Probabilistic Submodular Maximization in Sub-Linear Time

Serban Stan [1] Morteza Zadimoghaddam [2] Andreas Krause [3] Amin Karbasi [1]

Abstract

In this paper, we consider optimizing submodular functions that are drawn from some unknown distribution. This setting arises, e.g., in recommender systems, where the utility of a subset of items may depend on a user-specific submodular utility function. In modern applications, the ground set of items is often so large that even the widely used (lazy) greedy algorithm is not efficient enough. As a remedy, we introduce the problem of *sublinear time probabilistic submodular maximization*: Given training examples of functions (e.g., via user feature vectors), we seek to reduce the ground set so that optimizing new functions drawn from the same distribution will provide almost as much value when restricted to the reduced ground set as when using the full set.

We cast this problem as a two-stage submodular maximization and develop a novel efficient algorithm for this problem which offers a $\frac{1}{2}(1-\frac{1}{e^2})$ approximation ratio for general monotone submodular functions and general matroid constraints. We demonstrate the effectiveness of our approach on several real-world problem instances where running the maximization problem on the reduced ground set leads to two folds speed-up while incurring almost no loss.

1. Introduction

Motivated by applications in data summarization (Lin & Bilmes, 2011; Wei et al., 2013; Mirzasoleiman et al., 2016d) and recommender systems (El-Arini et al., 2009; Yue & Guestrin, 2011; Mirzasoleiman et al., 2016a), we tackle the challenge of efficiently solving many statistically related submodular maximization problems. In these applications, submodularity arises in the form of user-specific utility functions for valuating sets of items, and a prototypical problem is to find sets of say k items with near-maximal value (Krause & Golovin, 2012; Mirzasoleiman et al., 2016b). Even though efficient greedy algorithms exist for submodular maximization, those become infeasible when serving many users and optimizing over large item collections. To this end, given training examples of (users and their utility) functions drawn from some unknown distribution, we seek to reduce the ground set to a small (ideally *sublinear*) size. The hope is that optimizing new functions drawn from the same distribution will incur little loss when optimized on the reduced set compared to optimizing over the full set.

Optimizing the empirical objective is an instance of *two-stage submodular maximization*, a problem recently considered by Balkanski et al. (2016) who provide a 0.316 approximation guarantee for the case of coverage functions (more about it in the related work). One of our key technical contributions is a computationally efficient novel local-search based algorithm, called Replacement-Greedy, for two-stage submodular maximization, that provides a constant-factor 0.432 approximation guarantee for the general case of monotone submodular functions. Our approach also generalizes to arbitrary matroid constraints, and empirically compares favorably to prior work. We further analyze conditions under which our approach provably enables approximate submodular optimization based on substantially reduced ground sets, resulting the first viable approach towards sublinear-time submodular maximization.

Lastly, we demonstrate the effectiveness of our approach on recommender systems for which we compare the utility value and running time of maximizing a submodular function over the full data set and its reduced version returned by our algorithm. We consistently observe that the loss we incur is negligible (around 1%) while the speed up is enormous (about two orders of magnitude).

Related work Submodular maximization has found many applications in machine learning, ranging from feature/variable selection (Krause & Guestrin, 2005) to dictionary learning (Das & Kempe, 2011) to data summarization (Wei et al., 2014b; Lin & Bilmes, 2011; Tschiatschek et al., 2014) to recommender systems (Yue & Guestrin,

[1] Yale University, New Haven, Connecticut, USA [2] Google Research, New York, NY 10011, USA [3] ETH Zurich, Zurich, Switzerland. Correspondence to: Amin Karbasi <amin.karbasi@yale.edu>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

2011; El-Arini et al., 2009). A seminal result of Nemhauser et al. (1978) proves that a simple greedy algorithm provides an optimal constant factor $(1 - 1/e)$ approximation guarantee. Given the size of modern data sets, much work has focused on solving submodular maximization at scale. This work ranges from accelerating the greedy algorithm itself (Minoux, 1978; Mirzasoleiman et al., 2015; 2016a; Badanidiyuru & Jan, 2014; Buchbinder et al., 2014) to distributed (Kumar et al., 2013; Mirzasoleiman et al., 2013) and streaming (Krause & Gomes, 2010; Badanidiyuru et al., 2014) approaches, as well as algorithms based on filtering the ground set in multiple stages (Wei et al., 2014a; Feldman et al., 2017). All of these approaches aim to solve a *single* fixed problem instance, and have computational cost at least linear in the ground set size. In contrast, we seek to solve multiple related problems with *sublinear* effort.

Solving multiple submodular problems arises in online submodular optimization (Streeter & Golovin, 2008; Hazan & Kale, 2009; Jegelka & Bilmes, 2011). In this setting the goal is to design algorithms that perform well in hindsight (minimize some form of regret). The computational complexity of these algorithms is typically (super)-linear in the ground set size. Studying online variants of our problem is an interesting direction for future work.

Closest to our work is the approach of Balkanski et al. (2016) that we build on and compare within this paper. For general submodular objectives, they propose two algorithms: one based on continuous optimization (with prohibitive computational complexity in terms of the size of the ground set n), which provides constant factor approximation guarantees only for large values of k (i.e., cardinality constraint), as well as one that has exponential complexity in terms of k. To circumvent the large computational complexity of the above algorithms, they also proposed a heuristic local search method that offers a $\frac{1}{2}(1-\frac{1}{e})$ approximation for the special case of coverage functions. The query complexity of this algorithm is $O(km\ell n^2 \log n)$ where ℓ is the size of the summary and m is the number of considered submodular functions in the two-stage optimization. One of our key contributions is a novel and computationally efficient algorithm for the two-stage problem. More specifically, our method ReplacementGreedy provides a $\frac{1}{2}(1 - e^{-2})$ approximation guarantee for general monotone submodular functions subject to a general matroid constraint with only $O(rm\ell n)$ query complexity (here r denotes the rank of the matroid). As argued by Balkanski et al. (2016), two-stage submodular maximization can be seen as a discrete analogue of representation learning problems like dictionary learning. It is important to note that the two-stage submodular maximization problem is fractionally subadditive (XOS) (Feige, 2009). Although it is tempting to use the XOS property of our two-stage function especially given the positive results for social welfare XOS maximization, there are several obstacles preventing us from doing so. First, evaluating the two stage function is NP-hard, so we cannot have access to oracle value queries. Second, as shown by Singer (2010); Shahar Dobzinski & Schapira (2005); Ashwinkumar Badanidiyuru (2012), any $n^{1/2-\epsilon}$ approximation of a XOS requires exponentially many oracle value queries. Therefore the XOS property itself is not sufficient to get any positive algorithmic result for our problem. Nevertheless, we can still provide constant factor approximation with computationally efficient algorithms.

Repeatedly optimizing related classes of submodular functions is a key subroutine in many applications, such as structured prediction (Lin & Bilmes, 2012) or linear submodular bandits (Yue & Guestrin, 2011). In both of these problems, one needs to repeatedly maximize weighted combinations of submodular functions, for changing weights. Our work can be viewed as providing an approach towards accelerating this central subroutine.

2. Problem Setup

In this paper, we consider the problem of frequently optimizing monotone submodular functions $f : 2^{\Omega} \to \mathbb{R}_+$ that are drawn from some unknown probability distribution $\mathbb{D}$. Hereby, Ω denotes the ground set of size n over which the submodular functions are defined. W.l.o.g. we assume that the maximum value of any function f drawn from $\mathbb{D}$ does not exceed 1. This setting arises in many applications, such as recommender systems, where the random function $f = f_u$ refers to the (predicted) valuation over sets of items for a particular user u, which may vary depending on their features (c.f., Yue & Guestrin (2011)). These applications typically dictate some constraints, i.e., one seeks to solve

$$T^* = \arg\max f(T) \text{ s.t. } T \in \mathcal{I},$$

where $\mathcal{I} \subseteq 2^{\Omega}$ is a collection of feasible sets. In this paper we primarily consider cardinality constraints, i.e., $\mathcal{I} = \{T \subseteq \Omega : |T| \leq k\}$. Our results will hold also in the more general setting where $\mathcal{I}$ is the collection of independent sets in some matroid (Calinescu et al., 2011). Throughout the paper r denotes the rank of the matroid. In the special case of cardinality constraint, the rank is $r = k$.

While NP-hard, good approximation algorithms are known for submodular maximization. For example, the classical greedy algorithm of Nemhauser et al. (1978) or its accelerated variants (Minoux, 1978; Mirzasoleiman et al., 2015; 2016a; Badanidiyuru & Jan, 2014; Buchbinder et al., 2014) provide an optimal constant factor $(1-1/e)$ approximation for maximization under cardinality constraints. In modern applications, however, the system may face a large number of users, and large collections of items Ω. Hence the naive strategy of even greedily optimizing f_u for each user separately may be too costly.

To remedy this situation, in this paper we consider the following approach: Given training data (i.e., a sample collection of functions $f_1, \dots, f_m$), we invest computation *once* to obtain a reduced ground set S of size $\ell \ll n = |\Omega|$. The hope is that optimizing new functions arising at test time will provide almost as much value when restricting the choice to items in S, than when considering arbitrary items in Ω, while being substantially more computationally efficient.

More formally, the expected performance when using the candidate reduced ground set S is

$$G(S) = \mathbb{E}_{f \sim \mathbb{D}} \max_{T \in \mathcal{I}(S)} f(T), \tag{1}$$

where we use $\mathcal{I}(S) \equiv \{T \in \mathcal{I} \text{ and } T \subseteq S\}$ to refer to the collection of feasible sets restricted to those containing only elements from S. The optimum achievable performance would be $G(\Omega)$. Our goal will be to pick a set S of small size ℓ to maximize $G(S)$, or equivalently make $|G(\Omega) - G(S)|$ small.

Special cases. Some observations are in order. If $\mathbb{D}$ is deterministic, i.e., puts all mass on a single function f, then we simply recover classical constrained submodular maximization, since $G(S) = f(S)$ for sets up to size k. If $\mathbb{D}$ is known to be the uniform distribution over m functions, $G(S)$ becomes

$$G_m(S) = \frac{1}{m} \sum_{i=1}^{m} \max_{T \in \mathcal{I}(S)} f_i(T). \tag{2}$$

$G_m(S)$ is not generally submodular, but Balkanski et al. (2016) have developed approximation algorithms for maximizing G_m under the constraint that $|S| \le \ell$,[1] i.e., for the problem:

$$S^{m,\ell} = \operatorname*{arg\,max}_{S \subset \Omega, |S| \le \ell} \frac{1}{m} \sum_{i=1}^{m} \max_{T \in \mathcal{I}(S)} f_i(T). \tag{3}$$

The general case. In this paper, we consider the problem of maximizing $G(S)$ for *general* distributions $\mathbb{D}$. I.e., we seek to solve:

$$S^*_\ell = \operatorname*{arg\,max}_{S \subset \Omega, |S| \le \ell} G(S). \tag{4}$$

3. Probabilistic Submodular Maximization

Since the distribution $\mathbb{D}$ is unknown, we cannot find S^*_ℓ or its corresponding optimum value, $G(S^*_\ell)$. Instead, we can sample functions from $\mathbb{D}$ and construct the empirical average, similar to Problem (2), and try to optimize it by finding $S^{m,\ell}$. Hence, the total generalization error we incur in this process is bounded by

$$\begin{aligned} \text{error} &= |G(\Omega) - G(S^{m,\ell})| \\ &\le \underbrace{|G(\Omega) - G(S^*_\ell)|}_{\text{compression error}} + \underbrace{|G(S^*_\ell) - G(S^{m,\ell})|}_{\text{approximation error}}. \end{aligned} \tag{5}$$

Note that once we have error $\le \epsilon$ for some small $\epsilon > 0$ then maximizing over $S^{m,\ell}$ is almost as good as maximizing over Ω (but possibly much faster). To that end we need to bound both compression error and approximation error. In fact we can prove the following result for the required number of samples to ensure small approximation error.

Theorem 1. *For any $\epsilon, \delta > 0$, and any set S of size at most ℓ we can ensure that*

$$\Pr\left(\max_{\substack{S \subset \Omega \\ |S| \le \ell}} |G(S) - G_m(S)| > \epsilon \right) < \delta$$

as long as $m = O((\ell \log(n) + \log(1/\delta))/\epsilon^2)$.

In contrast, the compression error cannot be made arbitrarily small in general as the following example shows.

Bad Example. Suppose $\Omega = [n]$, $m = n$ and $\mathbb{D}$ is the uniform distribution over the functions $f_1, \dots, f_n$, where $f_i(S)$ is 1 if $i \in S$, 0 otherwise. Each f_i is in fact modular. Let $k = 1$. It is easy to see that $G(S) = |S|/n$. Hence to achieve compression error less than ϵ, $|S|$ must be greater than $(1 - \epsilon)n$. In particular for $\epsilon < 1/n$, S must be equal to the full set.

Sufficient conditions for sublinear ℓ. The above example shows that one needs additional structural assumptions. One simple special case arises when the union of the optimal sets for all functions in the support of $\mathbb{D}$ is of size $\ell < n$, i.e.,

$$\left|\{T^* : T^* = \operatorname*{arg\,max}_{T \in \mathcal{I}} f(T) \text{ for some } f \in \text{supp}(\mathbb{D})\}\right| = \ell.$$

I.e., if $\mathbb{D}$ is any distribution over at most m functions, clearly $\ell \ge km$ suffices.

This assumption might be too strong in practice, however. Instead, we will consider another set of assumptions that allow bounding the compression error in the case of cardinality constraints. We assume Ω is endowed with a metric d. This metric is extended to sets of equal size so that for any sets T, T' of equal size, $d(T, T')$ is the weight of a minimal matching of elements in T to elements in T', where the weight of (v, v') for $v \in T$ and $v' \in T'$ is $d(v, v')$. We will assume that any function $f \sim \mathbb{D}$ is L-Lipschitz continuous w.r.t. d, i.e., $|f(T) - f(T')| \le L d(T, T')$ for some constant

[1] Balkanski et al. (2016) only consider cardinality constraints.

L for all sets T and T' of size k. Many natural submodular functions arising in data summarization tasks satisfy this condition, such as exemplar-based clustering and certain log-determinants, see, e.g., (Mirzasoleiman et al., 2016c).

Theorem 2. *Suppose each function $f \sim \mathbb{D}$ is L-Lipschitz continuous in (Ω, d). Then, for any $\epsilon > 0$, the compression error is bounded by ϵ as long as $\ell \geq k n_{\epsilon/2kL}$, where n_δ is the δ-covering number of (Ω, d).*

The proofs of Theorems 1 and 2 are given in the appendix.

4. Algorithm

From the discussions in the previous section, we concluded that under appropriate statistical conditions, we can ensure that the error in Eq. 5 can be made small if we have enough samples dictated by Theorem 1. However, our conclusion heavily relied on the fact that we can find the set $S^{m,\ell}$ in Problem 3. As we noted earlier, the objective function in Problem 3 is not submodular in general (Balkanski et al., 2016), thus the classical greedy algorithm may not provide any approximation guarantees.

Our proposed algorithm ReplacementGreedy works in ℓ rounds where in each round it tries to augment the solution in a particular greedy fashion. It starts out with an empty set $S = \emptyset$, and checks (in each round) whether a new element can be added without violating the matroid constraint (i.e., stay an independent set) or otherwise it can be replaced with an element in the current solution while increasing the value of the objective function. To describe how these decisions are made we need a few definitions. Let

$$\Delta_i(x, A) = f_i(\{x\} \cup A) - f_i(A)$$

denote the marginal gain of adding x to the set A if we consider function f_i. Similarly, we can define the gain of removing an element y and replacing it with x as $\nabla_i(x, y, A) = f_i(\{x\} \cup A \setminus \{y\}) - f_i(A)$. Since f_i is monotone we know that $\Delta_i(x, A) \geq 0$. However, $\nabla_i(x, y, A)$ may or may not be positive. Let us consider $\mathcal{I}(x, A) = \{y \in A : A \cup \{x\} \setminus \{y\} \in \mathcal{I}\}$. This is the set of all elements in A such that if we replace them with x we will not violate the matroid constraint. Then, we define the replacement gain of x w.r.t. a set A as follows:

$$\nabla_i(x, A) = \begin{cases} \Delta_i(x, A) & \text{if } A \cup \{x\} \in \mathcal{I}, \\ \max\{0, \max_{y \in \mathcal{I}(x,A)} \nabla_i(x, y, A)\} & \text{o.w.} \end{cases}$$

In words, $\nabla_i(x, A)$ denotes how much we can increase the value of $f_i(A)$ by either inserting x into A or replacing x with one element of A while keeping A an independent set. Finally, let $\mathsf{Rep}_i(x, A)$ be the element that should be replaced by x to maximize the gain and stay independent. Formally,

$$\mathsf{Rep}_i(x, A) = \begin{cases} \emptyset & \text{if } A \cup \{x\} \in \mathcal{I}, \\ \arg\max_{y \in \mathcal{I}(x,A)} \nabla_i(x, y, A) & \text{o.w.} \end{cases}$$

With the above definitions it is easy to explain how ReplacementGreedy works. At all times, it maintains a solution S and a collection of feasible solutions $T_i \subseteq S$ for each function f_i (all initialized to the empty set in the beginning). In each iteration, it picks the top element x^* from the ground set Ω, based on its total contribution to f_i's, i.e, $\sum_{i=1}^m \nabla_i(x, T_i)$, and updates S. Then ReplacementGreedy checks whether any of T_i's can be augmented. This is done either by simply adding x^* (without violating the matroid constraint) or replacing it with an element from T_i. The condition $\nabla_i(x^*, T_i) > 0$ ensures that such replacement is executed only if the gain is positive.

Why does ReplacementGreedy work? Note that solving $\max_{T \in \mathcal{I}(S)} f_i(T)$ is an $\mathcal{NP}$-hard problem. However, ReplacementGreedy finds and maintains independent sets $T_i \subseteq S$ throughout the course of the algorithm. In fact, the collection $\{S, T_1, \ldots, T_m\}$ lower bounds $G_m(S)$ by $\frac{1}{m} \sum_{i=1}^m f_i(T_i)$. Moreover, each iteration increases the aggregate value $\sum_{i=1}^m f_i(T_i)$ by $\sum_{i=1}^m \nabla_i(x^*, T_i)$. What we show in the following theorem is that after ℓ iterations, the accumulation of those gains reaches a constant factor approximation to the optimum value.

Theorem 3. *In only $O(\ell m n r)$ function evaluations ReplacementGreedy returns a set S of size at most ℓ along with independent sets $T_i \in \mathcal{I}(S)$ such that*

$$G_m(S) \geq \frac{1}{2}\left(1 - \frac{1}{e^2}\right) G_m(S^{m,\ell}).$$

A few comments are in order. Balkanski et al. (2016) proposed an algorithm with $(1 - 1/e)/2$-approximation guarantee for the case where f_i's are coverage functions and the constraint is a uniform matroid. This is achieved by solving (potentially large) linear programs while maintaining $O(k \ell m n^2 \log n)$ function evaluations. In contrast, our result holds for any collection of monotone submodular functions and any matroid constraint. Our approximation guarantee $(1 - e^{-2})/2$ is better than $(1 - 1/e)/2$. Finally, our algorithm is arguably faster in terms of both running time (as it simply runs a modified greedy method) and query complexity (as it is linear in all the parameters).

5. Experiments

In this section, we describe our experimental setting. We offer details on the datasets we used and the baselines we ran ReplacementGreedy against. We first show that ReplacementGreedy is highly efficient (i.e., high utility,

Algorithm 1 ReplacementGreedy

```
$S \leftarrow \emptyset, T_i \leftarrow \emptyset \ (\forall 1 \leq i \leq m)$
for $1 \leq j \leq \ell$ do
    $x^* \leftarrow \arg\max_{x \in \Omega} \sum_{i=1}^{m} \nabla_i(x, T_i)$
    $S \leftarrow S \cup \{x^*\}$
    for all $1 \leq i \leq m$ do
        if $\nabla_i(x^*, T_i) > 0$ then
            $T_i \leftarrow T_i \cup \{x^*\} \backslash \mathsf{Rep}_i(x^*, T_i)$
        end if
    end for
end for
Return sets $S$ and $T_1, T_2, \cdots, T_m$
```

low running time) when solving the two-stage submodular maximization problem on two concrete summarization applications: article summarization and image summarization. We then demonstrate sublinear submodular maximization by showing that ReplacementGreedy can efficiently reduce the dataset while incurring minimum loss. We test the performance of ReplacementGreedy on a movie dataset where movies should be recommended to users with user-specific utility functions.

5.1. Baselines

LocalSearch. This is the main algorithm described in Balkanski et al. (2016)[2]. In our experiments, we use $\epsilon = 0.2$. We initialize S by incrementally picking ℓ elements such that at each step we maximize the sum of marginal gains for the m functions f_i.

GreedySum. It greedily selects ℓ elements while maximizing the submodular function $\hat{F}(S) = \sum_{i=1}^{m} f_i(S)$. To find k elements for each f_i it runs another greedy algorithm.

GreedyMerge. It ideally serves as an upper bound for the objective value, by greedily selecting k elements for each function f_i and returning their union. GreedyMerge can easily violate the cardinality constraint ℓ as its solution can have as many as mk elements.

5.2. Metrics

Objective value. We compare algorithms' solutions S according to the scores $G_m(S)$ for the two-stage (empirical) problem, and $G(S)$ for the sublinear maximization problem.

Loss. For the sublinear maximization problem, we measure the relative loss in performance when using the summary, compared to the full ground set, i.e., report $1 - G(S)/G(\Omega)$.

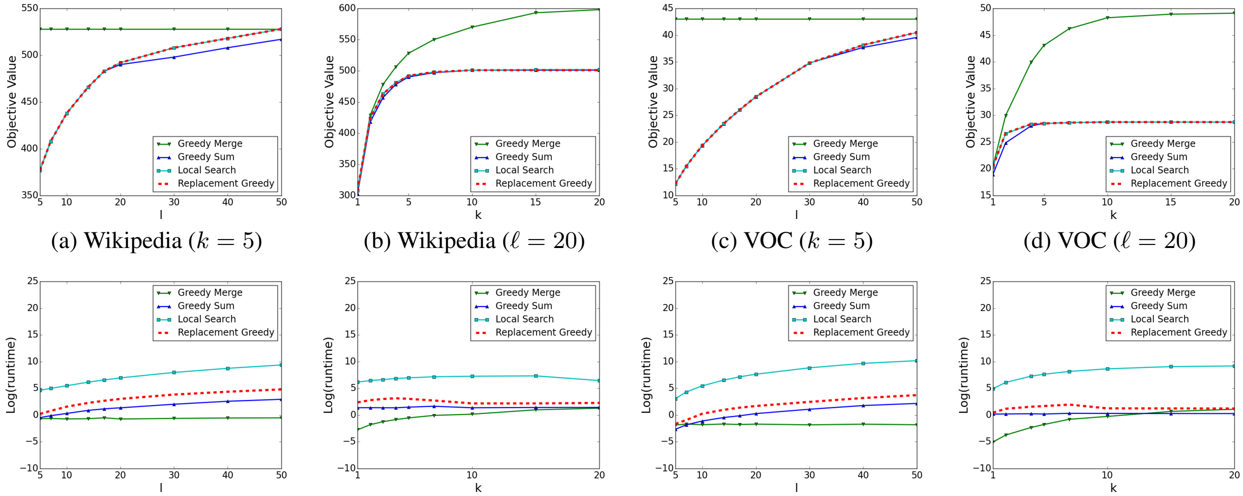
Running time. We also compare the algorithms based on their wall-clock running time. The experiments were ran in a Python 2.7 environment on a OSX 10.12.13 machine. The processor was a 2.5 GHz Intel Core i7 with 16 GB 1600 MHz DDR3 memory.

5.3. Two-Stage Submodular Maximization

Article summarization on Wikipedia. The aim is to select a small, highly relevant subset of wikipedia articles from a larger corpus. For this task, we reproduce the experiment from Balkanski et al. (2016) on Wikipedia articles for *Machine Learning*. The dataset contains $n = 407$ articles divided into $m = 22$ categories, where each category represents a subtopic from Machine Learning. The relevance of a set S with respect to a category i is measured by the submodular function f_i that counts the number of pages that belong to category i with a link to at least one page in S. These submodular functions are L-Lipschitz with $L = 1$ by considering the distance between two articles as the fraction of all pages that have a link to exactly one of these two articles. Fig. 1a and 1b show the objective values for a fix $k = 5$ (and varying ℓ) and a fixed $\ell = 20$ (and varying k). We find that ReplacementGreedy and LocalSearch perform the same while GreedySum falls off somewhat for larger values of ℓ. However, if we look at the running times (log scale) in Fig. 1e and 1f we observe that ReplacementGreedy is considerably faster than LocalSearch and close to GreedySum.

Image summarization on VOC2012. For this application we use a subset of the *VOC2012* dataset (Everingham et al., 2014) where we consider $n = 150$ with $m = 20$ categories. Each category indicates a certain visual queue appearing in the image such as chair, bird, hand, etc. We wish to obtain a subset of these images that are relevant to all the categories. To that end we use Exemplar Based Clustering (c.f., Mirzasoleiman et al. (2013)). Let Ω_i be the portion of the ground set associated to category i. For any set S we also let $S_i = \Omega_i \cap S$ denote its subset that is part of category i. We define $f_i(S) = L_i(\{e_0\}) - L_i(S \cup \{e_0\})$ where $L_i(S) = \frac{1}{|\Omega_i|} \sum_{x \in \Omega_i} \min_{y \in S_i} d(x, y)$. Here, d measures the distance between images (e.g., ℓ_2 norm), and e_0 is an auxiliary element. With respect to distance d, our submodular functions are L-Lipschitz with $L = 1$. Also, images are represented by feature vectors obtained from categories. For example, if there were two categories a and b, and an image had features $[a, a, b]$, its feature vector would be $(2, 1)$. Again if we look at Fig. 1c (for fixed $k = 5$ and varying ℓ) and Fig. 1d (for $\ell = 20$ and varying k) we see that ReplacementGreedy and LocalSearch achieve the same objective value. However, Fig. 1g and 1h show that LocalSearch is significantly slower than ReplacementGreedy.

[2] We thank the authors for providing us with their implementation.



(a) Wikipedia ($k = 5$) (b) Wikipedia ($\ell = 20$) (c) VOC ($k = 5$) (d) VOC ($\ell = 20$)

(e) Wikipedia ($k = 5$) (f) Wikipedia ($\ell = 20$) (g) VOC ($k = 5$) (h) VOC ($\ell = 20$)

Figure 1. Objective values and runtimes for our experiments. The two columns on the left correspond to the Wikipedia dataset and the two columns on the right correspond to the VOC2012 dataset. Note that the runtimes are all in log-scale. We let $k = 5$ and vary ℓ or fix $\ell = 20$ and vary k.

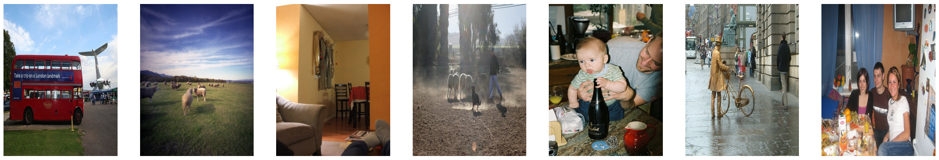


Figure 2. Images chosen by our algorithm on the VOC2012 dataset, for the case when $\ell = 7$ and $k = 5$

5.4. Sublinear Summarization

In this part, we experimentally show how ReplacementGreedy can reduce the size of the dataset to $\ell \ll n$ without incurring too much loss in the process. To do so, we consider a movie recommendation application.

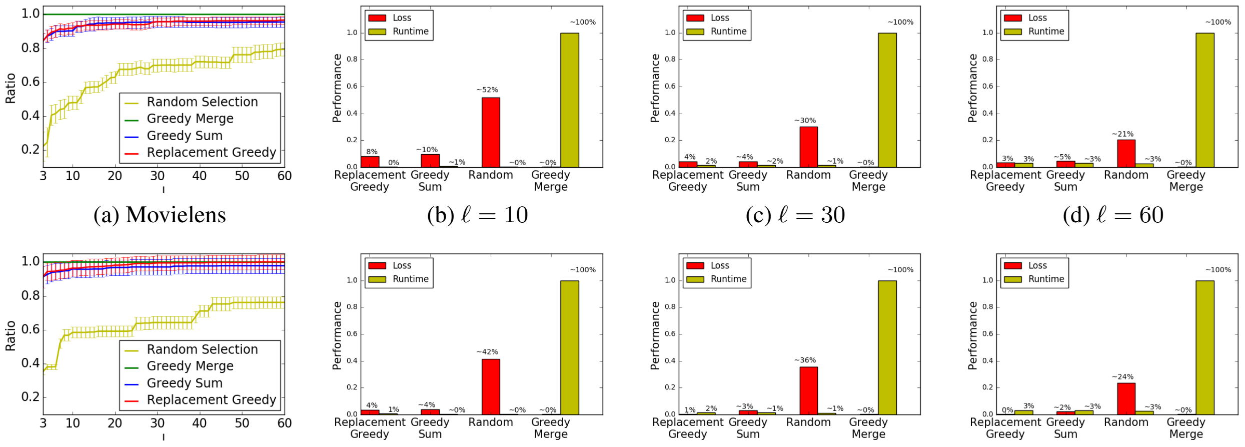
Movie recommendation with missing ratings. The dataset consists of user ratings on a scale of 1 to 5, along with some movie information. There are 20 genres (Animation, Comedy, Thriller, etc) and each movie can have one or more of these genres assigned to it. The goal is to find a small set S of movies with the property that each user will be able to find k enjoyable movies from S.

In our setting we consider the top 2000 highest rated movies (that were rated by at least 20 users) alongside the top 200 users ordered by number of movies they rated from this set. We assign a user-specific utility function f_i to each user i as follows. Let A_i be the subset of A which user i rated. Let g be the number of movie genres. Furthermore, we let $R(i, A, j)$ represent the highest rating given by user i to a movie from the set A with genre j. We also define $w_{i,j}$ to be the proportion of movies in genre j that user i rated out of all the movie-genre ratings she provided. So $w_{i,j}$ will be higher for the genres that the user provided more feedback on, indicating that she might like these genres better. Then, the valuation function by user i is as follows:

$$f_i(A) = \sum_{j=1}^{g} w_{i,j} R(i, A, j).$$

In words, the way a user evaluates a set A is by picking the highest rated movie from each genre (contained in A) and then take a weighted average. If we define the distance between two movies to be the maximum difference of ratings they received from the same user, the submodular functions f_i will be L-Lipschitz with $L = 1$.

For the experiment, we split the users in half. We use the first 100 of them as a training set for the algorithms to built up their reduced ground set S of size ℓ. We then compare submodular maximization with cardinality constraint $k = 3$ on the reduced sets S (returned by the baselines) and that of the whole ground set Ω. To this end, we sample 20 users from the test group and compute their average values. Given the size of this experiment, we were unable to run LocalSearch alongside ReplacementGreedy and GreedySum. In its place, we introduce the random baseline that simply returns a set of size ℓ at random. Fig 3a shows what fraction of the utility is preserved if we reduce the ground set by ReplacementGreedy, GreedySum,



(a) Movielens (b) $\ell = 10$ (c) $\ell = 30$ (d) $\ell = 60$

(e) Movielens Completed (f) $\ell = 10$ (g) $\ell = 30$ (h) $\ell = 60$

Figure 3. Experimental results in the sublinear regime. The first row corresponds to the sublinear experiment where the data set has missing entries, and the second row refers to the experiment where we used a completed user-ratings matrix. The first column portrays the ratio between the mean objective obtained on the summary set versus the ground set for 20 random users from the test set. Columns 2 through 4 showcase the loss versus runtime for $\ell = 10$, $\ell = 30$ and $\ell = 60$ respectively. In these experiments $k = 3$.

and RandomSelection. Clearly, RandomSelection performs poorly. ReplacementGreedy has the highest utility, starting from 80% and practically closing the gap by reducing the ground set to only 60 movies. GreedySum also closely follows ReplacementGreedy. Fig. 3b, 3c, 3d show the loss versus the running time for $\ell = 10$, $\ell = 30$, and $\ell = 60$. Of course, if we use use the whole ground set Ω, the loss will be zero. This is shown by GreedyMerge. However, this comes at the cost of maximizing the utility of each user on 2000 movies. Instead, by using ReplacementGreedy, we see that the loss is negligible (even for $\ell = 30$) while the running time suddenly improves by two orders of magnitude.

Complete matrix movie recommendation. The previous experiment suffers from the potential issue that values are estimated conservatively: i.e., a user derives value only if we happen to select movies that she actually rated in the data. To explore this potential bias, we repeat the same experiment, this time by first completing the matrix of (movie, rating) using standard techniques (Candés & Recht, 2008). We again consider 200 users and divide them into training and test sets. The results are shown in Fig. 3e, 3f, 3g, 3h. We observe exactly the same trends. Basically, a dataset with size 60 is approximately as good as a data set of size 2000 if the reduced ground set is carefully selected by ReplacementGreedy.

6. Analysis

In this section, we prove that Algorithm ReplacementGreedy returns a valid solution S of at most ℓ elements along with independent sets T_i for all different categories such that the aggregate value $\frac{1}{m}\sum_{i=1}^{m} f_i(T_i)$ is at least $\frac{1}{2}(1-\frac{1}{e^2}) > 0.43$ fraction of the optimum solution's objective value, namely $G_m(S^{m,\ell})$. We note that the objective value of ReplacementGreedy's solution, $G_m(S)$, is at least $\frac{1}{m}\sum_{i=1}^{m} f_i(T_i)$, and therefore Algorithm ReplacementGreedy is a $\frac{1}{2}(1-\frac{1}{e^2})$-approximation algorithm for our two stage submodular maximization problem.

Proof of Theorem 3. Since Algorithm ReplacementGreedy runs in ℓ iterations, and adds an element to S in each iteration, the final output size, $|S|$, will not be more than ℓ. Each set T_i is initialized with $\emptyset$ at the beginning which is an independent set. Also whenever ReplacementGreedy adds an element x^* to T_i, it removes $\text{Rep}_i(x^*, T_i)$. By definition, either $\text{Rep}_i(x^*, T_i)$ is equal to some element $y \in T_i$ such that set $\{x^*\} \cup T_i \setminus \{y\}$ is an independent set or $\text{Rep}_i(x^*, T_i)$ is the empty set and $\{x\} \cup T_i$ is an independent set. In either case, set T_i after the update will remain an independent set. Therefore the output of ReplacementGreedy consists of independent sets $T_i \in \mathcal{I}(S)$ for every category i. It remains to lower bound the aggregate values of these m sets.

We lower bound the aggregate increments of values incurred by adding each element we add to S in terms of the gap between the current objective value $\frac{1}{m}\sum_{i=1}^{m} f_i(T_i)$ and the optimum objective value $G_m(S^{m,\ell})$. This way, we can show that for each of the ℓ elements added to S, the current objective value is incremented enough that in total we reach at least $\frac{1}{2}(1-\frac{1}{e^2})$ fraction of the optimum value. By definition of ∇ and the update operations of Algorithm ReplacementGreedy, addition of element x to S, increases the summation $\sum_{i=1}^{m} f_i(T_i)$ by $\sum_{i=1}^{m} \nabla_i(x, T_i)$. We also note that the selected element x^* maximizes this aggregate increment. We prove a lower bound benchmark according to the potential increments of values by elements in $S^{m,\ell}$ if we add them instead of x^*. In particular, we know that:

$$\sum_{i=1}^{m} \nabla_i(x^*, T_i) \geq \frac{1}{|S^{m,\ell}|} \sum_{x \in S^{m,\ell}} \sum_{i=1}^{m} \nabla_i(x, T_i)$$

This equation holds because the rightmost side is the average increments of values of optimum elements if we add them instead of x^*, and we know that x^* is the maximizer of the total value increments. Let $S_i^{m,\ell}$ be the independent subset of $S^{m,\ell}$ with maximum f_i value, i.e. $S_i^{m,\ell} = \arg\max_{A \in \mathcal{I}(S^{m,\ell})} f_i(A)$. Since the ∇ values are all non-negative, we can narrow down the rightmost side of above equation, and imply that:

$$\sum_{i=1}^{m} \nabla_i(x^*, T_i) \geq \frac{1}{|S^{m,\ell}|} \sum_{i=1}^{m} \sum_{x \in S_i^{m,\ell}} \nabla_i(x, T_i) \tag{6}$$

We can apply exchange properties of matroids to lower bound the total ∇ values (the rightmost side) for each category. By Corollary $39.12a$ of (Schrijver, 2003), we know that there exists a mapping π that maps every element of $S_i^{m,\ell} \setminus T_i$ to either the empty set or an element of $T_i \setminus S_i^{m,\ell}$ such that:

- for every $x \in S_i^{m,\ell} \setminus T_i$, the set $\{x\} \cup T_i \setminus \{\pi(x)\}$ is an independent set, and
- for every $x_1, x_2 \in S_i^{m,\ell} \setminus T_i$, we either have $\pi(x_1) \neq \pi(x_2)$ or both of $\pi(x_1)$ and $\pi(x_2)$ are equal to the empty set.

We note that Corollary $39.12a$ of (Schrijver, 2003) is stated for equal size sets (e.g. $|T_i| = |S_i^{m,\ell}|$) which can be easily adapted to non-equal size sets and achieve the above mapping by applying the exchange property of matroids iteratively, and making these two sets equal size. Given the mapping π, the next step is to lower bound each $\nabla_i(x, T_i)$ by how much we can increase the value of T_i by replacing $\pi(x)$ with x in set T_i. Since $\{x\} \cup T_i \setminus \{\pi(x)\}$ is an independent set, we have

$$\begin{aligned} \nabla_i(x, T_i) &\geq f_i(\{x\} \cup T_i \setminus \{\pi(x)\}) - f_i(T_i) \\ &= \Delta_i(x, T_i) - \Delta_i(\pi(x), T_i \cup \{x\} \setminus \{\pi(x)\}) \\ &\geq \Delta_i(x, T_i) - \Delta_i(\pi(x), T_i \setminus \{\pi(x)\}) \end{aligned}$$

where the equality holds by definition of Δ_i values, and the last inequality holds by submodularity of f_i. Combining this lower bound on ∇ with Equation 6 implies that in each step, adding element x^* to set S increases the total value of the current solution by at least:

$$\frac{1}{|S^{m,\ell}|} \sum_{i=1}^{m} \sum_{x \in S_i^{m,\ell} \setminus T_i} \Delta_i(x, T_i) - \Delta_i(\pi(x), T_i \setminus \{\pi(x)\})$$

It is well-known (see Lemma 5 of (Bateni et al., 2010)) that submodularity of f_i implies $\sum_{x \in S_i^{m,\ell} \setminus T_i} \Delta_i(x, T_i) \geq f_i(S_i^{m,\ell}) - f_i(T_i)$. We also have $\sum_{x \in S_i^{m,\ell} \setminus T_i} \Delta_i(\pi(x), T_i \setminus \{\pi(x)\}) \leq \sum_{y \in T_i} \Delta_i(y, T_i \setminus \{y\})$ because range of mapping π is a subset of T_i, and no two elements are mapped to the same $y \in T_i$. By submodularity of f_i, the latter term $\sum_{y \in T_i} \Delta_i(y, T_i \setminus \{y\})$ is upper bounded by $f_i(T_i)$. We conclude that in each step the total value is increased by at least $\frac{1}{\ell m} \sum_{i=1}^{m} f_i(S_i^{m,\ell}) - 2f(T_i)$ (we assume $|S^{m,\ell}| = \ell$ since objective function G_m is monotone increasing). In other words, in each iteration $1 \leq t \leq \ell$, the increment $X_t - X_{t-1}$ is at least $\frac{1}{\ell} G_m(S^{m,\ell}) - \frac{2}{\ell} X_{t-1}$ where X_t is defined to be the total value $\frac{1}{m} \sum_{i=1}^{m} f_i(T_i)$ at the end of iteration t. Solving this recurrence equation inductively yields $X_t \geq \frac{1}{2}(1 - (1 - \frac{1}{\ell})^{2t}) G_m(S^{m,\ell})$. We note that $X_0 = 0$ denotes the total value before the algorithm starts. The induction step is proved as follows:

$$\begin{aligned} & X_{t+1} - X_t \geq \frac{G_m(S^{m,\ell})}{\ell} - \frac{2X_t}{\ell} \\ \Longrightarrow \quad & X_{t+1} \geq \frac{G_m(S^{m,\ell})}{\ell} + (1 - \frac{2}{\ell}) X_t \\ \geq \quad & \frac{G_m(S^{m,\ell})}{\ell} + (1 - \frac{2}{\ell}) \frac{1}{2} (1 - (1 - \frac{1}{\ell})^{2t}) G_m(S^{m,\ell}) \\ \geq \quad & \frac{1}{2} (1 - (1 - \frac{1}{\ell})^{2t+2}) G_m(S^{m,\ell}) \end{aligned}$$

At the end of Algorithm ReplacementGreedy, the total value X_ℓ is at least $\frac{1}{2}(1 - (1 - \frac{1}{\ell})^{2\ell}) G_m(S^{m,\ell}) \geq \frac{1}{2}(1 - \frac{1}{e^2}) G_m(S^{m,\ell})$ which completes the proof. □

7. Conclusions

In this paper, we have studied the novel problem of sublinear time probabilistic submodular maximization: By investing computational effort once to reduce the ground set based on training instances, faster optimization is achieved on test instances. Our key technical contribution is ReplacementGreedy, a novel algorithm for two-stage submodular maximization (the empirical variant of our problem). Compared to prior approaches, ReplacementGreedy provides constant factor approximation guarantees while applying to general submodular objectives, handling arbitrary matroid constraints and scaling linearly in all relevant parameters.

Acknowledgments. This work was supported by DARPA Young Faculty Award (D16AP00046), Simons-Berkeley fellowship, and ERC StG 307036. This work was done in part while Amin Karbasi and Andreas Krause were visiting Simons Institute for the Theory of Computing.

References

Ashwinkumar Badanidiyuru, Shahar Dobzinski, Sigal Oren. Optimization with demand oracles. *EC*, 2012.

Badanidiyuru, Ashwinkumar and Jan, Vondrák. Fast algorithms for maximizing submodular functions. In *SODA*, 2014.

Badanidiyuru, Ashwinkumar, Mirzasoleiman, Baharan, Karbasi, Amin, and Krause, Andreas. Streaming submodular maximization: Massive data summarization on the fly. In *ACM KDD*, 2014.

Balkanski, Erik, Krause, Andreas, Mirzasoleiman, Baharan, and Singer, Yaron. Learning sparse combinatorial representations via two-stage submodular maximization. In *Proc. International Conference on Machine Learning (ICML)*, June 2016.

Bateni, MohammadHossein, Hajiaghayi, MohammadTaghi, and Zadimoghaddam, Morteza. Submodular secretary problem and extensions. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pp. 39–52. Springer, 2010.

Buchbinder, Niv, Feldman, Moran, Naor, Joesph (Seffi), and Schwartz, Roy. Submodular maximization with cardinality constrains. In *SODA*, 2014.

Calinescu, Gruia, Chekuri, Chandra, Pál, Martin, and Vondrák, Jan. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

Candés, E. and Recht, B. Exact matrix completion via convex optimization. In *Foundations of Computational Mathematics*, 2008.

Das, Abhimanyu and Kempe, David. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. *arXiv:1102.3975*, 2011.

El-Arini, Khalid, Veda, Gaurav, Shahaf, Dafna, and Guestrin, Carlos. Turning down the noise in the blogosphere. In *KDD*, 2009.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results, 2014.

Feige, Uriel. On maximizing welfare when utility functions are subadditive. *SIAM Journal on Computing*, 2009.

Feldman, Moran, Harshaw, Christopher, and Karbasi, Amin. Greed is good: Near-optimal submodular maximization via greedy optimization. In *COLT*, 2017.

Goemans, Michel. Chernoff bounds, and some applications, 2015. URL http://math.mit.edu/~goemans/18310S15/chernoff-notes.pdf.

Hazan, Elad and Kale, Satyen. Online submodular minimization. In *NIPS*, 2009.

Jegelka, Stefanie and Bilmes, Jeff A. Online submodular minimization for combinatorial structures. In *International Conference on Machine Learning (ICML)*, Bellevue, Washington, 2011.

Krause, A. and Guestrin, C. Near-optimal nonmyopic value of information in graphical models. In *UAI*, 2005.

Krause, Andreas and Golovin, Daniel. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, 2012.

Krause, Andreas and Gomes, Ryan G. Budgeted nonparametric learning from data streams. In *ICML*, 2010.

Kumar, Ravi, Moseley, Benjamin, Vassilvitskii, Sergei, and Vattani, Andrea. In *SPAA*, 2013.

Lin, Hui and Bilmes, Jeff. A class of submodular functions for document summarization. In *ACL*, 2011.

Lin, Hui and Bilmes, Jeff. Learning mixtures of submodular shells with application to document summarization. In *Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, USA, July 2012. AUAI.

Minoux, Michel. Accelerated greedy algorithms for maximizing submodular set functions. In *Proc. of the 8th IFIP Conference on Optimization Techniques*. Springer, 1978.

Mirzasoleiman, Baharan, Karbasi, Amin, Sarkar, Rik, and Krause, Andreas. Distributed submodular maximization: Identifying representative elements in massive data. In *NIPS*, 2013.

Mirzasoleiman, Baharan, Badanidiyuru, Ashwinkumar, Karbasi, Amin, Vondrak, Jan, and Krause, Andreas. Lazier than lazy greedy. In *AAAI*, 2015.

Mirzasoleiman, Baharan, Badanidiyuru, Ashwinkumar, and Karbasi, Amin. Fast constrained submodular maximization: Personalized data summarization. In *ICML*, 2016a.

Mirzasoleiman, Baharan, Karbasi, Amin, Sarkar, Rik, and Krause, Andreas. Distributed submodular maximization. *Journal of Machine Learning Research*, 17:1–44, 2016b.

Mirzasoleiman, Baharan, Karbasi, Amin, Sarkar, Rik, and Krause, Andreas. Distributed submodular maximization. *Journal of Machine Learning Research (JMLR)*, 2016c.

Mirzasoleiman, Baharan, Zadimoghaddam, Morteza, and Karbasi, Amin. Fast distributed submodular cover: Public-private data summarization. In *NIPS*, 2016d.

Nemhauser, George L., Wolsey, Laurence A., and Fisher, Marshall L. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 1978.

Schrijver, Lex. Combinatorial optimization-polyhedra and efficiency. *Algorithms and Combinatorics*, 24:1–1881, 2003.

Shahar Dobzinski, Noam Nisan and Schapira, Michael. Approximation algorithms for combinatorial auctions with complement-free bidders. *STOC*, 2005.

Singer, Yaron. Budget feasible mechanisms. *FOCS*, 2010.

Streeter, Matthew and Golovin, Daniel. An online algorithm for maximizing submodular functions. In *NIPS*, 2008.

Tschiatschek, Sebastian, Iyer, Rishabh, Wei, Haochen, and Bilmes, Jeff. Learning Mixtures of Submodular Functions for Image Collection Summarization. In *Neural Information Processing Systems (NIPS)*, 2014.

Wei, Kai, Liu, Yuzong, Kirchhoff, Katrin, and Bilmes, Jeff. Using document summarization techniques for speech data subset selection. In *Proceedings of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2013.

Wei, Kai, Iyer, Rishabh, and Bilmes, Jeff. Fast multi-stage submodular maximization. *ICML*, 2014a.

Wei, Kai, Liu, Yuzong, Kirchhoff, Katrin, Bartels, Chris, and Bilmes, Jeff. Submodular subset selection for large-scale speech training data. In *ICASSP*, 2014b.

Yue, Yisong and Guestrin, Carlos. Linear submodular bandits and their application to diversified retrieval. In *NIPS*. 2011.

A. Omitted Proofs

Proof of Theorem 1. For any set $S \subseteq \Omega$, we upper bound the deviation probability, $\Pr_{\mathbb{D}}\left(|G(S) - g_m(S)| > \epsilon\right)$ in terms of the number of sampled functions m. We then use the union bound to prove that the maximum deviation is small over all sets with size at most ℓ with high probability if we choose a large enough m.

For any $1 \leq i \leq m$, define X_i to be $\max_{\substack{T:T\subseteq S \\ |T|\leq k}} f_i(T)$. So $g_m(S)$ is simply the average $\frac{\sum_{i=1}^{m} X_i}{m}$. Since each function f_i is drawn independently from distribution $\mathbb{D}$. Each X_i is a random variable whose mean is equal to $G(S)$ by definition of function G. We use concentration bounds (upper and lower tails) on the aggregate random variable $Y = \sum_{i=1}^{m} X_i$. Variable Y has mean $\mu = mG(S)$ as each of the variables X_i has mean $G(S)$. By applying Theorem 6 of (Goemans, 2015), we have:

$$\begin{aligned}
&\Pr_{\mathbb{D}}\left(|G(S) - g_m(S)| > \epsilon\right) = \\
&\Pr_{\mathbb{D}}\left(Y > \left(1 + \frac{\epsilon}{G(S)}\right)\mu\right) + \\
&\Pr_{\mathbb{D}}\left(Y < \left(1 - \frac{\epsilon}{G(S)}\right)\mu\right) \\
&\leq \exp\left(-\frac{2\left(\frac{\epsilon}{G(S)}\right)^2 \mu^2}{m}\right) + \exp\left(-\frac{\left(\frac{\epsilon}{G(S)}\right)^2 \mu^2}{m}\right) \\
&= \exp\left(-2\epsilon^2 m\right) + \exp\left(-\epsilon^2 m\right) < 2\exp\left(-\epsilon^2 m\right)
\end{aligned}$$

where $\exp(x)$ is defined to be e^x for any real number x. The only inequality in above equations hold by applying the upper and lower tails Chernoff bounds as in Theorem 6 of (Goemans, 2015). The rest of the equations hold by definition of Y and μ, and also the assumption that every function f in the support of distribution $\mathbb{D}$ is normalized such that $0 \leq f(A) \leq 1$ for every subset $A \subseteq \Omega$.

By choosing $m > \frac{1}{\epsilon^2}\ln(\frac{2n^\ell}{\delta})$, we can ensure that the deviation probability for any fixed set S is upper bounded by $\frac{\delta}{n^\ell}$ where $n = |\Omega|$. In other words, we have $\Pr_{\mathbb{D}}\left(|G(S) - g_m(S)| > \epsilon\right) < \frac{\delta}{n^\ell}$. The number of subsets of Ω with size ℓ is $\binom{n}{\ell} \leq n^\ell$. Therefore by applying the union bound, the probability that there exists a set S of size ℓ with deviation more than ϵ is not more than the number of them (at most n^ℓ) times the probability of deviation for each of them (at most $\frac{\delta}{n^\ell}$). We conclude that with probability at least $1-\delta$, we have $|G(S) - g_m(S)| \leq \epsilon$ for every $S \subset \Omega$ with size ℓ. □

We showed that a large enough number of samples m suffice to achieve a small approximation error. One might ask whether the dependence of m on ℓ is necessary or not. Following we show that the number of samples should depend on ℓ linearly otherwise we might have a large approximation error. This shows the optimality of our sample complexity in terms of ℓ.

Remark 1. *There exists distribution $\mathbb{D}$ for which choosing at most ℓ samples will result in an approximation error of $1/e$. We note that this happens when the set S^*_ℓ that maximizes function G achieves a value of $G(S^*_\ell) = 1 - 1/e$, and therefore this approximation error is significant compared to the optimum value we are aiming for.*

Proof. Consider a ground set Ω of n elements. For any set $T \subset \Omega$ with n/ℓ elements, define function f^T as follows. For any set $A \subseteq \Omega$, define $f^T(A) = \min\{1, |A \cap T|\}$. This means the value of A is 1 if only if it has some non-empty intersection with T, and it is 0 otherwise. Suppose we choose $m < \ell$ sample functions that are associated with sets $T_1, T_2, \cdots, T_m$. Construct set S by adding an arbitrarily element from each T_i, and also add some other arbitrary elements from Ω to make its size equal to ℓ. This is possible since $m \leq \ell$. Since set S has non-empty intersection with every T_i, its value $f^{T_i}(S)$ is always equal to 1. Therefore $g_m(S)$ is equal to 1. On the other hand, any set $S' \subset \Omega$ with size ℓ has value $G(S') = 1 - 1/e$. This is evident by noting that a random function f drawn from $\mathbb{D}$, is associated with a random set T of n/ℓ elements, and the probability that T and S' have non-empty intersection is approximately $1 - (1 - \frac{n/\ell}{n})^\ell = 1 - (1 - \frac{1}{\ell})^\ell \approx 1 - \frac{1}{e}$ for large enough n. Therefore for $m \leq \ell$, the maximum value $G(S^*_\ell)$ is $1 - \frac{1}{e}$ while there exists a set S with $g_m(S) = 1$ which eliminates the possibility of having a low approximation error. □

Proof of Theorem 2. Fix any $\epsilon > 0$, and let Ω_ϵ denote an $\epsilon/2kL$-cover of (Ω, d) of minimal size. For each element $x \in \Omega_\epsilon$, define $N^k(x)$ to be the set of k closest points to x in Ω with respect to distance metric d. Since distance of each element to itself is 0, we always have $x \in N^k(x)$. Define $\Omega'_\epsilon = \bigcup_{x\in\Omega_\epsilon} N^k(x)$.

Consider the function $[\cdot]_\epsilon : 2^\Omega \to 2^\Omega$, defined by

$$[T]_\epsilon = \arg\min\{d(T, T') : T' \subseteq \Omega'_\epsilon\}.$$

For any set T of size at most k, we claim that $d(T, [T]_\epsilon) \leq \epsilon/L$ by constructing a set $T' \subseteq \Omega'_\epsilon$ and a matching of T and T' establishing an upper bound of $d(T, T') \leq \epsilon/L$. In particular, let $A = T \setminus \Omega'_\epsilon$. All elements in $T \setminus A$ are matched with themselves, with distance 0. Let $A = \{v_1, \ldots, v_r\}$. Starting from v_1, we iteratively find a unique matching element $v'_i \in \Omega'_\epsilon$ for each $1 \leq i \leq r$ such that $d(v_i, v'_i) \leq \epsilon/kL$. Let x be the closest point in Ω_ϵ to v_i. Since there are at most k elements in set T, and there are k elements in $N^k(x)$. There exists $y \in N^k(x)$ that has not been matched to any previous element, and we can set $v_i' = y$. Since Ω_ϵ is an $\epsilon/2kL$-cover, distance $d(v_i, x)$ is at most $\epsilon/2kL$. On the other hand, v_i is not in set Ω_ϵ' which means it is not one of the k closest points to x. Therefore $d(y, x)$ is at most $d(v_i, x) \le \epsilon/2kL$. Triangle inequality implies $d(y, v_i) \le d(y, x) + d(x, v_i) \le \epsilon/kL$. Therefore there exists a unique matching element v_i' for each v_i with distance $d(v_i, v_i') \le \epsilon/kL$. Hence $d(T, T') \le r\epsilon/kL \le \epsilon/L$. Now suppose that $\ell \ge |\Omega_\epsilon'|$. Then it holds that

$$\begin{aligned}|G(\Omega) - G(S_\ell^*)| &\le |G(\Omega) - G(\Omega_\epsilon')| \\ &\le \mathbb{E}_{f\sim\mathbb{D}}|f(T_f^*) - f([T_f^*]_\epsilon)| \le \epsilon,\end{aligned}$$

where $T_f^* = \arg\max_{T\subseteq\Omega, |T|\le k} f(T)$. We should also note that $|\Omega_\epsilon'|$ is at most $\sum_{x\in\Omega_\epsilon} |N^k(x)| = kn_{\epsilon/2kL}$. □