(Optimistic?) Concurrency Control for Non-monotone Submodular Maximization

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Distributed non-monotone submodular maximization, serially equivalent to sequential algorithm.

1 Introduction

2 Submodular maximization

3 Algorithm

3.1 Sequential

Algorithm 1: Serial submodular maximization

$A^0 = \emptyset$, $B^0 = V$
for $i = 1$ *to* n **do**
    $\Delta_+(i) = [F(A^{i-1} \cup i) - F(A^{i-1})]_+$
    $\Delta_-(i) = [F(B^{i-1} \backslash i) - F(B^{i-1})]_+$
    Draw $u_i \sim Unif(0, 1)$
    if $u_i < \frac{\Delta_+(i)}{\Delta_+(i)+\Delta_-(i)}$ **then**
        $A^i := A^{i-1} \cup i$; $B^i := B^{i-1}$
    else
        $A^i := A^{i-1}$; $B^i := B^{i-1} \backslash i$

Algorithm 2: Validate(i,u_i)

Input: A^{i-1}, B^{i-1}
$\Delta_+(i) = [F(A^{i-1} \cup i) - F(A^{i-1})]_+$
$\Delta_-(i) = [F(B^{i-1} \backslash i) - F(B^{i-1})]_+$
if $u_i < \frac{\Delta_+(i)}{\Delta_+(i)+\Delta_-(i)}$ **then**
    $A^i := A^{i-1} \cup i$; $B^i := B^{i-1}$
else
    $A^i := A^{i-1}$; $B^i := B^{i-1} \backslash i$

Algorithm 3: Parallel processing of element i

Input: A^j, B^j, where $i' < i$
$C^{ji} = \{i' + 1, \ldots, i - 1\}$
$\Delta_+^{\max}(i) = [F(A^j \cup i) - F(A^j)]_+$
$\Delta_+^{\min}(i) = [F(A^j \cup C^{ji} \cup i) - F(A^j \cup C^{ji})]_+$
$\Delta_-^{\max}(i) = [F(B^j \backslash i) - F(B^j)]_+$
$\Delta_-^{\min}(i) = [F(B^j \backslash C^{ji} \backslash i) - F(B^j \backslash C^{ji})]_+$
Draw $u_i \sim Unif(0, 1)$
if $u_i < \frac{\Delta_+^{\min}(i)}{\Delta_+^{\min}(i)+\Delta_-^{\max}(i)}$ **then**
    $A^i := A^{i-1} \cup i$; $B^i := B^{i-1}$
else if $u_i > \frac{\Delta_+^{\max}(i)}{\Delta_+^{\max}(i)+\Delta_-^{\min}(i)}$ **then**
    $A^i := A^{i-1}$; $B^i := B^{i-1} \backslash i$
else
    Validate(i,u_i)

Figure 1: The serial submodular maximization algorithm and distributed implementation.

The sequential algorithm monotonically grows A^i and shrinks B^i.

3.2 Distributed

We assume a total ordering on the elements, without loss of generality, let the ordering be $1, 2, \ldots, n$. Let j be such that $j < i$, and $C^{ji} = \{j+1, \ldots, i-1\}$. The properties of the (sequential) algorithm ensures that

$$\begin{aligned} A^j \subseteq & A^{i-1} \subseteq A^j \cup C^{ji}, \\ B^j \backslash C^{ji} \subseteq & B^{i-1} \subseteq B^j. \end{aligned}$$

If we let $D^k = \{k' : k+1 \leq k' \leq n\}$, it is easy to see that $B^j = A^j \cup D^j = A^j \cup C^{ji} \cup i \cup D^i$ and $B^j \backslash C^{ji} = A^j \cup i \cup D^i$. We define

$$\begin{aligned} \Delta_+^{\min}(i) &= [F(A^j \cup C^{ji} \cup i) - F(A^j \cup C^{ji})]_+ \\ \Delta_+(i) &= [F(A^{i-1} \cup i) - F(A^{i-1})]_+ \\ \Delta_+^{\max}(i) &= [F(A^j \cup i) - F(A^j)]_+ \\ \Delta_-^{\min}(i) &= [F(B^j \backslash C^{ji} \backslash i) - F(B^j \backslash C^{ji})]_+ \\ &= [F(A^j \cup D^i) - F(A^j \cup D^i \cup i)]_+ \\ \Delta_-(i) &= [F(B^{i-1} \backslash i) - F(B^{i-1})]_+ \\ &= [F(A^{i-1} \cup D^i) - F(A^{i-1} \cup D^i \cup i)]_+ \\ \Delta_-^{\max}(i) &= [F(B^j \backslash i) - F(B^j)]_+ \\ &= [F(A^j \cup D^j \backslash i) - F(A^j \cup D^j)]_+ \\ &= [F(A^j \cup C^{ji} \cup D^i) - F(A^j \cup C^{ji} \cup D^i \cup i)]_+ \end{aligned}$$

Submodularity of F implies that

$$\begin{array}{ccccc} \Delta_+^{\min}(i) & \leq & \Delta_+(i) & \leq & \Delta_+^{\max}(i), \\ \Delta_-^{\min}(i) & \leq & \Delta_-(i) & \leq & \Delta_-^{\max}(i), \end{array}$$

so we can bound

$$lb(i) \;:=\; \frac{\Delta_+^{\min}(i)}{\Delta_+^{\min}(i) + \Delta_-^{\max}(i)} \;\leq\; \frac{\Delta_+(i)}{\Delta_+(i) + \Delta_-(i)} \;\leq\; \frac{\Delta_+^{\max}(i)}{\Delta_+^{\max}(i) + \Delta_-^{\min}(i)} \;=:\; ub(i).$$

Thus, if $u_i \leq lb(i)$, we can safely grow $A^i = A^{i-1} \cup i$. Conversely, if $u_i \geq ub(i)$, we can safely shrink $B^i = B^{i-1} \backslash i$. (In practice, growing A or shrinking B is done implicitly by setting an indicator variable.)

4 Distributed function computation

To make our concurrent submodular maximization algorithm work, it is essential that the computation of Δ's can be done efficiently. We maintain a *sketch* for A, which contains all essential information for computing Δ's, and update the sketch as elements are added to A. Similarly, we maintain sketches for D^i's; since the sets D^i's are known at the start of the algorithm, we can pre-compute the sketches in advance.

We'll show in the examples below that updating the sketch for A and pre-computing the sketch for D can be done efficiently, and that we can compute the Δ's easily from our sketches.

[XP: See Stefanie's write-up for greater exposition on sketching functions]

4.1 Graph cut

$F(A) = \sum_{i \in A} \sum_{j \in V \backslash A, (i,j) \in E} w(i,j)$. The sketch for A is simply A itself, and is maintained by each processor. The sketch for D^j is the single number j.

4.2 Separable sums

$F(A) = \sum_{l=1}^{k} g\left(\sum_{i \in A \cap S_l} w(i)\right) - \lambda \sum_{i \in A} v(i)$. The sketch for A is the $k+1$ vector containing the sums $\sum_{i \in A \cap S_l} w(i)$ and $\sum_{i \in A} v(i)$. Similarly for D. Updating A involves adding $w(i)$ and $v(i)$ to the sums for A. Pre-computing the sketch for D requires computing $k+1$ cumulative sums, one for each S_l, of length N.

5 Upper bound on expected number of elements sent for validation

Let N be the number of elements, i.e. the cardinality of the ground set. Let P be the number of processors.

We assume that the total ordering assigns elements to processors in a round robin fashion. Thus, we assume $C^{ji} = \{i-p+1, \ldots, i-1\}$ has $p-1$ elements.

We call element i *dependent* on i' if $\exists A, F(A \cup i) - F(A) \neq F(A \cup i' \cup i) - F(A \cup i')$ or $\exists B, F(B \backslash i) - F(B) \neq F(B \cup i' \backslash i) - F(B \cup i')$, i.e. the result of the transaction on i' will affect the computation of Δ's for i. For example, for the graph cut problem, every vertex is dependent on its neighbors; for the separable sums problem, i is dependent on $\{i' : \exists S_l, i \in S_l, i' \in S_l\}$.

Let n_i be the number of elements that i is dependent on.

Now, we note that if C^{ji} does not contain any elements on which i is dependent, then $\Delta_+^{\max}(i) = \Delta_+(i) = \Delta_+^{\min}(i)$ and $\Delta_-^{\max}(i) = \Delta_-(i) = \Delta_-^{\min}(i)$, so i will not be validated (in either deterministic or probabilistic versions). Conversely, if i is validated, there must be some element $i' \in C^{ji}$ such that i is dependent on i'.

$$
\begin{aligned}
&E(\text{number of validated elements}) \\
=&\sum_i P(i \text{ validated}) \\
\leq&\sum_i P(\exists i' \in C^{ji}, i \text{ depends on } i') \\
=&\sum_i 1 - P(\forall i' \in C^{ji}, i \text{ does not depend on } i') \\
=&\sum_i 1 - \prod_{k=1}^{P-1} \frac{N-k-n_i}{N-k} \\
=&\sum_i 1 - \prod_{k=1}^{P-1} \left(1 - \frac{n_i}{N-k}\right) \\
\leq&\sum_i 1 - \left(1 - \sum_{k=1}^{P-1} \frac{n_i}{N-k}\right) && \text{(Weierstrass inequality)} \\
=&\left(\sum_i n_i\right)\left(\sum_{k=1}^{P-1} \frac{1}{N-k}\right) \\
\leq&\frac{P-1}{N-P+1} \sum_i n_i.
\end{aligned}
$$

The key quantity in the above inequality is $\sum_i n_i$. Typically, we expect each element i to depend on a small fraction of the ground set. For example, in the graph cut problem, $\sum_i n_i = 2|E|$ is twice the number of edges. If the graph is sparse with $|E| \approx s|V| \log |V|$, where $0 \leq s \ll 1$ and $P \ll N$, then $\frac{P-1}{N-P+1} \sum_i n_i \approx 2s(P-1) \log N$, which grows sublinearly with N.

Note that the bound established above is generic to all algorithms that follow the basic transactional model we proposed (round-robin optimistic concurrency control), and is not specific to F or even submodular maximization. Thus, while our bounds provide a fundamental limit, additional knowledge of F can lead to better analyses on the algorithm's concurrency.

5.1 Tighter general bound?

Define $\rho_i = \max_{S \subseteq V} \{[F(S \cup i) - F(S)] - [F(S \cup C^{ji} \cup i) - F(S \cup C^{ji})]\} \leq F(i) - F(V) + F(V \backslash i)$

[XP: Is there theory along these lines?]

Then, we can bound

$$\begin{aligned} \Delta_+^{\min} \leq \Delta_+^{\max} \leq \Delta_+^{\min} + \rho_i & \qquad \text{(choosing } S = A^j\text{)} \\ \Delta_-^{\min} \leq \Delta_-^{\max} \leq \Delta_-^{\min} + \rho_i & \qquad \text{(choosing } S = A^j \cup D^i\text{)} \end{aligned}$$

Thus,

$$\begin{aligned} & E(\text{number of validated elements}) \\ =& \sum_i P(i \text{ validated}) \\ =& \sum_i P\left(\frac{\Delta_+^{\min}}{\Delta_+^{\min} + \Delta_-^{\max}} \leq u_i \leq \frac{\Delta_+^{\max}}{\Delta_+^{\max} + \Delta_-^{\min}} \right) \\ =& \sum_i \frac{\Delta_+^{\max}}{\Delta_+^{\max} + \Delta_-^{\min}} - \frac{\Delta_+^{\min}}{\Delta_+^{\min} + \Delta_-^{\max}} \\ \leq& \sum_i \frac{\Delta_+^{\min} + \rho_i}{\Delta_+^{\min} + \rho_i + \Delta_-^{\min}} - \frac{\Delta_+^{\min}}{\Delta_+^{\min} + \rho_i + \Delta_-^{\min}} \\ =& \sum_i \frac{\rho_i}{\Delta_+^{\min} + \rho_i + \Delta_-^{\min}} \end{aligned}$$

5.2 Upper bound for max graph cut

Denote $\tilde{A}^j = V \backslash A^j \backslash C^{ji} \backslash D^i = \{1, \ldots, j\} \backslash A^j$ be the elements up to j that are not included in A. Let $w_i(S) = \sum_{j \in S, (i,j) \in E} w(i,j)$. For the max graph cut function, it is easy to see that

$$\begin{aligned} \Delta_+^{\min} &= \max(0, -w_i(A^j) - w_i(C^{ji}) + w_i(D^i) + w_i(\tilde{A}^j)) \\ \Delta_+^{\max} &= \max(0, -w_i(A^j) + w_i(C^{ji}) + w_i(D^i) + w_i(\tilde{A}^j)) \\ \Delta_-^{\min} &= \max(0, +w_i(A^j) - w_i(C^{ji}) + w_i(D^i) - w_i(\tilde{A}^j)) \\ \Delta_-^{\max} &= \max(0, +w_i(A^j) + w_i(C^{ji}) + w_i(D^i) - w_i(\tilde{A}^j)) \end{aligned}$$

Consider the following cases.

- $\Delta_+^{\max} = 0$. Then $\Delta_+^{\min} = 0$ and also

  $$w_i(A^j) > w_i(C^{ji}) + w_i(D^i) + w_i(\tilde{A}^j) \quad \Longrightarrow \quad w_i(A^j) + w_i(D^i) > w_i(C^{ji}) + w_i(\tilde{A}^j)$$

  so $\Delta_-^{\min} > 0$ and $\Delta_-^{\max} > 0$. Thus $\frac{\Delta_+^{\max}}{\Delta_+^{\max} + \Delta_-^{\min}} - \frac{\Delta_+^{\min}}{\Delta_+^{\min} + \Delta_-^{\max}} = 0 - 0 = 0$.

- $\Delta_-^{\max} = 0$. Then $\Delta_-^{\min} = 0$ and also

  $$w_i(\tilde{A}^j) > w_i(C^{ji}) + w_i(D^i) + w_i(A^j) \quad \Longrightarrow \quad w_i(\tilde{A}^j) + w_i(D^i) > w_i(C^{ji}) + w_i(A^j)$$

  so $\Delta_+^{\min} > 0$ and $\Delta_+^{\max} > 0$. Thus $\frac{\Delta_+^{\max}}{\Delta_+^{\max} + \Delta_-^{\min}} - \frac{\Delta_+^{\min}}{\Delta_+^{\min} + \Delta_-^{\max}} = 1 - 1 = 0$.

- $\Delta_+^{\max} > 0$ and $\Delta_-^{\max} > 0$. Then,

$$
\begin{aligned}
&\frac{\Delta_+^{\max}}{\Delta_+^{\max}+\Delta_-^{\min}} - \frac{\Delta_+^{\min}}{\Delta_+^{\min}+\Delta_-^{\max}} \\
=&\frac{-w_i(A^j)+w_i(C^{ji})+w_i(D^i)+w_i(\tilde{A}^j)}{-w_i(A^j)+w_i(C^{ji})+w_i(D^i)+w_i(\tilde{A}^j)+\max(0,+w_i(A^j)-w_i(C^{ji})+w_i(D^i)-w_i(\tilde{A}^j))} \\
&-\frac{\max(0,-w_i(A^j)-w_i(C^{ji})+w_i(D^i)+w_i(\tilde{A}^j))}{\max(0,-w_i(A^j)-w_i(C^{ji})+w_i(D^i)+w_i(\tilde{A}^j))+w_i(A^j)+w_i(C^{ji})+w_i(D^i)-w_i(\tilde{A}^j)} \\
=&\min\left(1, \frac{-w_i(A^j)+w_i(C^{ji})+w_i(D^i)+w_i(\tilde{A}^j)}{2w_i(D^i)}\right) \\
&-\max\left(0, \frac{-w_i(A^j)-w_i(C^{ji})+w_i(D^i)+w_i(\tilde{A}^j)}{2w_i(D^i)}\right) \\
=&\min\left(1, \frac{w_i(C^{ji})}{w_i(D^i)}\right)
\end{aligned}
$$

Thus,

$$
E(\text{\# of validated elements}) = \sum_i \frac{\Delta_+^{\max}}{\Delta_+^{\max}+\Delta_-^{\min}} - \frac{\Delta_+^{\min}}{\Delta_+^{\min}+\Delta_-^{\max}} \le \sum_i \min\left(1, \frac{w_i(C^{ji})}{w_i(D^i)}\right)
$$

[XP: Not sure how to sum this over i.]

$$
\sum_\pi \sum_i \min(1, w_i(C)/w(D^i)) \le E(\sum_i w_i(C)) = c * \sum_i deg(i)/n
$$