Sample and Prune: An Efficient MapReduce Method for Submodular Optimization

Benjamin Moseley
Washington University in St. Louis
MapReduce Class [Karloff et al.]

- $N$ is the input size
- Sublinear $(N^{1-\epsilon})$ memory on each machine
- Sublinear $(N^{1-\epsilon})$ number of machines
- Mappers/reducers are $\text{poly}(N)$ computable
- $\mathcal{MRC}^0$: algorithms that run in $O(1)$ rounds
Algorithmic Design in MapReduce

- No one machine can see the entire input
- No communication between machines during a phase
- Total memory is large
Sampling

- Sample the input in parallel
- Well represent the input space
- Generally must be done adaptively
- Sample should be small
Monotone Submodular Function Maximization

- Universe of elements \( U \) where \( |U| = n \)
- A function \( f : 2^U \rightarrow \mathbb{R} \)
- Function is submodular and monotone

\[ \forall Y, X \subseteq U \text{ where } X \subseteq Y \text{ and every } x \in U \setminus Y \text{ we have} \]
\[ f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y) \]
Monotone Submodular Function Maximization

- Find a set $S$ such that
  $$\max_S f(S)$$
- Possible constraints (hereditary)
  - Cardinality
  - Matroid (system)
  - Knapsack
- Maximum solution size $k$
- Memory $\Omega(kn^c)$
Maximum Coverage

Users ———> Interests ———> Ads

- Star Wars
- Beach scene
- Car
- Book: "Breakfast of Champions"
Submodular Function Maximization

- Maximum submodular coverage
- Minimum spanning tree
- Maximum matching in bipartite graphs
- ...
Greedy Algorithm

- \( Y = \emptyset \)
- Add elements sequentially with the most value

\[
\max_x f(\{x\} \cup Y)
\]
Simulation of the Greedy Algorithm

- A natural approach

- Partition data across machines

- $U_i$ data given to machine $i$

- Run greedy on each machine to get a set $S_i$

- Map all sets to a single machine and run greedy on $\bigcup_i S_i$
Bad Example

$k\{\begin{array}{c}
\text{copies} \\
2k\end{array}\}^k$

$k\{\begin{array}{c}
\text{copies} \\
2k\end{array}\}^k$
Bad Example

Algorithm’s Objective \( k \)
Optimal Objective \( 2k - 1 \)
Greedy in MR

[Kumar-M-Vassilvitskii-Vattani]

- $v$ maximum value an element can give

- For submodular objectives, a $O(\log v)$ round algorithm that computes the greedy solution

- $(1 - \frac{1}{e} - \epsilon)$ -approximation for maximum submodular coverage

- $\frac{1}{3 + \epsilon}$ -approximation for weighted matching
Approximate Greedy Algorithm

- \( Y = \emptyset \)
- Add elements sequentially with \((\text{almost})\) the most value

\[
  f(\{x\} \cup Y) - f(Y) \geq \frac{1}{(1 + \epsilon')}(f(\{a\} \cup Y) - f(Y)) \quad \forall a \in U
\]
Algorithm Overview

- Solution set \( Y = \emptyset \)
- Find maximum additional value possible
  \[ v = \max_x f(\{x\}) \]
- In \( j \)th phase pick elements until
  \[ \max_x \{ f(\{x\} \cup Y) - f(Y) \} < \frac{v}{(1 + \epsilon')^i} \]
- Logarithmic rounds:
  \[ O(\log_{1+\epsilon'}(v)) \]
Analysis

• Solution the same as the approximate sequential greedy algorithm’s solution

• Only choose elements that could be chosen by the greedy algorithm
Sampling

- Sample $\tilde{\Theta}(kn^\epsilon)$ elements to get a set $S$
- Pick elements of value greater than $\frac{v}{(1 + \epsilon')^i}$
- $Y_i$ current solution
- Remove all elements $x$ with value less than $\frac{v}{(1 + \epsilon')^i}$
- Recurse until all points are removed
Analysis

- A factor of roughly $n^\epsilon$ elements removed each iteration
- $\mathcal{E}_{Y_i}$: Event that the solution is $Y_i$
- High value elements would change the solution if sampled

$$\Pr[\mathcal{E}_{Y_i}] \leq \left(1 - \frac{k}{n^{1-\epsilon}}\right)^{2n^{1-\epsilon}\log n} \leq e^{-2k\log n} = \frac{1}{n^{2k}}$$
Analysis

• Same guarantees as the greedy algorithm
• Number of rounds is \( O\left(\frac{1}{\epsilon} \log_{1+\epsilon'}(v)\right) \)
• Memory: \( \tilde{\Theta}(kn^\epsilon) \)
• Machines: \( \tilde{\Theta}(n^{1-\epsilon}) \)
Extensions

- Constant round algorithms?
- Tailor the sampling algorithm
  - Ensure only non-important elements are discarded
Extensions

• Constant round algorithms!
  • Optimal algorithm for a modular function subject to one matroid
  • Approximation for cardinality constraint
  • Approximation for $d$ matroids
  • Approximation for $d$ knapsacks
Streaming Algorithm for the Cardinality Constraint

• Select a set $S$ of size $k$ to maximize $f(S)$
• Let $OPT$ denote the optimal solution
• Streaming Algorithm
  • Set $S = \emptyset$
  • Consider elements in any order $e_1, e_2, \ldots, e_n$
  • Add $e_i$ to $S$ if $f(S \cup \{e_i\}) - f(S) \geq \frac{OPT}{2k}$
Analysis

• \( \frac{1}{2} \)-approximation

• Say \( k \) elements are added to \( S \)

• Each gave incremental value at least \( \frac{OPT}{2k} \)

• Total value is at least \( k \cdot \frac{OPT}{2k} = \frac{OPT}{2} \)
Analysis

- Say $|S| < k$
- Let $S^*$ denote the optimal solution
- For elements $e \in S^*$ it is the case that
  
  $f(S \cup \{e\}) - f(S) < \frac{OPT}{2k}$

- Thus, we have

  $f(S \cup S^*) - f(S) \leq \frac{OPT}{2k}|S^* \setminus S|$
Analysis

- We know:
  \[ f(S \cup S^*) - f(S) \leq \frac{OPT}{2k} |S^* \setminus S| \]

- Which implies
  \[ f(S \cup S^*) - \frac{OPT}{2k} |S^* \setminus S| \leq f(S) \]

- Using monotonicity
  \[ OPT = f(S^*) \leq f(S \cup S^*) \]

- Putting this together
  \[ OPT - \frac{OPT}{2} \leq f(S) \]
MapReduce: Sampling

- Set $Y = \emptyset$
- Sample $\tilde{\Theta}(kn^e)$ elements
- Pick elements of value greater than $\frac{OPT}{2k}$
- $Y_i$ current solution
- Remove all elements $x$ with value less than $\frac{OPT}{2k}$
- Recurse until all points are removed
Analysis

• A factor of roughly $n^\epsilon$ elements removed each iteration

• $\mathcal{E}_{Y_i}$: Event that the solution is $Y_i$

• High value elements would change the solution if sampled

$$\Pr[\mathcal{E}_{Y_i}] \leq (1 - \frac{k}{n^{1-\epsilon}})^{2n^{1-\epsilon}\log n} \leq e^{-2k \log n} = \frac{1}{n^{2k}}$$
Analysis

• $\frac{1}{2}$-approximation
• Number of rounds is $O\left(\frac{1}{\epsilon}\right)$
• Memory: $\tilde{\Theta}(kn^\epsilon)$
• Machines: $\tilde{\Theta}(n^{1-\epsilon})$
Abstracting the Idea: Sample-and-Prune

- Let $\mathcal{G}$ be an algorithm which
  - Returns a set of size at most $k$
  - $\mathcal{G}(A) \subseteq \mathcal{G}(B)$ for all $A \subseteq B$
Abstracting the Idea: Sample-and-Prune

- Sample $\tilde{\Theta}(kn^\epsilon)$ elements to get a set $S$
- Set $Y = G(S)$
- Remove all elements $x$ where

$$x \notin G(Y \cup \{x\})$$
Abstracting the Idea: Sample-and-Prune

- Key theorem
  - An $n^\epsilon$ fraction of the elements are removed with high probability
Application: Maximal Matching

- Maximal matching
  - No two edges incident to the same node
  - No edge can be added to the solution
Application: Maximal Matching

• Sequential algorithm
  • Consider edges in an arbitrary order
  • Add an edge if it is feasible

• Let this algorithm be $G$
  • $G(A) \subseteq G(B)$ for all $A \subseteq B$
  • Only discards edges incident to chosen edges
Conclusion

- Algorithmic techniques for MR
- Sampling
- Greedy algorithms
Thank You!
Questions?