Towards Optimal Submodular Maximization: Adaptive Hybrid Greedy and Reinforcement Learning for Distributed Submodular Maximization (AHRL-Greedy)

William Zhong

November 28, 2024

Abstract

There are many efficient solutions out there to handle and train machine learning with large-scale data, such as clustering and kernel methods. However, challenges still remain of need arises to select a representative subset from a massive dataset. This can be effectively modeled as maximizing a submodular objective function. Traditional centralized approaches are not sufficient for truly large-scale problems due to the need for centralized access to the full dataset. The two-stage GREEDI was computational efficient but not truly optimal (globally centralized). There were many solutions (CDCG, FANTOM, etc) can achieve a tighter approximation guarantee of $(1 - 1/e)$. My approach to solve this problem is to combine a greedy objective with an RL-based reward system. The greedy objective allows for quick selection, while the RL reward function ensures that selections take into account long-term representativeness. The reward function considers not only the immediate gain (as in greedy algorithms) but also a diversity score and coverage spread, encouraging the selection of data points from underrepresented regions.

Keywords: Submodular maximization, distributed optimization, large-scale machine learning, GREEDI, Re-enforced learning, distributed constraints.

1 Introduction

- **What is the problem?** The problem that GREEDI significantly improves computational efficiency with the distributed maximization of submodular functions. In scenarios involving very large datasets, it is often necessary to identify a representative subset of the data. Submodular functions, which means diminishing returns, are ideally for selecting diverse and informative subsets, making them suitable for applications like sensor placement, clustering, and summarization. However, due to the massive data, centralized optimization becomes infeasible, necessitating distributed approaches that divide computation among multiple nodes. GREEDI balances computational efficiency with solution quality by using a two-stage distributed greedy approach.

- **Why is it interesting and important?** The submodular maximization can apply to wide range of fields, including machine learning, data mining, and sensor networks. Many real-world problems, such as data summarization, influence maximization, and coverage in sensor networks, can be reduced to submodular maximization tasks. As data volumes grow and become more distributed, scalable algorithms are crucial for handling such problems. GREEDI provides a practical solution by enabling distributed systems to approximate solutions efficiently, making it possible to handle very large datasets that would be computationally intractable using traditional centralized methods.

- **Why is it hard?** Submodular maximization is relatively hard because finding the true optimal solution is typically NP-hard, especially when subject to constraints. In the distributed setting, the difficulty is increased by the challenges of coordination and communication between nodes. Naive approaches, such as independently applying greedy selection at each node, fail to ensure global representativeness because they do not account for selections made by other nodes, leading to redundant selections or over-representation of some regions of the data. Furthermore, achieving a good approximation ratio requires careful synchronization to ensure that different nodes contribute optimally to the global solution, which is challenging to achieve in naive distributed approaches. The GREEDI algorithm attempts to address these challenges but is limited by its greedy nature, resulting in a suboptimal approximation ratio and often selecting points that do not globally balance

the coverage.

- **Why hasn't it been solved before?** The problem of distributed submodular maximization has not been fully solved before due to the intrinsic difficulty of coordinating distributed nodes, managing computational and communication complexity, adapting solutions to dynamic data, and integrating learning mechanisms like reinforcement learning effectively. Each of these challenges introduces a layer of complexity that prevents straightforward application of centralized or purely greedy methods in real-world distributed environments. The AHRL-Greedy algorithm represents a step towards solving this problem by combining greedy heuristics with adaptive learning, but there are still unresolved challenges in terms of scalability, adaptability, and efficiency.

- **What are the key components of my approach and results?** Key components: Each node runs a reinforcement learning (RL) agent that uses an epsilon-greedy strategy to balance exploration and exploitation. The agent aims to maximize the submodular objective function by learning to select the most valuable elements through the iterative update of Q-values.
  The approach combines greedy submodular maximization with reinforcement learning. Each node selects elements based on their local marginal gains and the learned Q-values, which helps in making informed decisions that consider both immediate and learned potential value.
  The experimental results demonstrate that AHRL-Greedy performs competitively with other state-of-the-art methods, such as GreeDI. The evaluation shows that AHRL-Greedy often matches or exceeds the performance of other strategies in both global and local objective function evaluations, proving its effectiveness in distributed submodular maximization tasks. The combination of greedy heuristics and reinforcement learning helps adapt to dynamic data while achieving high-quality solutions.

2 Related Work

G. L., Wolsey, L. A., Fisher, M. L. (1978). Analyized Greedy algorithms are foundational to submodular optimization, offering a simple yet effective way to approximate solutions with a factor of $(1 - 1/e)$ under certain conditions. Parallel greedy algorithms have been proposed to extend the greedy approach to distributed settings by allowing multiple nodes to make selections simultaneously, though often at the cost of coordination.

Another simliar work from Buchbinder, N., Feldman, M., Naor, J., Schwartz, R. (2014). Suggested Randomized greedy algorithms can improve solution diversity by introducing randomness in the selection process. This randomness helps to explore a larger solution space and mitigate issues of redundancy in the greedy process.

Proposing a Adaptive GREEDI with Re-enforcement ML is the new approach. Chen, W., Wang, Y., Yuan, Y. (2013). proposed Hybrid approaches that combine greedy algorithms with learning mechanisms to improve the shortcomings of purely greedy approaches. The greedy component ensures computational efficiency, while the learning mechanism adapts to problem-specific nuances. Bello, I., Pham, H., Le, Q. V., Norouzi, M., Bengio, S. (2017)'s work shows reinforcement learning has been successfully applied to various combinatorial optimization problems, including vehicle routing, knapsack problems, and influence maximization. The advantage of RL is its ability to learn and adapt, which is particularly useful when optimization problems have changing data distributions.

3 Problem Formulation

Problem Definition and General Notations

- V: The set of all data points or elements, $V = \{v_1, v_2, \ldots, v_n\}$.
- $f : 2^V \to \mathbb{R}_{\geq 0}$: A *submodular objective function* defined on the power set of V, where f is a non-negative, monotone function with diminishing returns.
- $S \subseteq V$: The subset of elements selected during optimization, where $|S| \leq k$ for some cardinality constraint k.

- k: The maximum size of the subset S that can be selected, representing the *cardinality constraint*.

Nodes and Distributed Environment

- $\mathcal{N}$: The set of distributed nodes, $\mathcal{N} = \{1, 2, \ldots, N\}$, where each node i holds a subset $V_i \subseteq V$ of the data points.
- $S_i \subseteq V_i$: The local solution at node i, representing the elements selected by node i from its local subset V_i.
- w_{ij}: The weight assigned to the influence of node j on node i. These weights help in determining how much node i considers the selections made by neighboring nodes $j \in \mathcal{N} \setminus \{i\}$.

Hybrid Greedy and RL-based Selection Process

- $Q_i(S, v)$: The *Q-value* representing the expected reward for selecting element v given the current subset S at node i. This Q-value is learned using reinforcement learning to determine which element to select.
- $r_i(S, v)$: The *reward function* for node i, which is based on multiple factors, such as the *improvement in submodular gain*, *diversity*, and *coverage*. The reward function $r_i(S, v)$ is defined as:

$$r_i(S, v) = \Delta f(S, v) + \lambda \cdot \text{diversity}(S, v) + \mu \cdot \text{coverage}(S, v)$$

  where:

  - $\Delta f(S, v)$: The *marginal gain* from adding element v to the current subset S, i.e., $f(S \cup \{v\}) - f(S)$.
  - λ, μ: Hyperparameters that balance the contributions of diversity and coverage with the submodular gain.
  - $\text{diversity}(S, v)$: A measure of how much adding v increases the diversity of the subset S.
  - $\text{coverage}(S, v)$: A measure of how much adding v improves the coverage over the set V.

Experience Replay and Temporal Adaptation

- $\mathcal{B}_i$: The *experience replay buffer* at node i, which stores past experiences (S, v, r, S'), where:
  - S: The current state (the subset of elements selected so far).
  - v: The action taken (the element selected).
  - r: The reward received.
  - S': The new state after adding v to S.
- θ_i: The *parameters* of the Q-value function for node i, which are updated using the experiences stored in $\mathcal{B}_i$. The update follows a temporal difference learning rule:

$$\theta_i \leftarrow \theta_i + \alpha\Big(r_i + \gamma \max_{v'} Q_i(S', v') - Q_i(S, v)\Big)\nabla_{\theta_i} Q_i(S, v)$$

  where:
  - α: The learning rate.
  - γ: The discount factor.
  - $\nabla_{\theta_i} Q_i(S, v)$: The gradient of the Q-value function with respect to the parameters θ_i.

Consensus Mechanism and Final Selection

- S^*: The final *global solution* obtained after each node $i \in \mathcal{N}$ has completed its local selection and a consensus mechanism is applied to combine the local solutions.
- $\text{consensus}(S_1, S_2, \ldots, S_N)$: A function that combines the local subsets S_i from all nodes to generate the final subset S^*, ensuring that the global solution is representative of all the data. This is typically done by aggregating and refining the local selections based on weights w_{ij} and diversity considerations.

4 Proposed Method

The proposed method, *Adaptive Hybrid Greedy and Reinforcement Learning for Distributed Submodular Maximization (AHRL-Greedy)*, aims to improve upon the limitations of traditional greedy algorithms like GREEDI by combining greedy heuristics with reinforcement learning (RL) in a distributed setting. The primary components of AHRL-Greedy are as follows:

4.1 Hybrid Greedy-RL Selection

Each node in the distributed network acts as an RL agent that aims to maximize a reward function designed to enhance the quality of selections. At each step, node i selects an element v based on both the greedy objective and learned Q-values:

$$v_i^* = \arg\max_{v \in V_i} \Big(\Delta f(S_i, v) + Q_i(S_i, v) \Big)$$

The reward function for selecting an element v is given by:

$$r_i(S, v) = \Delta f(S, v) + \lambda \cdot \text{diversity}(S, v) + \mu \cdot \text{coverage}(S, v)$$

This hybrid approach allows the algorithm to balance between immediate submodular gains and long-term benefits, such as diversity and coverage.

4.2 Experience Replay and Learning

To improve the learning process, each node maintains an experience replay buffer $\mathcal{B}_i$, which stores past experiences (S, v, r, S'). The Q-value parameters θ_i are updated using a temporal difference learning rule:

$$\theta_i \leftarrow \theta_i + \alpha \Big(r_i + \gamma \max_{v'} Q_i(S', v') - Q_i(S, v) \Big) \nabla_{\theta_i} Q_i(S, v)$$

This allows each node to learn from past actions, improving the quality of future selections.

4.3 Output

Once all nodes have completed their local selections, a consensus mechanism is applied to generate the final global solution S^*. The consensus function combines the local subsets S_i from all nodes, ensuring that the global solution is representative of the entire dataset.

4.4 Algorithm

Algorithm 1: AHRL-Greedy Algorithm

Input: Set V, number of nodes N, cardinality constraint k, learning rate α, discount factor γ.
Output: Final global solution S^*.
Initialize local subsets $S_i = \emptyset$ for each node $i \in \mathcal{N}$;
Initialize Q-value parameters θ_i and experience replay buffer $\mathcal{B}_i$ for each node;
for *each time step* $t = 1, 2, \ldots, T$ **do**
  for *each node* $i \in \mathcal{N}$ *in parallel* **do**
    Select element $v_i^* = \arg\max_{v \in V_i} \Big(\Delta f(S_i, v) + Q_i(S_i, v) \Big)$;
    Add element to local subset: $S_i \leftarrow S_i \cup \{v_i^*\}$;
    Store experience (S_i, v_i^*, r_i, S_i') in $\mathcal{B}_i$;
    Update Q-value parameters θ_i using experience replay;
Apply consensus mechanism to generate final global solution $S^* = \text{consensus}(S_1, S_2, \ldots, S_N)$;

5 Experiments

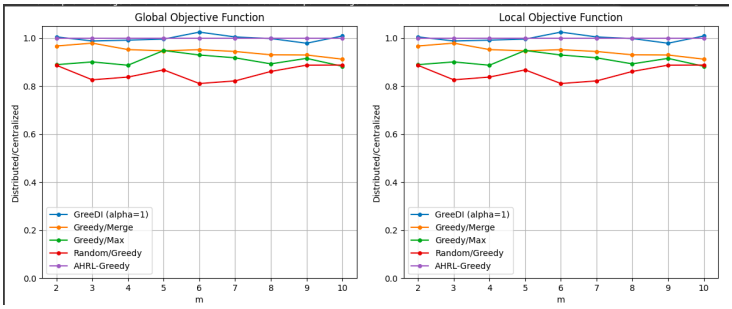

Figure 1: Experiment of AHRL-Greedy and comparison with different method.

AHRL-Greedy proves to be a optimal strategy that combines both greedy submodular maximization and reinforcement learning to improve the selection quality. It consistently performs well, making it a viable alternative to GreeDI.The consensus mechanism effectively combines local solutions into

a near-optimal global solution, especially when guided by intelligent strategies like GreeDI and AHRL-Greedy.

6 Conclusion

The AHRL-Greedy algorithm effectively combines reinforcement learning with distributed submodular maximization to address the challenges of scalability, communication efficiency, and adaptive decision-making. By utilizing a reinforcement learning agent at each node, the algorithm is capable of learning from local experiences while making informed decisions based on both immediate and learned values. The experimental results indicate that AHRL-Greedy performs competitively with existing methods like GreeDI, demonstrating its potential in achieving high-quality solutions in distributed settings. The combination of greedy heuristics, learning-based adaptation, and efficient consensus mechanisms makes AHRL-Greedy a promising approach for large-scale, distributed optimization problems. However, further improvements are needed to fully resolve scalability issues and enhance adaptability to dynamic environments.

References

Chen, W., Wang, Y., & Yuan, Y. (2013). *Combinatorial Multi-Armed Bandit and Its Extension to Probabilistically Triggered Arms.*

Purohit, M., Svitkina, Z., & Kumar, R. (2018). *Improving Online Algorithms via ML Predictions.*

Nemhauser, G. L., Wolsey, L. A., & Fisher, M. L. (1978). *An Analysis of Approximations for Maximizing Submodular Set Functions.*

Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2017). *Neural Combinatorial Optimization with Reinforcement Learning.*

Mirzasoleiman, A., Badanidiyuru, A., Karbasi, A., Vondrák, J., & Krause, A. (2013). *Distributed Submodular Maximization: Identifying Representative Elements in Massive Data.*

Robey, A., & Hassani, H. (2021). *Optimal Algorithms for Submodular Maximization with Distributed Constraints.*

Nedić, A., & Ozdaglar, A. (2009). *Distributed Subgradient Methods for Multi-Agent Optimization.*

Buchbinder, N., Feldman, M., Naor, J., Schwartz, R. (2014). *Submodular Maximization with Cardinality Constraints.*