今回は過去のデータで c0 と c1 を後方集約した値が da には格納され、qa には、c0,c1 の最大値を qa[1]から後方集約した値が格納された状態から始める。ここから新しいストリーム:5 が来ると、前方集約が始まり da[0]に 5 が格納される。次の

Efficient Multiple continuous aggregation algorithm for data streams

Shun Kawakami†, Bou Savong‡ and Toshiyuki Amagasa‡
†College of Information Science, University of Tsukuba
‡Center for Computational Sciences, University of Tsukuba



		c0					c1					
	w = [5, 7, 8]											
	index	0	1	2	3	4	5	6	7		0	1
初期状態	da	24	16	11	8	3	13	11	7	qa	37	13
	w[0] = da[0] + qa[1] + da[4], w[1] = da[0] + qa[1] + da[2], w[2] = da[0] + qa[1] + da[1]											
s = 5	da	5	16	11	8	3	13	11	7	qa	37	13
	w[0] = da[0] + qa[1] + da[4], w[1] = da[0] + qa[1] + da[2], w[2] = da[0] + qa[1] + da[1]											
s = 3	da	5	8	11	8	3	13	11	7	qa	37	13
	w[0] = da[1] + qa[1], w[1] = da[1] + qa[1] + da[3], w[2] = da[1] + da[1] + da[2]											
s = 7	da	5	3	15	8	3	13	11	7	qa	37	13
	w[0] = da[2] + da[6], w[1] = da[2] + qa[1] + da[4], w[2] = da[2] + qa[1] + da[3]											
s = 8	da	5	3	7	23	3	13	11	7	qa	37	13
	w[0] = da[3] + da[7], w[1] = da[3] + qa[1], w[2] = da[3] + qa[1] + da[4]											
s = 2	da	25	20	17	10	2	13	11	7	qa	25	38
	w[0] = qa[0], w[1] = qa[1] + da[6], w[2] = qa[1]											

図 2　合計を求める 提案手法の例

ストリーム:3 が来ると、da[0]+3=8 が da[1]に格納される。次のストリーム:7 が来ると、da[1]+7=15 が da[2]に格納され、一つ前のストリームである 3 が da[1]に格納される。次のストリーム:8 が来ると、da[2]+8=23 が da[3]に格納され、ひとつ前のストリームである 7 が da[2]に格納される。次のストリーム:2 が来た時に c1 のサイクル内のすべてが更新されるので、後方集約が始める。まず、da[4]に 2 が格納され、他の値は以下の式で求められる。

$$da[i] = da[i] + da[i+1] \quad (i = 3 \sim 0)$$

そして、qa でも後方集約が行われ、c0 に対応する qa[0]から始まり、qa[0]には c0 の最大値の 25 が格納され、qa[1]には c1 の最大値の 13+と qa[0]を足した 38 が格納される。

これが 1 サイクルの動きとなる。次のストリームが来たら c1 が始まり、c1 が終わったら c0 が始めるというのを繰り返して集約を行う。

クエリの集約結果は図 2 に書いてある通りで、ストリーム:5 が来たときはそれぞれのクエリは

$$W[5,2] = da[0] + qa[1] + da[4]$$
$$W[7,2] = da[0] + qa[1] + da[2]$$
$$W[8,2] = da[0] + qa[1] + da[1]$$

と 2 回以下の集約操作（今回は足し算）で求めることができる。他の場面でも同様に 2 回以下の集約操作で求めることができる。

4　実験

実験では、複数のクエリの SWAG を求める最新のアルゴリズムである MCQA[1]と比較する。

データセットとしては、DEBS を使用した。DEBS は工場内の様々な大規模センサーによって生成されたレコードで、51 のフィールドを持つ。実験では、そのうちの 1 フィールドを使って行った。このデータセットには約 3200 万レコードが含まれる。

実験では、slide サイズを 2 に固定して、window サイズを変化させて行った。window サイズは W = [10, 13, 19, 40]を W, W*2, W*5, W*10, W*20, W*50 と倍率をかけて変化させた。その結果が図 3 になる。提案手法は、window サイズに依存しないため、平均時間はほぼ変わることはない。それに対して、MCQA は、window サイズと slide のサイズの比に相関を持つため、平均時間は window サイズが大きくなるにつれて大きくなっている。そのため、window サイズが大きくなればなるほど、平均時間の差が顕著に現れる結果となった。

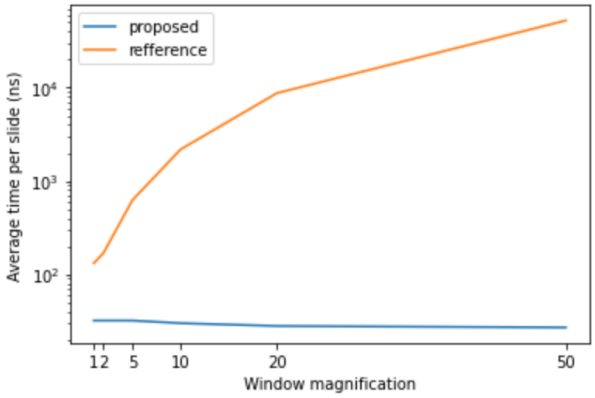


図 3 window を変化させた実験

5　結論

本論文では、da と qa の二つの配列を用いた SWAG の複数のクエリに対しての効率的な集約アルゴリズムを提案した。実験により、最新の手法である MCQA に比べて、提案手法の方が時間を少なく集約できることがわかった。今後は、ペースによる最適な分類について検討する予定である。

参考文献

[1] Wen Liu, Tuqian Zhang, Junxia Liu, “Window-based multiple continuous query algorithm for data streams”, 75:5782-5807, The Journal of Supercomputing, 2019

[2]Savong Bou, Hiroyuki Kitagawa, Toshiyuki Amagasaa, “L-Bix: incremental sliding-window aggreagation over data streams using linear bidirectional aggregating indexes”, 62:3107-3131, Knowledge and information System, 2020