Journal of Machine Learning Research 16 (2015) 1-41 Submitted 10/14; Published 00/00

Distributed Submodular Maximization

Baharan Mirzasoleiman BAHARANM@INF.ETHZ.CH
Department of Computer Science
ETH Zurich
Universitaetstrasse 6, 8092 Zurich, Switzerland

Amin Karbasi AMIN.KARBASI@YALE.EDU
School of Engineering and Applied Science
Yale University
New Haven, USA

Rik Sarkar RSARKAR@INF.ED.AC.UK
Department of Informatics
University of Edinburgh
10 Crichton St, Edinburgh EH8 9AB, United Kingdom

Andreas Krause KRAUSEA@ETHZ.CH
Department of Computer Science
ETH Zurich
Universitaetstrasse 6, 8092 Zurich, Switzerland

Editor: name

Abstract

Many large-scale machine learning problems–clustering, non-parametric learning, kernel machines, etc.–require selecting a small yet representative subset from a large dataset. Such problems can often be reduced to maximizing a submodular set function subject to various constraints. Classical approaches to submodular optimization require centralized access to the full dataset, which is impractical for truly large-scale problems. In this paper, we consider the problem of submodular function maximization in a distributed fashion. We develop a simple, two-stage protocol GreeDi, that is easily implemented using MapReduce style computations. We theoretically analyze our approach, and show that under certain natural conditions, performance close to the centralized approach can be achieved. We begin with monotone submodular maximization subject to a cardinality constraint, and then extend this approach to obtain approximation guarantees for (not necessarily monotone) submodular maximization subject to more general constraints including matroid or knapsack constraints. In our extensive experiments, we demonstrate the effectiveness of our approach on several applications, including sparse Gaussian process inference and exemplar based clustering on tens of millions of examples using Hadoop.

Keywords: distributed computing, submodular functions, approximation algorithms, greedy algorithms, map-reduce

1. Introduction

Numerous machine learning tasks require selecting representative subsets of manageable size out of large datasets. Examples range from exemplar based clustering (Dueck and

Frey, 2007) to active set selection for non-parametric learning (Rasmussen, 2004), to viral marketing (Kempe et al., 2003), and data subset selection for the purpose of training complex models (Lin and Bilmes, 2011). Many such problems can be reduced to the problem of *maximizing a submodular set function* subject to cardinality or other feasibility constraints such as matroid, or knapsack constraints (Krause and Gomes, 2010; Krause and Golovin, 2012; Lee et al., 2009a).

Submodular functions exhibit a natural diminishing returns property common in many well known objectives: the marginal benefit of any given element decreases as we select more and more elements. Functions such as entropy or maximum weighted coverage are typical examples of functions with diminishing returns. As a result, submodular function optimization has numerous applications in machine learning and social networks: viral marketing (Kempe et al., 2003; Babaei et al., 2013; Mirzasoleiman et al., 2012), information gathering (Krause and Guestrin, 2011), document summarization (Lin and Bilmes, 2011), and active learning (Golovin and Krause, 2011; Guillory and Bilmes, 2011).

Although maximizing a submodular function is NP-hard in general, a seminal result of Nemhauser et al. (1978) states that a simple greedy algorithm produces solutions competitive with the optimal (intractable) solution (Nemhauser and Wolsey, 1978; Feige, 1998). However, such greedy algorithms or their accelerated variants (Minoux, 1978; Badanidiyuru and Vondrák, 2014; Mirzasoleiman et al., 2015a) do not scale well when the dataset is massive. As data volumes in modern applications increase faster than the ability of individual computers to process them, we need to look at ways to adapt our computations using parallelism.

MapReduce (Dean and Ghemawat, 2008) is arguably one of the most successful programming models for reliable and efficient parallel computing. It works by distributing the data to independent machines: *map* tasks redistribute the data for appropriate parallel processing and the output then gets sorted and processed in parallel by *reduce* tasks.

To perform submodular optimization in MapReduce, we need to design suitable parallel algorithms. The greedy algorithms that work well for centralized submodular optimization do not translate easily to parallel environments. The algorithms are inherently sequential in nature, since the marginal gain from adding each element is dependent on the elements picked in previous iterations. This mismatch makes it inefficient to apply classical algorithms directly to parallel setups.

In this paper, we develop a distributed procedure for maximizing submodular functions, that can be easily implemented in MapReduce. Our strategy is to partition the data (e.g., randomly) and process it in parallel. In particular:

- We present a simple, parallel protocol, called GreeDi for distributed submodular maximization subject to cardinality constraints. It requires minimal communication, and can be easily implemented in MapReduce style parallel computation models.

- We show that under some natural conditions, for large datasets the quality of the obtained solution is provably competitive with the best centralized solution.

- We discuss extensions of our approach to obtain approximation algorithms for (not-necessarily monotone) submodular maximization subject to more general types of constraints, including matroid and knapsack constraints.

- We implement our approach for exemplar based clustering and active set selection in Hadoop, and show how our approach allows to scale exemplar based clustering and sparse Gaussian process inference to datasets containing tens of millions of points.
- We extensively evaluate our algorithm on several machine learning problems, including exemplar based clustering, active set selection and finding cuts in graphs, and show that our approach leads to parallel solutions that are very competitive with those obtained via centralized methods (98% in exemplar based clustering, 97% in active set selection, 90% in finding cuts).

This paper is organized as follows. We begin in Section 2 by discussing background and related work. In Section 3, we formalize the distributed submodular maximization problem under cardinality constraints, and introduce example applications as well as naive approaches toward solving the problem. We subsequently present our GreeDi algorithm in Section 4, and prove its approximation guarantees. We then consider maximizing a submodular function subject to more general constraints in Section 5. We also present computational experiments on very large datasets in Section 6, showing that in addition to its provable approximation guarantees, our algorithm provides results close to the centralized greedy algorithm. We conclude in Section 7.

2. Background and Related Work

Distributed Data Analysis and MapReduce Due to the rapid increase in dataset sizes, and the relatively slow advances in sequential processing capabilities of modern CPUs, parallel computing paradigms have received much interest. Inhabiting a sweet spot of resiliency, expressivity and programming ease, the MapReduce style computing model (Dean and Ghemawat, 2008) has emerged as prominent foundation for large scale machine learning and data mining algorithms (Chu et al., 2007; Ekanayake et al., 2008). A MapReduce job takes the input data as a set of $< key; value >$ pairs. Each job consists of three stages: the *map* stage, the *shuffle* stage, and the *reduce* stage. The map stage, partitions the data randomly across a number of machines by associating each element with a *key* and produce a set of $< key; value >$ pairs. Then, in the shuffle stage, the value associated with all of the elements with the same key gets merged and send to the same machine. Each reducer then processes the values associated with the same key and outputs a set of new $< key; value >$ pairs with the same key. The reducers' output could be input to another MapReduce job and a program in MapReduce paradigm can consist of multiple rounds of map and reduce stages (Karloff et al., 2010).

Centralized and Streaming Submodular Maximization The problem of centralized maximization of submodular functions has received much interest, starting with the seminal work of Nemhauser et al. (1978). Recent work has focused on providing approximation guarantees for more complex constraints (for a more detailed account, see the recent survey by Krause and Golovin, 2012). Golovin et al. (2010) consider an algorithm for online distributed submodular maximization with an application to sensor selection. However, their approach requires k stages of communication, which is unrealistic for large k in a MapReduce style model. Krause and Gomes (2010) consider the problem of submodular maximization in a streaming model; however, their approach makes strong assumptions

about the way the data stream is generated and is not applicable to the general distributed setting. Recently, Badanidiyuru et al. (2014) provide a single pass streaming algorithm for cardinality-constrained submodular maximization with $1/2-\varepsilon$ approximation guarantee to the optimum solution that makes no assumptions on the data stream.

There has also been new improvements in the running time of the standard greedy solution for solving SET-COVER (a special case of submodular maximization) when the data is large and disk resident (Cormode et al., 2010). More generally, Badanidiyuru and Vondrák (2014) and Mirzasoleiman et al. (2015a) improve the running time of the greedy algorithm for maximizing a monotone submodular function by reducing the number of oracle calls to the objective function. In a similar spirit, Wei et al. (2014) propose a multi-stage framework for submodular maximization. In order to reduce the memory and computation cost, they apply an approximate greedy procedure to maximize surrogate (proxy) submodular functions instead of optimizing the target function at each stage. The above approaches are sequential in nature and it is not clear how to parallelize them. However, they can be naturally integrated into our distributed framework to achieve further acceleration.

Scaling Up: Distributed Algorithms Recent work has focused on specific instances of submodular optimization in distributed settings. Such scenarios often occur in large-scale graph mining problems where the data itself is too large to be stored on one machine. In particular, Chierichetti et al. (2010) address the MAX-COVER problem and provide a $(1-1/e-\epsilon)$ approximation to the centralized algorithm at the cost of passing over the dataset many times. Their result is further improved by Blelloch et al. (2011). Lattanzi et al. (2011) address more general graph problems by introducing the idea of filtering, namely, reducing the size of the input in a distributed fashion so that the resulting, much smaller, problem instance can be solved on a single machine. This idea is, in spirit, similar to our distributed method GreeDi. In contrast, we provide a more general framework, and characterize settings where performance competitive with the centralized setting can be obtained. The present version is a significant extension of our previous conference paper (Mirzasoleiman et al., 2013), providing theoretical guarantees for both monotone and non-monotone submodular maximization problems subject to more general types of constraints, including matroid and knapsack constraints (described in Section 5), and additional empirical results (Section 6). Parallel to our efforts (Mirzasoleiman et al., 2013), Kumar et al. (2013) has taken the approach of adapting the sequential greedy algorithm to distributed settings. However, their method requires knowledge of the ratio between the largest and smallest marginal gains of the elements, and generally requires a non-constant (logarithmic) number of rounds. We provide empirical comparisons in Section 6.4.

3. Submodular Maximization

In this section, we first review submodular functions and how to greedily maximize them. We then describe the *distributed submodular maximization* problem, the focus of this paper. Finally, we discuss two naive approaches towards solving this problem.

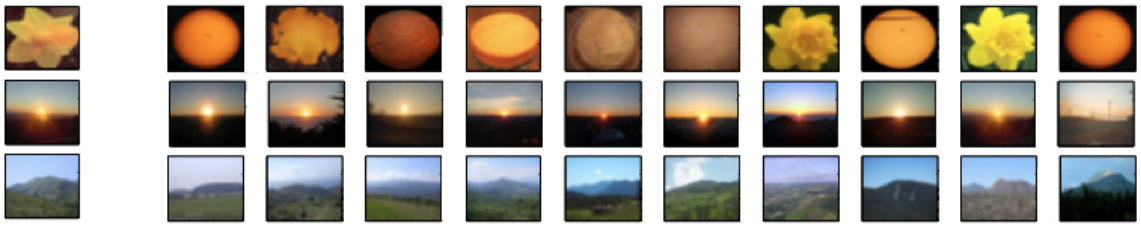


Figure 1: Cluster exemplars (left column) discovered by our distributed algorithm GreeDi described in Section 4 applied to the Tiny Images dataset (Torralba et al., 2008), and a set of representatives from each cluster.

3.1 Greedy Submodular Maximization

Suppose that we have a large dataset of images, e.g. the set of all images on the Web or an online image hosting website such as Flickr, and we wish to retrieve a subset of images that best represents the visual appearance of the dataset. Collectively, these images can be considered as *exemplars* that *summarize* the visual categories of the dataset as shown in Fig. 1.

One way to approach this problem is to formalize it as the *k-medoid* problem. Given a set $V = \{e_1, e_2, \ldots, e_n\}$ of images (called ground set) associated with a (not necessarily symmetric) dissimilarity function, we seek to select a subset $S \subseteq V$ of at most k exemplars or cluster centers, and then assign each image in the dataset to its least dissimilar exemplar. If an element $e \in V$ is assigned to exemplar $v \in S$, then the cost associated with e is the dissimilarity between e and v. The goal of the k-medoid problem is to choose exemplars that minimize the sum of dissimilarities between every data point $e \in V$ and its assigned cluster center.

Solving the k-medoid problem optimally is NP-hard, however, as we discuss in Section 3.4, we can transform this problem, and many other summarization tasks, to the problem of maximizing a monotone submodular function subject to a cardinality constraint

$$\max_{S \subseteq V} f(S) \quad \text{s.t.} \quad |S| \leq k. \tag{1}$$

Submodular functions are set functions which satisfy the following natural diminishing returns property.

Definition 1 (c.f., Nemhauser et al. (1978)) *A set function $f : 2^V \to \mathbb{R}$ is* submodular*, if for every $A \subseteq B \subseteq V$ and $e \in V \setminus B$*

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B).$$

Furthermore, f is called monotone *iff for all $A \subseteq B \subseteq V$ it holds that $f(A) \leq f(B)$.*

We will generally additionally require that f is nonnegative, i.e., $f(A) \geq 0$ for all sets A.

Problem (1) is NP-hard for many classes of submodular functions (Feige, 1998). A fundamental result by Nemhauser et al. (1978) establishes that a simple greedy algorithm

that starts with the empty set and iteratively augments the current solution with an element of maximum incremental value

$$v^* = \arg\max_{v \in V \setminus A} f(A \cup \{v\}), \tag{2}$$

continuing until k elements have been selected, is guaranteed to provide a constant factor approximation.

Theorem 2 ***(Nemhauser et al., 1978)*** *For any non-negative and monotone submodular function f, the greedy heuristic always produces a solution $A^{gc}[k]$ of size k that achieves at least a constant factor $(1 - 1/e)$ of the optimal solution.*

$$f(A^{gc}[k]) \geq (1 - 1/e) \max_{|A| \leq k} f(A).$$

This result can be easily extended to $f(A^{\mathrm{gc}}[l]) \geq (1 - e^{-l/k}) \max_{|A| \leq k} f(A)$, where l and k are two positive integers (see, Krause and Golovin, 2012).

3.2 Distributed Submodular Maximization

In many today's applications where the size of the ground set $|V| = n$ is very large and cannot be stored on a single computer, running the standard greedy algorithm or its variants (e.g., lazy evaluations, Minoux, 1978; Leskovec et al., 2007; Mirzasoleiman et al., 2015a) in a centralized manner is infeasible. Hence, we seek a solution that is suitable for large-scale parallel computation. The greedy method described above is in general difficult to parallelize, since it is inherently sequential: at each step, only the object with the highest marginal gain is chosen and every subsequent step depends on the preceding ones.

Concretely, we consider the setting where the ground set V is very large and cannot be handled on a single machine, thus must be distributed among a set of m machines. While there are several approaches towards parallel computation, in this paper we consider the following model that can be naturally implemented in MapReduce. The computation proceeds in a sequence of rounds. In each round, the dataset is distributed to m machines. Each machine i carries out computations independently in parallel on its local data. After all machines finish, they synchronize by exchanging a limited amount of data (of size polynomial in k and m, but independent of n). Hence, any distributed algorithm in this model must specify: 1) how to distribute V among the m machines, 2) which algorithm should run on each machine, and 3) how to communicate and merge the resulting solutions.

In particular, the distributed submodular maximization problem requires the specification of the above steps in order to implement an approach for submodular maximization. More precisely, given a monotone submodular function f, a cardinality constraint k, and a number of machines m, we wish to produce a solution $A^{\mathrm{d}}[m, k]$ of size k such that $f(A^{\mathrm{d}}[m, k])$ is competitive with the optimal *centralized* solution $\max_{|A| \leq k, A \subseteq V} f(A)$.

3.3 Naive Approaches Towards Distributed Submodular Maximization

One way to solve problem (1) in a distributed fashion is as follows. The dataset is first partitioned (randomly, or using some other strategy) onto the m machines, with V_i representing the data allocated to machine i. We then proceed in k rounds. In each round,

all machines–in parallel–compute the marginal gains of all elements in their sets V_i. Next, they communicate their candidate to a central processor, who identifies the globally best element, which is in turn communicated to the m machines. This element is then taken into account for computing the marginal gains and selecting the next elements. This algorithm (up to decisions on how break ties) implements exactly the centralized greedy algorithm, and hence provides the same approximation guarantees on the quality of the solution. Unfortunately, this approach requires synchronization after each of the k rounds. In many applications, k is quite large (e.g., tens of thousands), rendering this approach impractical for MapReduce style computations.

An alternative approach for large k would be to greedily select k/m elements independently on each machine (without synchronization), and then merge them to obtain a solution of size k. This approach that requires only two rounds (as opposed to k), is much more communication efficient, and can be easily implemented using a single MapReduce stage. Unfortunately, many machines may select redundant elements, and thus the merged solution may suffer from diminishing returns. It is not hard to construct examples for which this approach produces solutions that are a factor $\Omega(m)$ worse than the centralized solution.

In Section 4, we introduce an alternative protocol GreeDi, which requires little communication, while at the same time yielding a solution competitive with the centralized one, under certain natural additional assumptions.

3.4 Applications of Distributed Submodular Maximization

In this part, we discuss two concrete problem instances, with their corresponding submodular objective functions f, where the size of the datasets often requires a distributed solution for the underlying submodular maximization.

3.4.1 Large-scale nonparametric learning

Nonparametric learning (i.e., learning of models whose complexity may depend on the dataset size n) are notoriously hard to scale to large datasets. A concrete instance of this problem arises from training Gaussian processes or performing MAP inference in Determinantal Point Processes, as considered below. Similar challenges arise in many related learning methods, such as training kernel machines, when attempting to scale them to large data sets.

Active Set Selection in Sparse Gaussian Processes (GPs) Formally a GP is a joint probability distribution over a (possibly infinite) set of random variables $\mathbf{X}_V$, indexed by the ground set V, such that every (finite) subset $\mathbf{X}_S$ for $S = \{e_1, \ldots, e_s\}$ is distributed according to a multivariate normal distribution. More precisely, we have

$$P(\mathbf{X}_S = \mathbf{x}_S) = \mathcal{N}(\mathbf{X}_S; \mu_S, \Sigma_{S,S}),$$

where $\mu = (\mu_{e_1}, \ldots, \mu_{e_s})$ and $\Sigma_{S,S} = [\mathcal{K}_{e_i,e_j}]$ are prior mean and covariance matrix, respectively. The covariance matrix is parametrized via a positive definite kernel $\mathcal{K}(\cdot, \cdot)$. As a concrete example, when elements of the ground set V are embedded in a Euclidean space, a commonly used kernel in practice is the squared exponential kernel defined as follows:

$$\mathcal{K}(e_i, e_j) = \exp(-||e_i - e_j||_2^2/h^2).$$

Gaussian processes are commonly used as priors for nonparametric regression. In GP regression, each data point $e \in V$ is considered a random variable. Upon observations $\mathbf{y}_A = \mathbf{x}_A + \mathbf{n}_A$ (where $\mathbf{n}_A$ is a vector of independent Gaussian noise with variance σ^2), the predictive distribution of a new data point $e \in V$ is a normal distribution $P(\mathbf{X}_e \mid \mathbf{y}_A) = \mathcal{N}(\mu_{e|A}, \Sigma^2_{e|A})$, where mean $\mu_{e|A}$ and variance $\sigma^2_{e|A}$ are given by

$$\mu_{e|A} = \mu_e + \Sigma_{e,A}(\Sigma_{A,A} + \sigma^2\mathbf{I})^{-1}(\mathbf{x}_A - \mu_A), \tag{3}$$

$$\sigma^2_{e|A} = \sigma^2_e - \Sigma_{e,A}(\Sigma_{A,A} + \sigma^2\mathbf{I})^{-1}\Sigma_{A,e}. \tag{4}$$

Evaluating (3) and (4) is computationally expensive as it requires solving a linear system of $|A|$ variables. Instead, most efficient approaches for making predictions in GPs rely on choosing a small–so called *active*–set of data points. For instance, in the Informative Vector Machine (IVM) one seeks a set S such that the *information gain*, defined as

$$f(S) = I(\mathbf{Y}_S; \mathbf{X}_V) = H(\mathbf{X}_V) - H(\mathbf{X}_V|\mathbf{Y}_S) = \frac{1}{2}\log\det(\mathbf{I} + \sigma^{-2}\Sigma_{S,S})$$

is maximized. It can be shown that this choice of f is monotone submodular (Krause and Guestrin, 2005a). For medium-scale problems, the standard greedy algorithms provide good solutions. For massive data however, we need to resort to distributed algorithms. In Section 6, we will show how GreeDi can choose near-optimal subsets out of a dataset of 45 million vectors.

Inference for Determinantal Point Processes A very similar problem arises when performing inference in Determinantal Point Processes (DPPs). DPPs (Macchi, 1975) are distributions over subsets with a preference for diversity, i.e., there is a higher probability associated with sets containing dissimilar elements. Formally, a point process $\mathcal{P}$ on a set of items $V = \{1, 2, ..., N\}$ is a probability measure on 2^V (the set of all subsets of V). $\mathcal{P}$ is called *determinantal point process* if for every $S \subseteq V$ we have:

$$\mathcal{P}(S) \propto \det(K_S),$$

where K is a positive semidefinite kernel matrix, and $K_S \equiv [K_{ij}]_{i,j\in S}$, is the restriction of K to the entries indexed by elements of S (we adopt that $\det(K_\emptyset) = 1$). The normalization constant can be computed explicitly from the following equation

$$\sum_S \det(K_S) = \det(\mathbf{I} + K),$$

where I is the $N \times N$ identity matrix. Intuitively, the kernel matrix determines which items are similar and therefore less likely to appear together.

In order to find the most diverse and informative subset of size k, we need to find $\arg\max_{|S|\leq k} \det(K_S)$ which is NP-hard, as the total number of possible subsets is exponential (Ko et al., 1995). However, the objective function is log-submodular, i.e. $f(S) = \log\det(K_S)$ is a submodular function (Kulesza, 2012). Hence, MAP inference in large DPPs is another potential application of distributed submodular maximization.

3.4.2 Large-scale Exemplar Based Clustering

Suppose we wish to select a set of exemplars, that best represent a massive dataset. One approach for finding such exemplars is solving the k-medoid problem (Kaufman and Rousseeuw, 2009), which aims to minimize the sum of pairwise dissimilarities between exemplars and elements of the dataset. More precisely, let us assume that for the dataset V we are given a nonnegative function $l : V \times V \to \mathbb{R}$ (not necessarily assumed symmetric, nor obeying the triangle inequality) such that $l(\cdot, \cdot)$ encodes dissimilarity between elements of the underlying set V. Then, the cost function for the k-medoid problem is:

$$L(S) = \frac{1}{|V|} \sum_{v \in V} \min_{e \in S} l(e, v). \tag{5}$$

Finding the subset

$$S^* = \arg \min_{|S| \leq k} L(S)$$

of cardinality at most k that minimizes the cost function (5) is NP-hard. However, by introducing an auxiliary element e_0, a.k.a. phantom exemplar, we can turn L into a monotone submodular function (Krause and Gomes, 2010)

$$f(S) = L(\{e_0\}) - L(S \cup \{e_0\}). \tag{6}$$

In words, f measures the decrease in the loss associated with the set S versus the loss associated with just the auxiliary element. We begin with a phantom exemplar and try to find the active set that together with the phantom exemplar reduces the value of our loss function more than any other set. Technically, any point e_0 that satisfies the following condition can be used as a phantom exemplar:

$$\max_{v' \in V} l(v, v') \leq l(v, e_0), \quad \forall v \in V \setminus S.$$

This condition ensures that once the distance between any $v \in V \setminus S$ and e_0 is greater than the maximum distance between elements in the dataset, then $L(S \cup \{e_0\}) = L(S)$. As a result, maximizing f (a monotone submodular function) is equivalent to minimizing the cost function L. This problem becomes especially computationally challenging when we have a large dataset and we wish to extract a manageable-size set of exemplars, further motivating our distributed approach.

3.4.3 Other Examples

Numerous other real world problems in machine learning can be modeled as maximizing a monotone submodular function subject to appropriate constraints (e.g., cardinality, matroid, knapsack). To name a few, specific applications that have been considered range from efficient content discovery for web crawlers and multi topic blog-watch (Chierichetti et al., 2010), over document summarization (Lin and Bilmes, 2011) and speech data subset selection (Wei et al., 2013), to outbreak detection in social networks (Leskovec et al., 2007), online advertising and network routing (De Vries and Vohra, 2003), revenue maximization in social networks (Hartline et al., 2008), and inferring network of influence (Gomez Rodriguez et al., 2010). In all such examples, the size of the dataset (e.g., number of webpages,

size of the corpus, number of blogs in the blogosphere, number of nodes in social networks) is massive, thus GreeDi offers a scalable approach, in contrast to the standard greedy algorithm, for such problems.

4. The GreeDi Approach for Distributed Submodular Maximization

In this section we present our main results. We first provide our distributed solution GreeDi for maximizing submodular functions under cardinality constraints. We then show how we can make use of the geometry of data inherent in many practical settings in order to obtain strong data-dependent bounds on the performance of our distributed algorithm.

4.1 An Intractable, yet Communication Efficient Approach

Before we introduce GreeDi, we first consider an intractable, but communication–efficient two-round parallel protocol to illustrate the ideas. This approach, shown in Algorithm 1, first distributes the ground set V to m machines. Each machine then finds the *optimal* solution, i.e., a set of cardinality at most k, that maximizes the value of f in each partition. These solutions are then merged, and the optimal subset of cardinality k is found in the combined set. We denote this distributed solution by $f(A^{\mathrm{d}}[m,k])$.

As the optimum centralized solution $A^{\mathrm{c}}[k]$ achieves the maximum value of the submodular function, it is clear that $f(A^{\mathrm{c}}[k]) \geq f(A^{\mathrm{d}}[m,k])$. For the special case of selecting a single element $k=1$, we have $f(A^{\mathrm{c}}[1]) = f(A^{\mathrm{d}}[m,1])$. Furthermore, for *modular* functions f (i.e., those for which f and $-f$ are both submodular), it is easy to see that the distributed scheme in fact returns the optimal centralized solution as well. In general, however, there can be a gap between the distributed and the centralized solution. Nonetheless, as the following theorem shows, this gap cannot be more than $1/\min(m,k)$. Furthermore, this result is tight.

Theorem 3 *Let f be a monotone submodular function and let $k>0$. Then, $f(A^{d}[m,k])) \geq \frac{1}{\min(m,k)} f(A^{c}[k])$. In contrast, for any value of m and k, there is a monotone submodular function f such that $f(A^{c}[k]) = \min(m,k)\cdot f(A^{d}[m,k])$.*

The proof of all the theorems can be found in the appendix. The above theorem fully characterizes the performance of Algorithm 1 in terms of the best centralized solution. In practice, we cannot run Algorithm 1, since there is no efficient way to identify the optimum subset $A_i^c[k]$ in set V_i, unless P=NP. In the following, we introduce an efficient distributed approximation – GreeDi. We will further show, that under some additional assumptions, much stronger guarantees can be obtained.

4.2 Our GreeDi Approximation

Our efficient distributed method GreeDi is shown in Algorithm 2. It parallels the intractable Algorithm 1, but replaces the selection of optimal subsets, i.e., $A_i^{\mathrm{c}}[k]$, by greedy solutions $A_i^{\mathrm{gc}}[k]$. Due to the approximate nature of the greedy algorithm, we allow it to pick sets slightly larger than k. More precisely, GreeDi is a two-round algorithm that takes the ground set V, the number of partitions m, and the cardinality constraint κ. It first distributes the ground set over m machines. Then each machine separately runs the

Algorithm 1 Inefficient Distributed Submodular Maximization

Input: Set V, #of partitions m, constraints k.
Output: Set $A^{\mathrm{d}}[m,k]$.
1: Partition V into m sets $V_1, V_2, \ldots, V_m$.
2: In each partition V_i find the optimum set $A_i^c[k]$ of cardinality k.
3: Merge the resulting sets: $B = \cup_{i=1}^m A_i^c[k]$.
4: Find the optimum set of cardinality k in B. Output this solution $A^{\mathrm{d}}[m,k]$.

Algorithm 2 Greedy Distributed Submodular Maximization (GreeDi)

Input: Set V, #of partitions m, constraints κ.
Output: Set $A^{\mathrm{gd}}[m,\kappa]$.
1: Partition V into m sets $V_1, V_2, \ldots, V_m$ (arbitrarily or at random).
2: Run the standard greedy algorithm on each set V_i to find a solution $A_i^{\mathrm{gc}}[\kappa]$.
3: Find $A_{\max}^{\mathrm{gc}}[\kappa] = \arg\max_A \{F(A) : A \in \{A_1^{\mathrm{gc}}[\kappa], \ldots, A_m^{\mathrm{gc}}[\kappa]\}\}$
4: Merge the resulting sets: $B = \cup_{i=1}^m A_i^{\mathrm{gc}}[\kappa]$.
5: Run the standard greedy algorithm on B to find a solution $A_B^{\mathrm{gc}}[\kappa]$.
6: Return $A^{\mathrm{gd}}[m,\kappa] = \arg\max_A \{F(A) : A \in \{A_{\max}^{\mathrm{gc}}[\kappa], A_B^{\mathrm{gc}}[\kappa]\}\}$.

standard greedy algorithm by sequentially finding an element $e \in V_i$ that maximizes the discrete derivative (2). Each machine i–in parallel–continues adding elements to the set $A_i^{\mathrm{gc}}[\cdot]$ until it reaches κ elements. We define $A_{\max}^{\mathrm{gc}}[\kappa]$ to be the set with the maximum value among $\{A_1^{\mathrm{gc}}[\kappa], A_2^{\mathrm{gc}}[\kappa], \ldots, A_m^{\mathrm{gc}}[\kappa]\}$. Then the solutions are merged, i.e., $B = \cup_{i=1}^m A_i^{\mathrm{gc}}[\kappa]$, and another round of greedy selection is performed over B until κ elements are selected. We denote this solution by $A_B^{\mathrm{gc}}[\kappa]$. The final distributed solution with parameters m and κ, denoted by $A^{\mathrm{gd}}[m,\kappa]$, is the set with a higher value between $A_{\max}^{\mathrm{gc}}[\kappa]$ and $A_B^{\mathrm{gc}}[\kappa]$ (*c.f.*, Figure 2 shows GreeDi schematically). The following result parallels Theorem 3.

Theorem 4 *Let f be a monotone submodular function and $\kappa \geq k$. Then*

$$f(A^{gd}[m,\kappa]) \geq \frac{(1-e^{-\kappa/k})}{\min(m,k)} f(A^c[k]).$$

For the special case of $\kappa = k$ the result of 4 simplifies to $f(A^{\mathrm{gd}}[m,\kappa]) \geq \frac{(1-1/e)}{\min(m,k)} f(A^{\mathrm{c}}[k])$. Moreover, it is straightforward to generalize GreeDi to multiple rounds (i.e., more than two) for very large datasets.

In light of Theorem 3, one can expect that in general it is impossible to eliminate the dependency of the distributed solution on $\min(k,m)$[1]. However, as we show in the sequel, in many practical settings, the ground set V exhibits rich geometrical structure that can be used to obtain stronger guarantees.

1. It has been very recently shown by Mirzasoleiman et al. (2015b) that the tightest dependency is $\Theta(\sqrt{\min(m,k)})$.

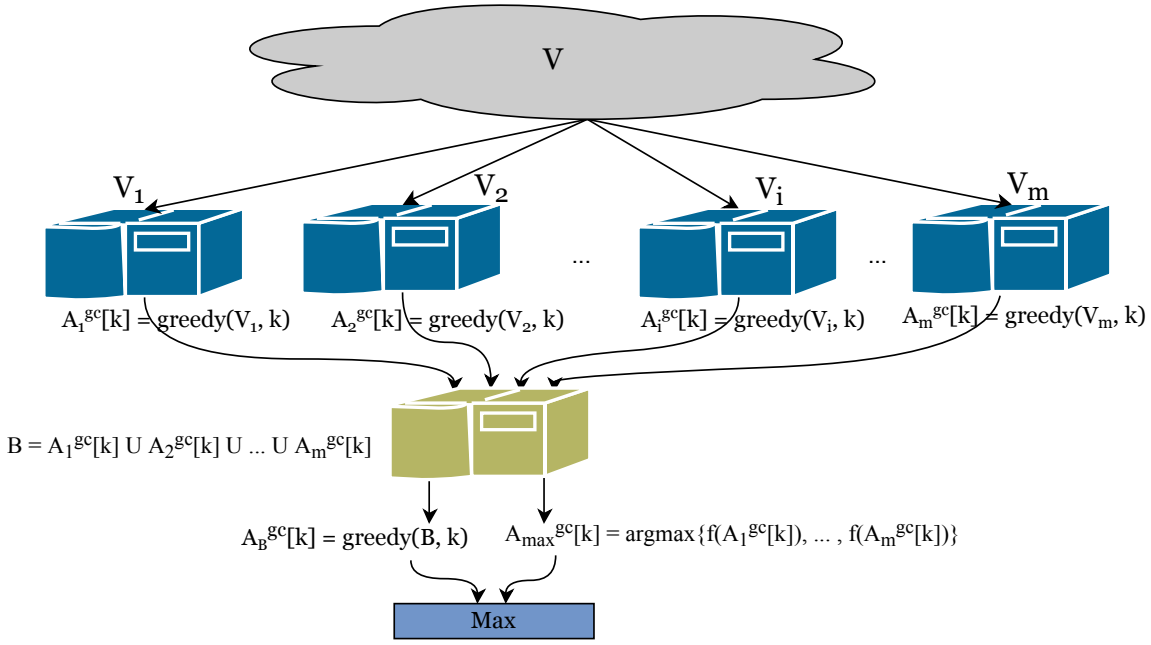


Figure 2: Illustration of our two-round algorithm GreeDi

4.3 Performance on Datasets with Geometric Structure

In practice, we can hope to do much better than the worst case bounds shown previously by exploiting underlying structure often present in real data and important set functions. In this part, we assume that a metric $d : V \times V \to \mathbb{R}$ exists on the data elements, and analyze performance of the algorithm on functions that vary slowly with changes in the input. We refer to these as *Lipschitz functions*:

Definition 5 *Let $\lambda > 0$. A set function $f : 2^V \to \mathbb{R}$ is λ-Lipschitz w.r.t. metric d on V, if for any integer k, any equal sized sets $S = \{e_1, e_2, \ldots, e_k\} \subseteq V$ and $S' = \{e'_1, e'_2, \ldots, e'_k\} \subseteq V$ and any matching of elements: $M = \{(e_1, e'_1), (e_2, e'_2) \ldots, (e_k, e'_k)\}$, the difference between $f(S)$ and $f(S')$ is bounded by:*

$$\left|f(S) - f(S')\right| \leq \lambda \sum_i d(e_i, e'_i). \tag{7}$$

We can show that the objective functions from both examples in Section 3.4 are λ-Lipschitz for suitable kernels/distance functions:

Proposition 6 *Suppose that the covariance matrix of a Gaussian process is parametrized via a positive definite kernel $\mathcal{K} : V \times V \to \mathbb{R}$ which is Lipschitz continuous with respect to metric $d : V \times V \to \mathbb{R}$ with constant $\mathcal{L}$, i.e., for any triple of points $x_1, x_2, x_3 \in V$, we have $|\mathcal{K}(x_1, x_3) - \mathcal{K}(x_2, x_3)| \leq \mathcal{L}d(x_1, x_2)$. Then, the mutual information $I(\mathbf{Y}_S; \mathbf{X}_V) = \frac{1}{2} \log\det(\mathbf{I} + K)$ for the Gaussian process is λ-Lipschitz with $\lambda = \mathcal{L}k^3$, where k is the number of elements in the selected subset S.*

Proposition 7 *Let $d : V \times V \to \mathbb{R}$ be a metric on the elements of the dataset. Furthermore, let $l : V \times V \to \mathbb{R}$ encode the dissimilarity between elements of the underlying set V. Then for $l = d^\alpha$, $\alpha \geq 1$ the loss function $L(S) = \frac{1}{|V|}\sum_{v\in V} \min_{e\in S} l(e, v)$ (and hence also the corresponding submodular utility function f) is λ-Lipschitz with $\lambda = \alpha R^{\alpha-1}$, where R is the diameter of the ball encompassing elements of the dataset in the metric space. In particular, for the k-medoid problem, which minimizes the loss function over all clusters with respect to $l = d$, we have $\lambda = 1$, and for the k-means problem, which minimizes the loss function over all clusters with respect to $l = d^2$, we have $\lambda = 2R$.*

Beyond Lipschitz-continuity, many practical instances of submodular maximization can be expected to satisfy a natural *density* condition. Concretely, whenever we consider a representative set (i.e., optimal solution to the submodular maximization problem), we expect that any of its constituent elements has potential candidates for replacement in the ground set. For example, in our exemplar-based clustering application, we expect that cluster centers are not isolated points, but have many almost equally representative points close by. Formally, for any element $v \in V$, we define its α-*neighborhood* as the set of elements in V within distance α from v (i.e., α-close to v):

$$N_\alpha(v) = \{w : d(v, w) \leq \alpha\}.$$

By λ-Lipschitz-continuity, it must hold that if we replace element v in set S by an α-close element v' (i.e., $v' \in N_\alpha(v)$) to get a new set S' of equal size, it must hold that $|f(S) - f(S')| \leq \alpha\lambda$.

As described earlier, our algorithm GreeDi partitions V into sets $V_1, V_2, \ldots V_m$ for parallel processing. If in addition we assume that elements are assigned uniformly at random to different machines, α-neighborhoods are sufficiently dense, and the submodular function is Lipschitz continuous, then GreeDi is guaranteed to produce a solution close to the centralized one. More formally, we have the following theorem.

Theorem 8 *Under the conditions that 1) elements are assigned uniformly at random to m machines, 2) for each $e_i \in A^c[k]$ we have $|N_\alpha(e_i)| \geq km\log(k/\delta^{1/m})$, and 3) f is λ-Lipschitz continuous, then with probability at least $(1 - \delta)$ the following holds:*

$$f(A^{gd}[m, \kappa]) \geq (1 - e^{-\kappa/k})(f(A^c[k]) - \lambda\alpha k).$$

Note that once the above conditions are satisfied for small values of α (meaning that there is a high density of data points within a small distance from each element of the optimal solution) then the distributed solution will be close to the optimal centralized one. In particular if we let $\alpha \to 0$, the distributed solution is guaranteed to be within a $1 - e^{\kappa/k}$ factor from the optimal centralized solution. This situation naturally corresponds to very large datasets. In the following, we discuss more thoroughly this important scenario.

4.4 Performance Guarantees for Very Large datasets

Suppose that our dataset is a finite sample V drawn i.i.d. from an underlying *infinite* set $\mathcal{V}$, according to some (unknown) probability distribution. Let $A^c[k]$ be an optimal solution in the infinite set, i.e., $A^c[k] = \arg\max_{S\subseteq\mathcal{V}} f(S)$, such that around each $e_i \in A^c[k]$, there is

a neighborhood of radius at least α^* where the probability density is at least β at all points (for some constants α^* and β). This implies that the solution consists of elements coming from reasonably dense and therefore representative regions of the dataset.

Let us suppose $g : \mathbb{R} \to \mathbb{R}$ is the *growth function of the metric*: $g(\alpha)$ is defined to be the volume of a ball of radius α centered at a point in the metric space. This means, for $e_i \in A^c[k]$ the probability of a random element being in $N_\alpha(e_i)$ is at least $\beta g(\alpha)$ and the expected number of α neighbors of e_i is at least $E[|N_\alpha(e_i)|] = n\beta g(\alpha)$. As a concrete example, Euclidean metrics of dimension D have $g(\alpha) = O(\alpha^D)$. Note that for simplicity we are assuming the metric to be homogeneous, so that the growth function is the same at every point. For heterogeneous spaces, we require g to have a uniform lower bound on the growth function at every point.

In these circumstances, the following theorem guarantees that if the dataset V is sufficiently large and f is λ-Lipschitz, then GreeDi produces a solution close to the centralized one.

Theorem 9 *For* $n \geq \dfrac{8km\log(k/\delta^{1/m})}{\beta g(\frac{\varepsilon}{\lambda k})}$, *where* $\frac{\varepsilon}{\lambda k} \leq \alpha^*$, *if the algorithm* GreeDi *assigns elements uniformly randomly to* m *processors , then with probability at least* $(1-\delta)$,

$$f(A^{gd}[m,\kappa]) \geq (1 - e^{-\kappa/k})(f(A^c[k]) - \varepsilon).$$

The above theorem shows that for very large datasets, GreeDi provides a solution that is within a $1 - e^{\kappa/k}$ factor of the optimal centralized solution. This result is based on the fact that for sufficiently large datasets, there is a suitably dense neighborhood around each member of the optimal solution. Thus, if the elements of the dataset are partitioned uniformly randomly to m processors, at least one partition contains a set $A_i^c[k]$ such that its elements are very close to the elements of the optimal centralized solution and provides a constant factor approximation of the optimal centralized solution.

4.5 Handling Decomposable Functions

So far, we have assumed that the objective function f is given to us as a black box, which we can evaluate for any given set S *independently* of the dataset V. In many settings, however, the objective f depends itself on the entire dataset. In such a setting, we cannot use GreeDi as presented above, since we cannot evaluate f on the individual machines without access to the full set V. Fortunately, many such functions have a simple structure which we call *decomposable*. More precisely, we call a submodular function f *decomposable* if it can be written as a sum of submodular functions as follows (Krause and Gomes, 2010):

$$f(S) = \frac{1}{|V|}\sum_{i \in V} f_i(S)$$

In other words, there is separate submodular function associated with every data point $i \in V$. We require that each f_i can be evaluated without access to the full set V. Note that the exemplar based clustering application we discussed in Section 3.4 is an instance of this framework, among many others. Let us define the evaluation of f restricted to $D \subseteq V$ as

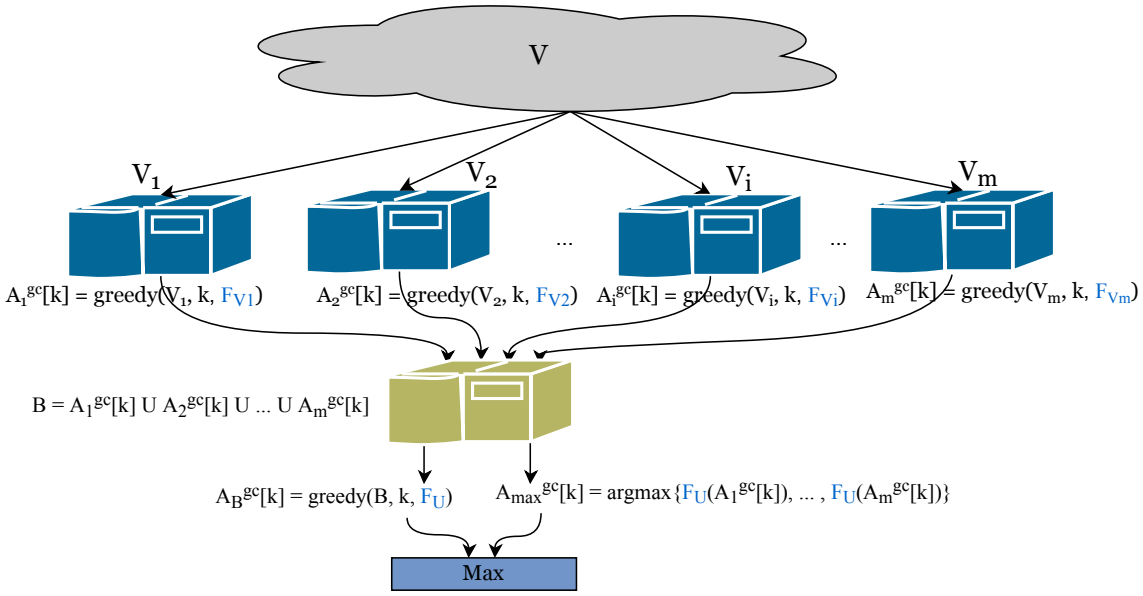


Figure 3: Illustration of our two-round algorithm GreeDi for decomposable functions

follows:

$$f_D(S) = \frac{1}{|D|}\sum_{i\in D} f_i(S)$$

In the remaining of this section, we show that assigning each element of the dataset randomly to a machine and running GreeDi will provide a solution that is with high probability close to the optimum solution. For this, let us assume that f_i's are bounded, and without loss of generality $0 \leq f_i(S) \leq 1$ for $1 \leq i \leq |V|, S \subseteq V$. Similar to Section 4.3 we assume that GreeDi performs the partition by assigning elements uniformly at random to the machines. These machines then each greedily optimize f_{V_i}. The second stage of GreeDi optimizes f_U, where $U \subseteq V$ is chosen uniformly at random with size $\lceil n/m \rceil$.

Then, we can show the following result. First, for any fixed ϵ, m, k, let us define n_0 to be the smallest integer such that for $n \geq n_0$ we have $\ln(n)/n \leq \epsilon^2/(mk)$.

Theorem 10 *For $n \geq \max(n_0, m\log(\delta/4m)/\epsilon^2)$, $\epsilon < 1/4$, and under the assumptions of Theorem 9, we have, with probability at least $1-\delta$,*

$$f(A^{gd}[m,\kappa]) \geq (1-e^{-\kappa/k})(f(A^c[k]) - 2\varepsilon).$$

The above result demonstrates why GreeDi performs well on decomposable submodular functions with massive data even when they are evaluated locally on each machine. We will report our experimental results on exemplar-based clustering in the next section.

4.6 Performance of GreeDi on random partitions without geometric structure

Very recently, a constant $(1-e^{-1})/2$-approximation guarantee was proven for GreeDi for the case of random partitioning of the data among the m machines.

Theorem 11 (Barbosa et al. (2015); Mirrokni and Zadimoghaddam (2015)) *If elements are assigned uniformly at random to the machines, and $\kappa = k$,* GreeDi *gives a $(1-1/e)/2$ approximation guarantee (in the average case) to the optimum centralized solution.*

$$\mathbb{E}[f(A^{gd}[m,k])] \geq \frac{1-1/e}{2} f(A^{c}[k]).$$

These results show that random partitioning of the data is sufficient to guarantee that GreeDi provides a constant factor approximation, irrespective of m and k, and without the requirement of any geometric structure. On the other hand, if geometric structure is present, the bounds from the previous sections can provide sharper approximation guarantees.

5. (Non-monotone) Submodular Functions with General Constraints

In this section we show how GreeDi can be extended to handle 1) more general constraints, and 2) non-monotone submodular functions. More precisely, we consider the following optimization setting

$$\begin{aligned} &\text{Maximize } f(S) \\ &\text{Subject to } S \in \zeta. \end{aligned}$$

Here, we assume that the feasible solutions should be members of the constraint set $\zeta \subseteq 2^V$. The function $f(\cdot)$ is submodular but may not be monotone. By overloading the notation we denote the set that achieves the above constrained optimization problem by $A^{c}[\zeta]$. Throughout this section we assume that the constraint set ζ is hereditary, meaning that if $A \in \zeta$ then for any $B \subseteq A$ we also require that $B \in \zeta$. Cardinality constraints are obviously hereditary, so are all the examples we mention below.

5.1 Matroid Constraints

A matroid $\mathcal{M}$ is a pair $(V, \mathcal{I})$ where V is a finite set (called the ground set) and $\mathcal{I} \subseteq 2^V$ is a family of subsets of V (called the independent sets) satisfying the following two properties:

- *Heredity property*: $A \subseteq B \subseteq V$ and $B \in \mathcal{I}$ implies that $A \in \mathcal{I}$, i.e. every subset of an independent set is independent.
- *Augmentation property*: If $A, B \in \mathcal{I}$ and $|B| > |A|$, there is an element $e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$.

Maximizing a submodular function subject to matroid constraints has found several applications in machine learning and data mining, ranging from content aggregation on the web (Abbassi et al., 2013) to viral marketing (Narayanam and Nanavati, 2012) and online advertising (Streeter et al., 2009).

One way to approximately maximize a monotone submodular function $f(S)$ subject to the constraint that each S is independent, i.e., $S \in \mathcal{I}$, is to use a generalization of the greedy algorithm. This algorithm, which starts with an empty set and in each iteration picks the feasible element with maximum benefit until there is no more element e such that $S \cup \{e\} \in \mathcal{I}$, is guaranteed to provide a $\frac{1}{2}$-approximation of the optimal solution (Fisher

et al., 1978). Recently, this bound has been improved to $(1 - 1/e)$ using the continuous greedy algorithm (Calinescu et al., 2011). For non-negative and non-monotone submodular functions with matroid constraints, the best known result is a 0.325-approximation based on simulated annealing (Gharan and Vondrák, 2011).

Curvature: For a submodular function f, the total curvature of f with respect to a set S is defined as:

$$c = 1 - \min_{j \in V} \frac{f(j|S \setminus j)}{f(j)}.$$

Intuitively, the notion of curvature determines how far away f is from being modular. In other words, it measures how much the marginal gain of an element w.r.t. set S can decrease as a function of S. In general, $c \in [0, 1]$, and for additive (modular) functions, $c = 0$, i.e., the marginal values are independent of S. In this case, the greedy algorithm returns the optimal solution to $\max\{f(S) : S \in \mathcal{I}\}$. In general, the greedy algorithm gives a $\frac{1}{1+c}$-approximation to maximizing a non-decreasing submodular function with curvature c subject to a matroid constraint (Conforti and Cornuéjols, 1984). In case of the uniform matroid $\mathcal{I} = \{S : |S| \leq k\}$, the approximation factor is $(1 - e^{-c})/c$.

Intersection of Matroids: A more general case is when we have p matroids $\mathcal{M}_1 = (V, \mathcal{I}_1), \mathcal{M}_2 = (V, \mathcal{I}_2), ..., \mathcal{M}_p = (V, \mathcal{I}_p)$ on the same ground set V, and we want to maximize the submodular function f on the intersection of p matroids. That is, $\mathcal{I} = \bigcap_i \mathcal{I}_i$ consists of all subsets of V that are independent in all p matroids. This constraint arises, e.g., when optimizing over rankings (which can be modeled as intersections of two partition matroids). Another recent application considered is finding the influential set of users in viral marketing when multiple products need to be advertised and each user can tolerate only a small number of recommendations (Du et al., 2013). For p matroid constraints, the $\frac{1}{p+1}$-approximation provided by the greedy algorithm (Fisher et al., 1978) has been improved to a $(\frac{1}{p} - \varepsilon)$-approximation for $p \geq 2$ by Lee et al. (2009b). For the non-monotone case, a $1/(p + 2 + 1/p + \varepsilon)$-approximation based on local search is also given by Lee et al. (2009b) .

p-systems: p-independence systems generalize constraints given by the intersection of p matroids. Given an independence family $\mathcal{I}$ and a set $V' \subseteq V$, let $S(V')$ denote the set of maximal independent sets of $\mathcal{I}$ included in V', i.e., $S(V') = \{A \in \mathcal{I} \mid \forall e \in V' \setminus A : A \cup \{e\} \notin \mathcal{I}\}$. Then we call $(V, \mathcal{I})$ a p-system if for all nonempty $V' \subseteq V$ we have

$$\max_{A \in S(V')} |A| \leq p \cdot \min_{A \in S(V')} |A|.$$

Similar to p matroid constraints, the greedy algorithm provides a $\frac{1}{p+1}$-approximation guarantee for maximizing a monotone submodular function subject to a p-systems constraint (Fisher et al., 1978). For the non-monotone case, a $2/(3(p+2+1/p))$-approximation can be achieved by combining an algorithm of Gupta et al. (2010) with the result for unconstrained submodular maximization of Buchbinder et al. (2012).

5.2 Knapsack Constraints

In many applications, including feature and variable selection in probabilistic models (Krause and Guestrin, 2005a) and document summarization (Lin and Bilmes, 2011), elements $e \in V$

have non-uniform costs $c(e) > 0$, and we wish to find a collection of elements S that maximize f subject to the constraint that the total cost of elements in S does not exceed a given budget $\mathcal{R}$, i.e.

$$\max_S f(S) \;\; \text{s.t.} \sum_{v \in S} c(v) \leq \mathcal{R}.$$

Since the simple greedy algorithm ignores cost while iteratively adding elements with maximum marginal gains according (see Eq. 2) until $|S| \leq \mathcal{R}$, it can perform arbitrary poorly. However, it has been shown that taking the maximum over the solution returned by the greedy algorithm that works according to Eq. 2 and the solution returned by the modified greedy algorithm that optimizes the cost-benefit ratio

$$v^* = \arg\max_{\substack{e \in V \setminus S \\ c(v) \leq \mathcal{R} - c(S)}} \frac{f(S \cup \{e\}) - f(S)}{c(v)},$$

provides a $(1 - 1/\sqrt{e})$-approximation of the optimal solution (Krause and Guestrin, 2005b). Furthermore, a more computationally expensive algorithm which starts with all feasible solutions of cardinality 3 and augments them using the cost-benefit greedy algorithm to find the set with maximum value of the objective function provides a $(1 - 1/e)$-approximation (Sviridenko, 2004). For maximizing non-monotone submodular functions subject to knapsack constraints, a $(1/5 - \varepsilon)$-approximation algorithm based on local search was given by Lee et al. (2009a).

Multiple Knapsack Constraints: In some applications such as procurement auctions (Garg et al., 2001), video-on-demand systems and e-commerce (Kulik et al., 2009), we have a d-dimensional budget vector $\mathcal{R}$ and a set of element $e \in V$ where each element is associated with a d-dimensional cost vector. In this setting, we seek a subset of elements $S \subseteq V$ with a total cost of at most $\mathcal{R}$ that maximizes a non-decreasing submodular function f. Kulik et al. (2009) proposed a two-phase algorithm that provides a $(1 - 1/e - \varepsilon)$-approximation for the problem by first guessing a constant number of elements of highest value, and then taking the value residual problem with respect to the guessed subset. For the non-monotone case, Lee et al. (2009a) provided a $(1/5 - \varepsilon)$-approximation based on local search.

p-system and d knapsack constraints: A more general type of constraint that has recently found interesting applications in viral marketing (Du et al., 2013) can be constructed by combining a p-system with d knapsack constraints which comprises the intersection of p matroids or d knapsacks as special cases. Badanidiyuru and Vondrák (2014) proposed a modified version of the greedy algorithm that guarantees a $1/(p + 2d + 1)$-approximation for maximizing a monotone submodular function subject to p-system and d knapsack constraints.

Table 1 summarizes the approximation guarantees for monotone and non-monotone submodular maximization under different constraints.

5.3 GreeDi Approximation Guarantee under More General Constraints

Assume that we have a set of constraints $\zeta \subseteq 2^V$ that is hereditary. Further assume we have access to a "black box" algorithm X that gives us a constant factor approximation guar-

Constraint	Approximation (τ)	
	monotone submodular functions	non-monotone submodular functions
Cardinality	$1-1/e$ (Fisher et al., 1978)	0.325 (Gharan and Vondrák, 2011)
1 matroid	$1-1/e$ (Calinescu et al., 2011)	0.325 (Gharan and Vondrák, 2011)
p matroid	$1/p-\varepsilon$ (Lee et al., 2009b)	$1/(p+2+1/p+\varepsilon)$ (Lee et al., 2009b)
1 knapsack	$1-1/e$ (Sviridenko, 2004)	1/5 - ε (Lee et al., 2009a)
d knapsack	$1-1/e-\varepsilon$ Kulik et al. (2009)	1/5 - ε (Lee et al., 2009a)
p-system	$1/(p+1)$ (Fisher et al., 1978)	$2/(3(p+2+1/p))$ (Gupta et al., 2010)
p-system + d knapsack	$1/(p+2d+1)$ (Badanidiyuru and Vondrák, 2014)	–

Table 1: Approximation guarantees (τ) for monotone and non-monotone submodular maximization under different constraints.

antee for maximizing a non-negative (but not necessarily monotone) submodular function f subject to ζ, i.e.

$$X : (f, \zeta) \mapsto A^X \in \zeta \;\text{ s.t. }\; f(A^X[\zeta]) \geq \tau \max_{A \in \zeta} f(A). \tag{8}$$

We can modify GreeDi to use any such approximation algorithm as a black box, and provide theoretical guarantees about the solution. In order to process a large dataset, it first distributes the ground set over m machines. Then instead of greedily selecting elements, each machine i–in parallel–separately runs the black box algorithm X on its local data in order to produce a feasible set $A_i^X[\zeta]$ meeting the constraints ζ. We denote by $A^{\text{gc}}_{\max}[\zeta]$ the set with maximum value among $A_i^X[\zeta]$. Next, the solutions are merged: $B = \cup_{i=1}^m A_i^X[\zeta]$, and the black box algorithm is applied one more time to set B to produce a solution $A_B^{\text{gc}}[\zeta]$. Then, the distributed solution for parameter m and constraints ζ, $A^{Xd}[m, \zeta]$, is the best among $A^{\text{gc}}_{\max}[\zeta]$ and $A_B^{\text{gc}}[\zeta]$. This procedure is given in more detail in Algorithm 3.

Algorithm 3 GreeDi under General Constraints

Input: Set V, #of partitions m, constraints ζ, submodular function f.
Output: Set $A^{Xd}[m, \zeta]$.

1: Partition V into m sets $V_1, V_2, \ldots, V_m$.
2: In parallel: Run the approximation algorithm X on each set V_i to find a solution $A_i^X[\zeta]$.
3: Find $A^{\text{gc}}_{\max}[\zeta] = \arg\max_A \{F(A) | A \in \{A_1^X[\zeta], \ldots, A_m^X[\zeta]\}\}$.
4: Merge the resulting sets: $B = \cup_{i=1}^m A_i^X[\zeta]$.
5: Run the approximation algorithm X on B to find a solution $A_B^{\text{gc}}[\zeta]$.
6: Return $A^{Xd}[m, \zeta] = \arg\max\{A^{\text{gc}}_{\max}[\zeta], A_B^{\text{gc}}[\zeta]\}$.

The following result generalizes Theorem 4 for maximizing a submodular function subject to more general constraints.

Theorem 12 *Let f be a non-negative submodular function and X be a black box algorithm that provides a τ-approximation guarantee for submodular maximization subject to a set of hereditary constraints ζ. Then*

$$f(A^{Xd}[m,\zeta])) \geq \frac{\tau}{\min\big(m, \rho([\zeta])\big)} f(A^c[\zeta]),$$

where $f(A^c[\zeta])$ is the optimum centralized solution, and $\rho([\zeta]) = \max_{A\in\zeta} |A|$.

Specifically, for submodular maximization subject to the matroid constraint $\mathcal{M}$, we have $\rho([A \in \mathcal{I}]) = r_{\mathcal{M}}$ where $r_{\mathcal{M}}$ is the rank of the matroid (i.e., the maximum size of any independent set in the system). For submodular maximization subject to the knapsack constraint $\mathcal{R}$, we can bound $\rho([c(A) \leq \mathcal{R}])$ by $\lceil \mathcal{R}/\min_v c(v) \rceil$ (i.e. the capacity of the knapsack divided by the smallest weight of any element).

Performance on Datasets with Geometric Structure. When the submodular function $f(\cdot)$ and the constraint set ζ have more structure, then we can provide much better approximation guarantees. Assuming the elements of V are embedded in metric space with distance $d : V \times V \to \mathbb{R}^+$, we say that ζ is *locally replaceable* with respect to a set $S \subseteq V$ with parameter $\alpha > 0$ if

$$\forall S' \subseteq V \text{ s.t. } |S'| = |S| \text{ and } d_\infty(S, S') \leq \alpha \Rightarrow S' \in \zeta.$$

Here, we define the distance d_∞ between two sets S and S' of the same size k as follows. Let M be the set of all possible matchings between S and S', i.e.,

$$M = \{((e_1, e'_1), \ldots, (e_k, e'_k)) \text{ s.t } e_i \in S \text{ and } e'_i \in S' \text{ for } 1 \leq i \leq k\}.$$

Then $d_\infty(S, S') = \min_M \max_i d(e_i, e'_i)$. We require locality only with respect to $A^c[\zeta]$ to ensure that the optimum solution can be well approximated. What the locally replaceable property requires is that as elements of $A^c[\zeta]$ get replaced by nearby elements, the resulting set is also a feasible solution. Combining this property with λ-Lipschitzness will provide us with the following theorem.

Theorem 13 *Under the conditions that 1) elements are assigned uniformly at random to m machines, 2) for each $e_i \in A^c[\zeta]$ we have $|N_\alpha(e_i)| \geq \rho([\zeta])m\log(\rho([\zeta])/\delta^{1/m})$, 3) $f(\cdot)$ is λ-Lipschitz, and 4) ζ is locally replaceable with respect to $A^c[\zeta]$ with parameter α, then with probability at least $(1-\delta)$,*

$$f(A^{Xd}[m,\zeta])) \geq \tau(f(A^c[\zeta]) - \lambda\alpha\rho([\zeta])).$$

The above result generalizes Theorem 8 for maximizing non-negative submodular functions subject to different constraints.

Performance Guarantee for Very Large datasets. Similarly, we can generalize Theorem 9 for maximizing non-negative submodular functions subject to more general constraints. Suppose that our dataset is a finite sample V drawn i.i.d. from an underlying *infinite* set $\mathcal{V}$, according to some (unknown) probability distribution. Let $A^c[\zeta]$ be an optimal solution in the infinite set, i.e., $A^c[\zeta] = \arg\max_{S\subseteq\mathcal{V}} f(S)$, such that around each

$e_i \in A^c[\zeta]$, there is a neighborhood of radius at least α^* where the probability density is at least β at all points (for some constants α^* and β). Recall that $g : \mathbb{R} \to \mathbb{R}$ is the growth function where $g(\alpha)$ measures the volume of a ball of radius α centered at a point in the metric space.

Theorem 14 *For* $n \geq \frac{8\rho([\zeta])m\log(\rho([\zeta])/\delta^{1/m})}{\beta g(\frac{\varepsilon}{\lambda\rho([\zeta])})}$, *where* $\frac{\varepsilon}{\lambda\rho([\zeta])} \leq \alpha^*$, *if* GreeDi *assigns elements uniformly at random to* m *processors and under the conditions that* f *is* λ*-Lipschitz, and* ζ *is locally replaceable with respect to* $A^c[\zeta]$ *with parameter* α^*, *then with probability at least* $(1-\delta)$, *we have*

$$f(A^{Xd}[m,\zeta])) \geq \tau(f(A^c[\zeta]) - \varepsilon).$$

Performance Guarantee for Decomposable Functions. For the case of decomposable functions described in Section 4.5, the following generalization of Theorem 10 holds for maximizing a non-negative submodular function subject to more general constraints. Let us define n_0 to be the smallest integer such that for $n \geq n_0$ we have $\ln(n)/n \leq \epsilon^2/(m \cdot \rho([\zeta]))$.

Theorem 15 *For* $n \geq \max(n_0, m\log(\delta/4m)/\epsilon^2)$, $\epsilon < 1/4$, *and under the assumptions of Theorem 14, we have, with probability at least* $1-\delta$,

$$f(A^{Xd}[m,\zeta])) \geq \tau(f(A^c[\zeta]) - 2\varepsilon).$$

6. Experiments

In our experimental evaluation we wish to address the following questions: 1) how well does GreeDi perform compared to the centralized solution, 2) how good is the performance of GreeDi when using decomposable objective functions (see Section 4.5), and finally 3) how well does GreeDi scale in the context of massive datasets. To this end, we run GreeDi on three scenarios: exemplar based clustering, active set selection in GPs and finding the maximum cuts in graphs.

We compare the performance of our GreeDi method to the following naive approaches:

- *random/random*: in the first round each machine simply outputs k randomly chosen elements from its local data points and in the second round k out of the merged mk elements, are again randomly chosen as the final output.

- *random/greedy*: each machine outputs k randomly chosen elements from its local data points, then the standard greedy algorithm is run over mk elements to find a solution of size k.

- *greedy/merge*: in the first round k/m elements are chosen greedily from each machine and in the second round they are merged to output a solution of size k.

- *greedy/max*: in the first round each machine greedily finds a solution of size k and in the second round the solution with the maximum value is reported.

For GreeDi, we let each of the m machines select a set of size αk, and select a final solution of size k among the union of the m solutions (i.e., among αkm elements). We present the performance of GreeDi for different parameters $\alpha > 0$. For datasets where we are able to find the centralized solution, we report the ratio of $f(A_{\text{dist}}[k])/f(A^{\text{gc}}[k])$, where $A_{\text{dist}}[k]$ is the distributed solution (in particular $A^{\text{gd}}[m, \alpha k, k] = A_{\text{dist}}[k]$ for GreeDi).

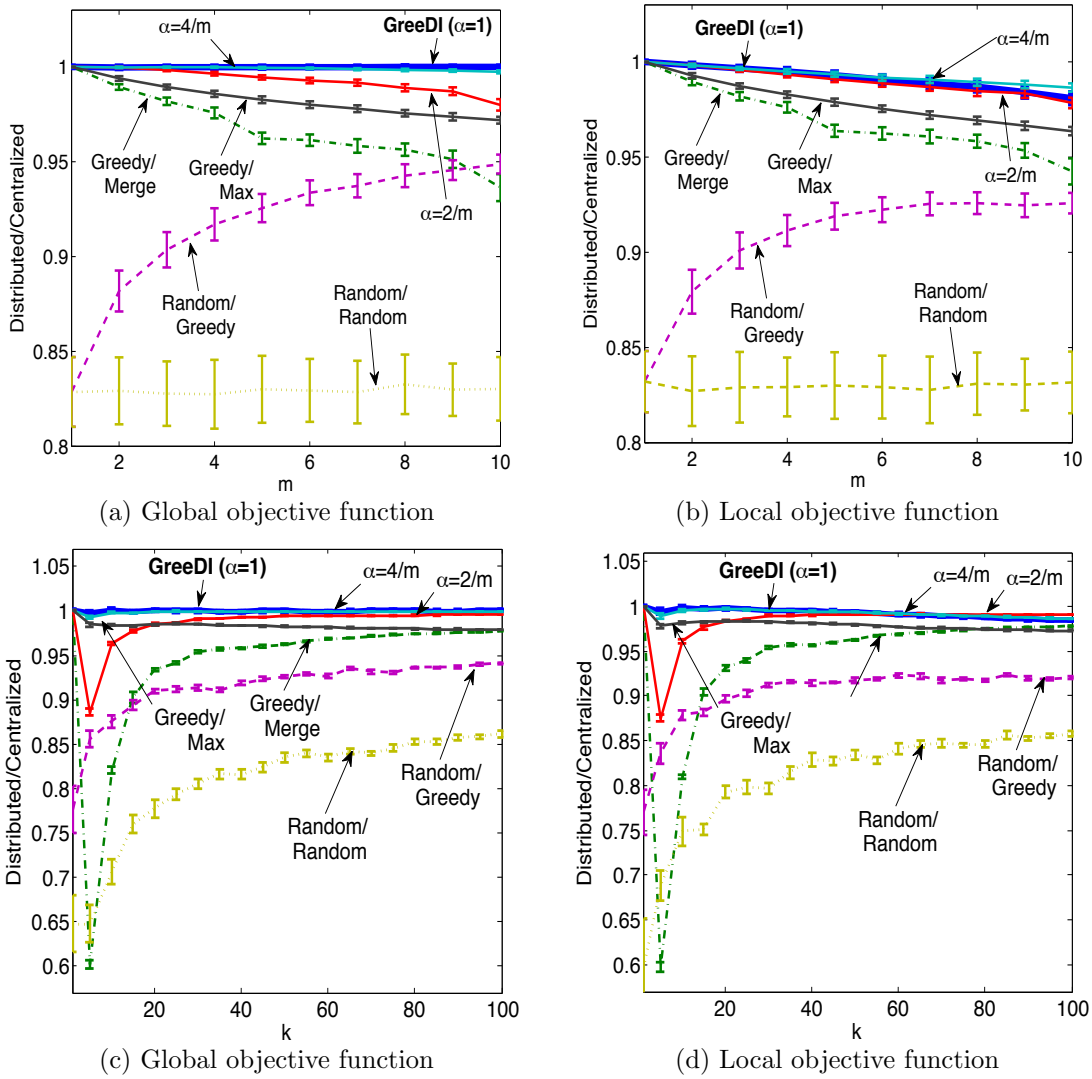
6.1 Exemplar Based Clustering

Our exemplar based clustering experiment involves GreeDi applied to the clustering utility $f(S)$ (see Sec. 3.4) with $d(x, x') = \|x - x'\|^2$. We performed our experiments on a set of 10,000 *Tiny Images* (Torralba et al., 2008). Each 32 by 32 RGB pixel image was represented by a 3,072 dimensional vector. We subtracted from each vector the mean value, normalized it to unit norm, and used the origin as the auxiliary exemplar. Fig. 4a compares the performance of our approach to the benchmarks with the number of exemplars set to $k = 50$, and varying number of partitions m. It can be seen that GreeDi significantly outperforms the benchmarks and provides a solution that is very close to the centralized one. Interestingly, even for very small $\alpha = \kappa/k < 1$, GreeDi performs very well. Since the exemplar based clustering utility function is decomposable, we repeated the experiment for the more realistic case where the function evaluation in each machine was restricted to the local elements of the dataset in that particular machine (rather than the entire dataset). Fig 4b shows similar qualitative behavior for decomposable objective functions.

Large scale experiments with Hadoop. As our first large scale experiment, we applied GreeDi to the whole dataset of 80,000,000 *Tiny Images* (Torralba et al., 2008) in order to select a set of 64 exemplars. Our experimental infrastructure was a cluster of 10 quad-core machines running Hadoop with the number of reducers set to $m = 8000$. Hereby, each machine carried out a set of reduce tasks in sequence. We first partitioned the images uniformly at random to reducers. Each reducer separately performed the lazy greedy algorithm on its own set of 10,000 images (≈123MB) to extract 64 images with the highest marginal gains w.r.t. the local elements of the dataset in that particular partition. We then merged the results and performed another round of lazy greedy selection on the merged results to extract the final 64 exemplars. Function evaluation in the second stage was performed w.r.t a randomly selected subset of 10,000 images from the entire dataset. The maximum running time per reduce task was 2.5 hours. As Fig. 5a shows, GreeDi highly outperforms the other distributed benchmarks and can scale well to very large datasets. Fig. 5b shows a set of cluster exemplars discovered by GreeDi where Fig. 5c and Fig. 5d show 100 nearest images to exemplars 26 and 63 (shown with red borders) in Fig. 5b.

6.2 Active Set Selection

Our active set selection experiment involves GreeDi applied to the information gain $f(S)$ (see Sec. 3.4) with Gaussian kernel, $h = 0.75$ and $\sigma = 1$. We used the *Parkinsons Telemonitoring* dataset (Tsanas et al., 2010) consisting of 5,875 bio-medical voice measurements with 22 attributes from people with early-stage Parkinson's disease. We normalized the vectors to zero mean and unit norm. Fig. 6b compares the performance GreeDi to the benchmarks with fixed $k = 50$ and varying number of partitions m. Similarly, Fig 6a shows



(a) Global objective function

(b) Local objective function

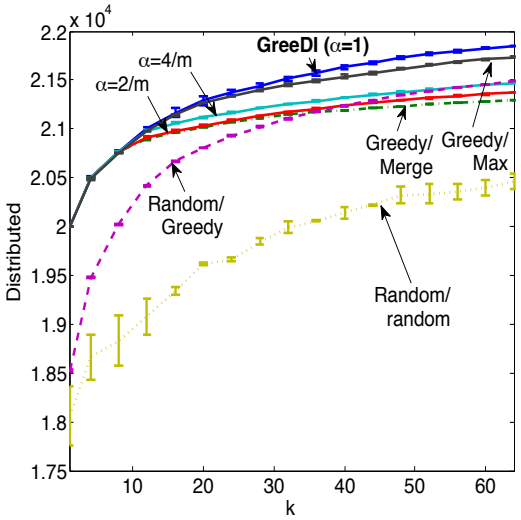
(c) Global objective function

(d) Local objective function

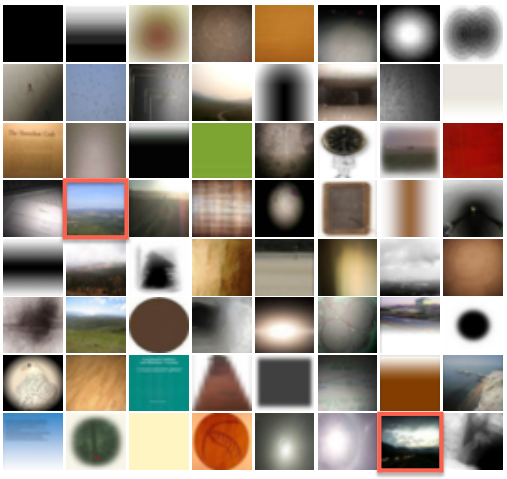
Figure 4: Performance of GreeDi compared to the other benchmarks. a) and b) show the mean and standard deviation of the ratio of distributed vs. centralized solution for global and local objective functions with budget $k = 50$ and varying the number m of partitions. c) and d) show the same ratio for global and local objective functions for $m = 5$ partitions and varying budget k, for a set of 10,000 *Tiny Images*.

the results for fixed $m = 10$ and varying k. We find that GreeDi significantly outperforms the benchmarks.

Large scale experiments with Hadoop. Our second large scale experiment consists of 45,811,883 user visits from the Featured Tab of the Today Module on Yahoo! Front Page (web, 2012). For each visit, both the user and each of the candidate articles are associated with a feature vector of dimension 6. Here, we used the normalized user features. Our experimental setup was a cluster of 8 quad-core machines running Spark with the number

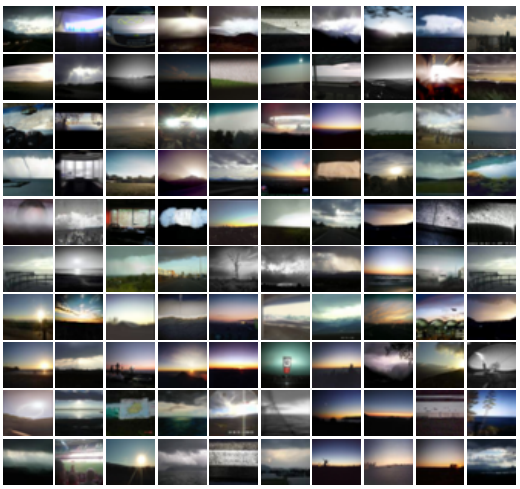


(a) Tiny Images 80M



(b)

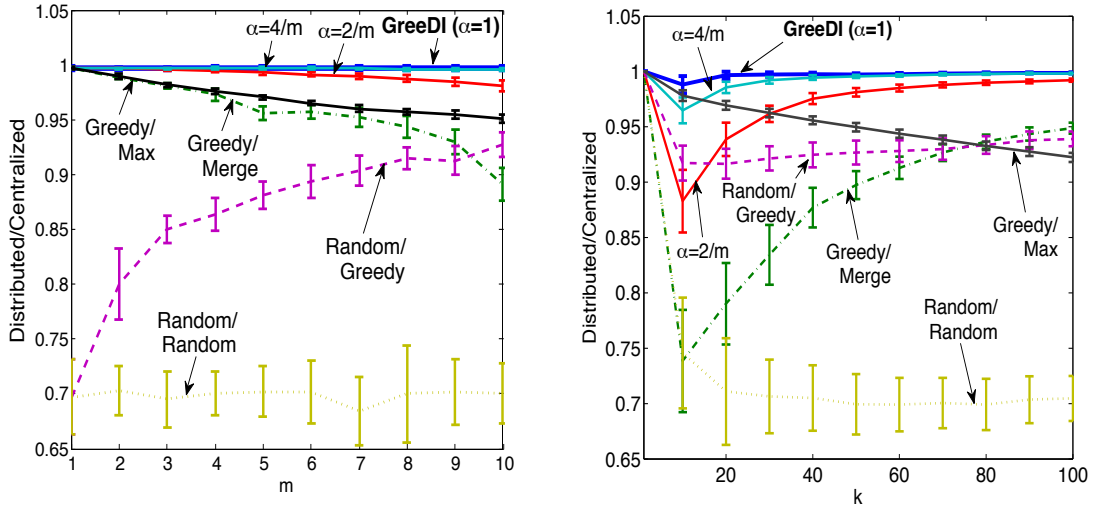
(c)



(d)

Figure 5: Performance of GreeDi compared to the other benchmarks. a) shows the distributed solution with $m = 8000$ and varying k for local objective functions on the whole dataset of 80,000,000 *Tiny Images*. b) shows a set of cluster exemplars discovered by GreeDi, and each column in c) shows 100 images nearest to exemplars 26 and d) shows 100 images nearest to exemplars 63 in b).

of reducers set to $m = 32$. Each reducer performed the lazy greedy algorithm on its own set of ≈1,431,621 vectors (≈34MB) in order to extract 256 elements with the highest marginal gains w.r.t the local elements of the dataset in that particular partition. We then merged the results and performed another round of lazy greedy selection on the merged results to extract the final active set of size 256. The maximum running time per reduce task was 12 minutes for selecting 128 elements and 48 minutes for selecting 256 elements. Fig. 7 shows the performance of GreeDi compared to the benchmarks. We note again that GreeDi significantly outperforms the other distributed benchmarks and can scale well to very large datasets.



(a) Parkinsons Telemonitoring (b) Parkinsons Telemonitoring

Figure 6: Performance of GreeDi compared to the other benchmarks. a) shows the ratio of distributed vs. centralized solution with $k = 50$ and varying m for *Parkinsons Telemonitoring*. b) shows the same ratio with $m = 10$ and varying k on the same dataset.

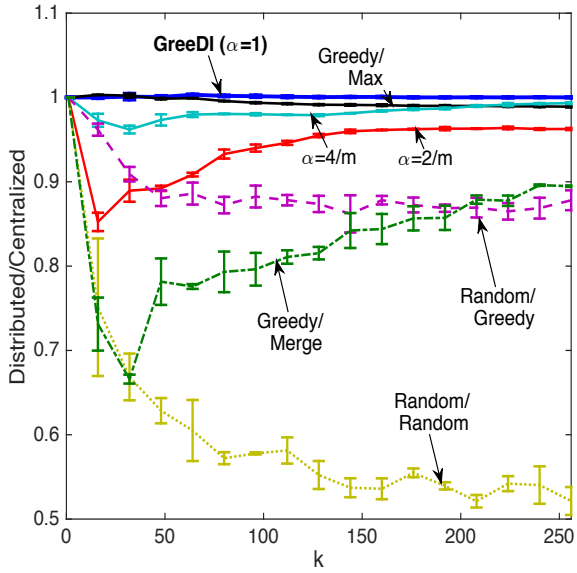
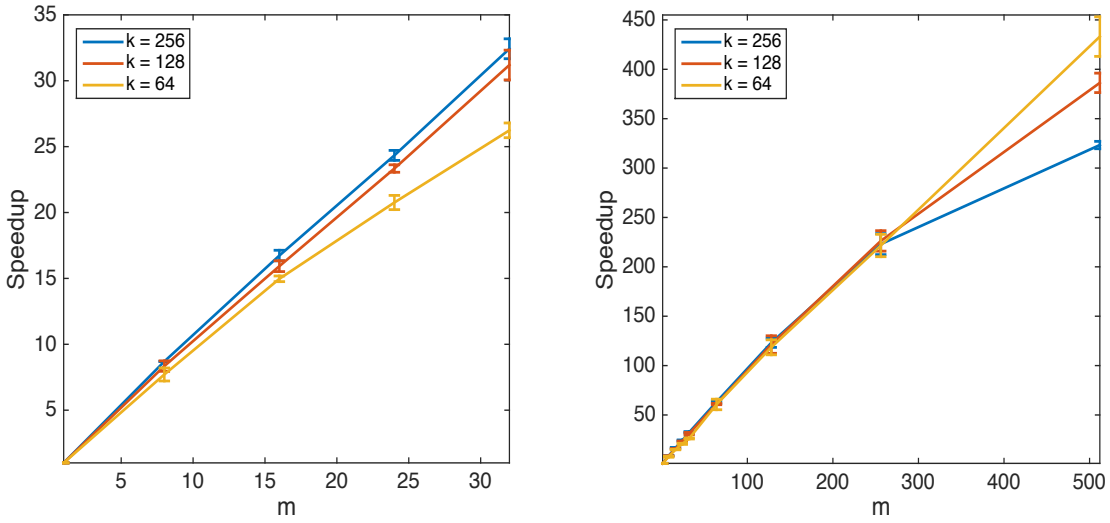


Figure 7: Performance of GreeDi with $m = 32$ and varying budget k compared to the other benchmarks on *Yahoo! Webscope data*.

Performance Comparison. Fig. 8 shows the speedup of GreeDi compared to the centralized greedy benchmark for different values of k and varying number of partitions m. As Fig. 8a shows, for small values of m, the speedup is almost linear in the number of machines. However, for large values of m the running time of the second stage of GreeDi increases and ultimately dominates the whole running time. Hence, we do not observe a linear speedup anymore. This effect can be observed in Fig. 8b. For larger values of k, the



(a) Yahoo! front page

(b) Yahoo! front page

Figure 8: Running time of GreeDi compared to the centralized greedy algorithm. a) shows the ratio of centralized vs. distributed solution with $k = 64, 128, 256$ and up to $m = 32$ machines for *Yahoo Webscope* data. b) shows the same ratio with $k = 64, 128, 256$ and up to $m = 512$ machines on the same dataset. Both experiments are performed on a cluster of 8 quad core machines.

speedup is higher on fewer machines, but decreases more quickly by increasing m, as the second stage takes longer to complete.

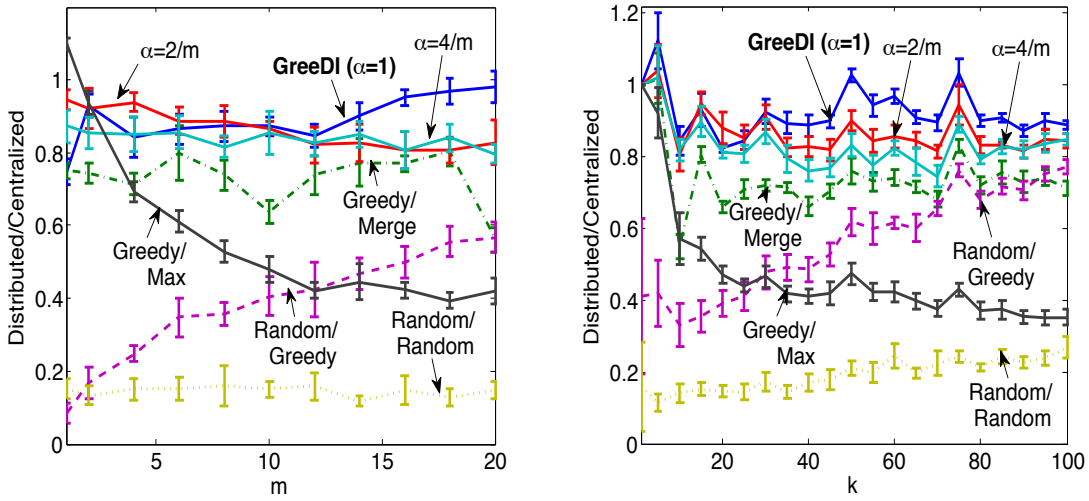
6.3 Non-Monotone Submodular Function (Finding Maximum Cuts)

We also applied GreeDi to the problem of finding maximum cuts in graphs. In our setting we used a *Facebook-like social network* (Opsahl and Panzarasa, 2009). This dataset includes the users that have sent or received at least one message in an online student community at University of California, Irvine and consists of 1,899 users and 20,296 directed ties. Fig. 9a and 9b show the performance of GreeDi applied to the cut function on graphs. We evaluated the objective function locally on each partition. Thus, the links between the partitions are disconnected. Since the problem of finding the maximum cut in a graph is non-monotone submodular, we applied the RandomGreedy algorithm proposed by Buchbinder et al. (2014) to find the near optimal solution in each partition.

Although the cut function does not decompose additively over individual data points, perhaps surprisingly, GreeDi still performs very well, and significantly outperforms the benchmarks. This suggests that our approach is quite robust, and may be more generally applicable.

6.4 Comparision with Greedy Scaling.

Kumar et al. (2013) recently proposed an alternative approach–GreedyScaling–for parallel maximization of submodular functions. GreedyScaling is a randomized algorithm



(a) Facebook-like social network (b) Facebook-like social network

Figure 9: Performance of GreeDi compared to the other benchmarks. a) shows the mean and standard deviation of the ratio of distributed to centralized solution for budget $k = 20$ with varying number of machines m and b) shows the same ratio for varying budget k with $m = 10$ on *Facebook-like social network*.

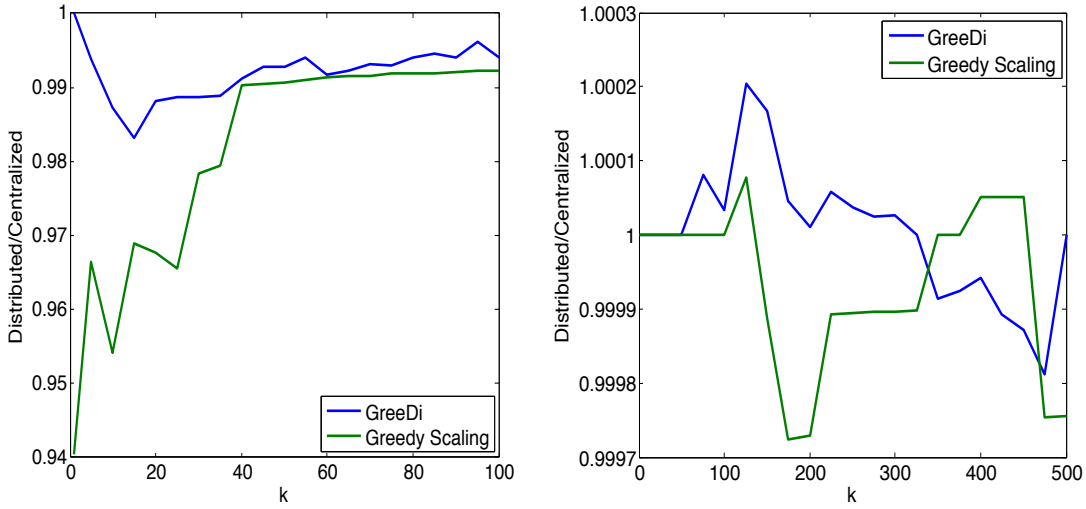
that carries out a number (typically less than k) rounds of MapReduce computations. We applied GreeDi to the submodular coverage problem in which given a collection V of sets, we would like to pick at most k sets from V in order to maximize the size of their union. We compared the performance of our GreeDi algorithm to the reported performance of GreedyScaling on the same datasets, namely *Accidents* (Geurts et al., 2003) and *Kosarak* (Bodon, 2012). As Fig 10a and 10b shows, GreeDi outperforms GreedyScaling on the *Accidents* dataset and its performance is comparable to that of GreedyScaling in the *Kosarak* dataset.

7. Conclusion

We have developed an efficient distributed protocol GreeDi, for constrained submodular function maximization. We have theoretically analyzed the performance of our method and showed that under certain natural conditions it performs very close to the centralized (albeit impractical in massive datasets) solution. We have also demonstrated the effectiveness of our approach through extensive experiments, including active set selection in GPs on a dataset of 45 million examples, and exemplar based summarization of a collection of 80 million images using Hadoop. We believe our results provide an important step towards solving submodular optimization problems in very large scale, real applications.

Acknowledgments

This research was supported by SNF 200021-137971, DARPA MSEE FA8650-11-1-7156, ERC StG 307036, a Microsoft Faculty Fellowship, an ETH Fellowship, and a Scottish Informatics and Computer Science Alliance.



(a) Accidents (b) Kosarak

Figure 10: Performance of GreeDi compared to the GreedyScaling algorithm of Kumar et al. (2013) (as reported in their paper). a) shows the ratio of distributed to centralized solution on *Accidents* dataset with 340,183 elements and b) shows the same ratio for *Kosarak* dataset with 990,002 elements. The results are reported for varying budget k and varying number of machines $m = n/\mu$ where $\mu = O(kn^{\delta} \log n)$ and n is the size of the dataset. The results are reported for $\delta = 1/2$. Note that the results presented by Kumar et al. (2013) indicate that GreedyScaling generally requires a substantially larger number of MapReduce rounds compared to GreeDi.

Appendix A. Proofs

This section presents the complete proofs of theorems presented in the article.

Proof of Theorem 3

$\Rightarrow$ direction:
The proof easily follows from the following lemmas.

Lemma 16 $\max_i f(A_i^c[k]) \geq \frac{1}{m} f(A^c[k])$.

Proof Let B_i be the elements in V_i that are contained in the optimal solution, $B_i = A^c[k] \cap V_i$. Then we have:

$$f(A^c[k]) = f(B_1 \cup \ldots \cup B_m) = f(B_1) + f(B_2|B_1) + \ldots + f(B_m|B_{m-1}, \ldots, B_1).$$

Using submodularity of f, for each $i \in \{1 \ldots m\}$, we have

$$f(B_i|B_{i-1} \ldots B_1) \leq f(B_i),$$

and thus,

$$f(A^c[k]) \leq f(B_1) + \ldots + f(B_m).$$

Since, $f(A_i^c[k]) \geq f(B_i)$, we have

$$f(A^c[k]) \leq f(A_1^c[k]) + \ldots + f(A_m^c[k]).$$

Therefore,

$$f(A^c[k]) \leq m \max_i f(A_i^c[k]).$$

■

Lemma 17 $\max_i f(A_i^c[k]) \geq \frac{1}{k} f(A^c[k])$.

Proof Let $f(A^c[k]) = f(\{u_1, \ldots u_k\})$. Using submodularity of f, we have

$$f(A^c[k]) \leq \sum_{i=1}^{k} f(u_i).$$

Thus, $f(A^c[k]) \leq k f(u^*)$ where $u^* = \arg\max_i f(u_i)$. Suppose that the element with highest marginal gain (i.e., u^*) is in V_j. Then the maximum value of f on V_j would be greater or equal to the marginal gain of u^*, i.e., $f(A_j^c[k]) \geq f(u^*)$ and since $f(\max_i f(A_i^c[k])) \geq f(A_j^c[k])$, we can conclude that

$$f(\max_i f(A_i^c[k])) \geq f(u^*) \geq \frac{1}{k} f(A^c[k]).$$

■

Since $f(A^d[m,k]) \geq \max_i f(A_i^c[k])$; from Lemma 16 and 17 we have

$$f(A^d[m,k]) \geq \frac{1}{\min(m,k)} f(A^c[k]).$$

$\Leftarrow$ direction:
Let us consider a set of unbiased and independent Bernoulli random variables $X_{i,j}$ for $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, k\}$, i.e., $\Pr(X_{i,j} = 1) = \Pr(X_{i,j} = 0) = 1/2$ and $(X_{i,j} \perp X_{i',j'})$ if $i \neq i'$ or $j \neq j'$. Let us also define $Y_i = (X_{i,1}, \ldots, X_{i,k})$ for $i \in \{1, \ldots, m\}$. Now assume that $V_i = \{X_{i,1}, \ldots, X_{i,k}, Y_i\}, V = \bigcup_{i=1}^m V_i$ and $f(S) = H(S)$, where H is the entropy of the subset S of random variables. Note that H is a monotone submodular function. It is easy to see that $A_i^c[k] = \{X_{i,1}, \ldots, X_{i,k}\}$ or $A_i^c[k] = Y_i$ as in both cases $H(A_i^c[k]) = k$. If we assume $A_i^c[k] = \{X_{i,1}, \ldots, X_{i,k}\}$, then $B = \{X_{i,j} | 1 \leq i \leq m, 1 \leq j \leq k\}$. Hence, by selecting at most k elements from B, we have $H(A^d[m,k]) = k$. On the other hand, the set of k elements that maximizes the entropy is $\{Y_1, \ldots, Y_m\}$. Note that $H(Y_i) = k$ and $Y_i \perp Y_j$ for $i \neq j$. Hence, $H(A^c) = k \cdot m$ if $m \geq k$ or otherwise $H(A^c[k]) = k^2$.

Proof of Theorem 4

Let us first mention a slight generalization over the performance of the standard greedy algorithm. It follows easily from the argument in (Nemhauser et al., 1978).

Lemma 18 *Let f be a non-negative submodular function, and let $A^{gc}[q]$ of cardinality q be the greedy selected set by the standard greedy algorithm. Then,*

$$f(A^{gc}[q]) \geq \left(1 - e^{-\frac{q}{k}}\right) f(A^{c}[k]).$$

By Lemma 18 we know that

$$f(A_i^{\mathrm{gc}}[\kappa]) \geq (1 - \exp(-\kappa/k)) f(A_i^{\mathrm{c}}[k]).$$

Now, let us define

$$\begin{aligned} B^{\mathrm{gc}} &= \cup_{i=1}^{m} A_i^{\mathrm{gc}}[\kappa], \\ A_{\max}^{\mathrm{gc}}[\kappa] &= \max_i f(A_i^{\mathrm{gc}}[\kappa]), \\ \tilde{A}[\kappa] &= \arg\max_{S \subseteq B^{\mathrm{gc}} \& |S| \leq \kappa} f(S). \end{aligned}$$

Then by using Lemma 18 again, we obtain

$$\begin{aligned} f(A^{\mathrm{gd}}[m,\kappa]) &\geq \max\left\{f(A_{\max}^{\mathrm{gc}}[\kappa]),\ (1 - \exp(-\kappa/\kappa)) f(\tilde{A}[\kappa])\right\} \\ &\geq \frac{(1 - \exp(-\kappa/k))}{\min(m,k)} f(A^{\mathrm{c}}[k]). \end{aligned}$$

Proof of Proposition 6

Let K be a positive definite kernel matrix defined in section 3.4.1. If we replace a point $e_i \in S$ with another point $e_i' \in V \setminus S$, the corresponding row and column i in the modified kernel matrix K' will be changed. W.l.o.g assume that we replace the first element $e_1 \in S$ with another element $e_1' \in V \setminus S$, i.e., $\Delta K = K' - K$ has the following form with non-zero entries only on the first row and first column,

$$\Delta K \equiv K' - K \leq \begin{pmatrix} a_1 & a_2 & \cdots & a_k \\ a_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_k & 0 & \cdots & 0 \end{pmatrix}.$$

Note that kernel is Lipschitz continuous with constant $\mathcal{L}$, hence we have $|a_i| \leq \mathcal{L} d(e_1, e_1')$ for $1 \leq i \leq k$. Then the absolute value of the change in the objective function would be

$$\begin{aligned} \left|f(S) - f(S')\right| &= \left|\frac{1}{2}\log\det(\mathbf{I} + K') - \frac{1}{2}\log\det(\mathbf{I} + K)\right| \\ &= \frac{1}{2}\left|\log\frac{\det(\mathbf{I} + K')}{\det(\mathbf{I} + K)}\right| \\ &= \frac{1}{2}\left|\log\frac{\det(\mathbf{I} + K + \Delta K)}{\det(\mathbf{I} + K)}\right| \\ &= \frac{1}{2}\left|\log[\det(\mathbf{I} + K + \Delta K).\det(\mathbf{I} + K)^{-1}]\right| \\ &= \frac{1}{2}\left|\log\det(\mathbf{I} + \Delta K(\mathbf{I} + K)^{-1})\right|. \end{aligned} \tag{9}$$

Note that since K is positive-definite, $\mathbf{I}+K$ is an invertible matrix. Furthermore, since ΔK and K are symmetric matrices they both have k real eigenvalues. Therefore, $(\mathbf{I}+K)^{-1}$ has k eigenvalues $\lambda_i = \frac{1}{1+\lambda'_i} \leq 1$, for $1 \leq i \leq k$, where $\lambda'_1 \cdots \lambda'_k$ are (non-negative) eigenvalues of kernel matrix K.

Now, we bound the maximum eigenvalues of ΔK and $\Delta K(\mathbf{I}+K)^{-1}$ respectively. Consider vectors $x, x' \in \mathbb{R}^n$, such that $||x||_2 = ||x'||_2 = 1$. We have,

$$
\begin{aligned}
\left|x^T \Delta K\ x'\right| &= \left| \begin{pmatrix} \\ x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix}^T \begin{pmatrix} a_1 & a_2 & \cdots & a_k \\ a_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_k & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_k \end{pmatrix} \right| \\
&= \left| \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix}^T \begin{pmatrix} \sum_{i=1}^k a_i x'_i \\ a_2 x'_1 \\ \vdots \\ a_k x'_1 \end{pmatrix} \right| \\
&= \left| x_1 \sum_{i=1}^k a_i x'_i + x'_1 \sum_{i=2}^k a_i x_i \right| \\
&= |x_1|. \left| \sum_{i=1}^k a_i x'_i \right| + |x'_1|. \left| \sum_{i=2}^k a_i x_i \right| \\
&= |x_1|. \sum_{i=1}^k |a_i x'_i| + |x'_1|. \sum_{i=2}^k |a_i x_i| \\
&\leq 2k\mathcal{L} d(e_1, e'_1), \qquad (10)
\end{aligned}
$$

where we used the following facts to derive the last inequality: 1) the Lipschitz continuity of the kernel gives us an upperbound on the values of $|a_i|$, i.e., $|a_i| \leq \mathcal{L} d(e_1, e'_1)$ for $1 \leq i \leq k$; and 2) since $||x||_2 = ||x'||_2 = 1$, the absolute value of the elements in vectors x and x' cannot be greater than 1, i.e., $|x_i| \leq 1,\ |x'_i| \leq 1$, for $1 \leq i \leq k$. Therefore,

$$
\lambda_{\max}(\Delta K) = \max_{x:\ ||x||_2 = 1} |x^T \Delta K x| \leq 2k\mathcal{L} d(e_1, e'_1).
$$

Now, let $v_1, \cdots v_k \in \mathbb{R}^n$ be the k eigenvectors of matrix $(\mathbf{I}+K)^{-1}$. Note that $\{v_1, \cdots v_k\}$ is an orthonormal system and thus for any $x \in \mathbb{R}^n$ we can write it as $x = \sum_{i=1}^k c_i v_i$, and

we have $||x||_2^2 = \sum_{i=1}^k c_i^2$. In order to bound the largest eigenvalue of $\Delta K(\mathbf{I}+K)$, we write

$$\begin{aligned}
\left|x^T \Delta K\ (\mathbf{I}+K)^{-1} x\right| &= \left|x^T \Delta K\ (\mathbf{I}+K)^{-1} \sum_{i=1}^k c_i v_i\right| \\
&= \left|x^T \Delta K\ \sum_{i=1}^k \lambda_i c_i v_i\right| \\
&= \left|\left(\sum_{j=1}^k c_j v_j\right)^T \Delta K \left(\sum_{i=1}^k \lambda_i c_i v_i\right)\right| \\
&= \left|\sum_{i,j=1}^k \lambda_i c_i c_j v_j^T \Delta K v_i\right| \\
&\overset{(a)}{\leq} 2k\mathcal{L}d(e_1, e_1') \sum_{i,j=1}^k |c_i||c_j| \\
&= 2k\mathcal{L}d(e_1, e_1') \left(\sum_{i=1}^k |c_i|\right)^2,
\end{aligned}$$

where in (a) we used Eq. 10 and the fact that $\lambda_i \leq 1$ for $1 \leq i \leq k$. Using Cauchy-Schwarz inequality

$$\left(\sum_{i=1}^k |c_i|\right)^2 \leq k \sum_{i=1}^k |c_i|^2$$

and the assumption $||x||_2 = 1$, we conclude

$$\begin{aligned}
\left|x^T \Delta K\ (\mathbf{I}+K)^{-1} x\right| &\leq 2k^2 \mathcal{L}d(e_1, e_1') \sum_{i=1}^k \left|c_i^2\right| \\
&\leq 2k^2 ||x||_2^2 \mathcal{L}d(e_1, e_1') \\
&\leq 2k^2 \mathcal{L}d(e_1, e_1').
\end{aligned}$$

Therefore,

$$\lambda_{\max}\left(\Delta K(\mathbf{I}+K)^{-1}\right) = \max_{x:\ ||x||_2=1} \left|x^T \Delta K\ (\mathbf{I}+K)^{-1} x\right| \leq 2k^2 \mathcal{L}d(e_1, e_1'). \tag{11}$$

Finally, we can write the determinant of a matrix as the product of its eigenvalues, i.e.

$$\det(\mathbf{I} + \Delta K(\mathbf{I}+K)^{-1}) \leq (1 + 2k^2 \mathcal{L}d(e_1, e_1'))^k. \tag{12}$$

By substituting Eq. 11 and Eq. 12 into Eq. 9 we obtain

$$\begin{aligned}
\left|f(S) - f(S')\right| &\leq \frac{1}{2} \left|\log(1 + 2k^2 \mathcal{L}d(e_1, e_1'))^k\right| \\
&\leq \frac{k}{2} \left|\log(1 + 2k^2 \mathcal{L}d(e_1, e_1'))\right| \\
&\leq k^3 \mathcal{L}d(e_1, e_1'),
\end{aligned}$$

where in the last inequality we used $\log(1+x) \leq x$, for $x \geq 0$.
Replacing all the k points in set S with another set S' of the same size, we get

$$\big|f(S) - f(S')\big| \leq k^3 \mathcal{L} \sum_{i=1}^{k} d(e_i, e'_i).$$

Hence, the differential entropy of the Gaussian process is λ-Lipschitz with $\lambda = \mathcal{L}k^3$.

Proof of Proposition 7

Assume we have a set S of k exemplars, i.e., $S_0 = \{e_1, \cdots, e_k\}$, and each element of the dataset $v \in V$ is assigned to its closest exemplar. Now, if we replace set S with another set S' of the same size, the loss associated with every element $v \in V$ may be changed. W.l.o.g, assume we swap one exemplar at a time, i.e., in step $i, 1 \leq i \leq k$, we have $S_i = \{e'_1, \cdots, e'_i, e_{i+1}, \cdots, e_k\}$. Swapping the i^{th} exemplar $e_i \in S_{i-1}$ with another element $e'_i \in S'$, 4 cases may happen: 1) element v was not assigned to e_i before and doesn't get assigned to e'_i, 2) element v was assigned to e_i before and gets assigned to e'_i, 3) element v was not assigned to e_i before and gets assigned to e'_i, 4) element v was assigned to e_i before and gets assigned to another exemplar $e_x \in S_i \setminus \{e'_i\}$. For any element $v \in V$, we look into the four cases and show that in each case

$$|l(e'_i, v) - l(e_i, v)| \leq d(e_i, e'_i)\ \alpha R^{\alpha-1}.$$

- Case 1: In this case, element v was assigned to another exemplar $e_x \in S_i \setminus e_i$ and the assignment doesn't change. Therefore, there is no change in the value of the loss function.

- Case 2: In this case, element v was assigned to e_i before and gets assigned to e'_i. let $a = d(e_i, v)$ and $b = d(e'_i, v)$. Then we can write

$$\begin{aligned} |l(e'_i, v) - l(e_i, v)| &= |a^\alpha - b^\alpha| \\ &= |(a-b)|(a^{\alpha-1} + a^{\alpha-2}b + \cdots + ab^{\alpha-2} + b^{\alpha-1}) \\ &\leq d(e_i, e'_i)\ \alpha R^{\alpha-1}, \end{aligned} \tag{13}$$

  where in the last step we used triangle inequality $|d(e'_t, v) - d(e_t, v)| \leq d(e_t, e'_t)$ and the fact that data points are in a ball of diameter R in the metric space.

- Case 3: In this case, v was assigned to another exemplar $e_x \in S_{i-1} \setminus \{e_i\}$ and gets assigned to e'_i, which implies that $|l(e'_i, v) - l(e_x, v)| \leq |l(e_i, v) - l(e'_i, v)|$, since otherwise e would have been assigned to e_t before.

- Case 4: In the last case, element v was assigned to e_i before and gets assigned to another exemplar $e_x \in S_i \setminus \{e'_i\}$. Thus, we have $|l(e_x, v) - l(e_i, v)| \leq |l(e'_i, v) - l(e_i, v)|$ since otherwise v would have been assigned to e_x before. Hence, in all four cases the following inequality holds:

$$|\min_{e \in S_{i-1}} l(e, v) - \min_{e \in S_i} l(e, v)| \leq |l(e'_i, v) - l(e_i, v)| \leq d(e_i, e'_i)\ \alpha R^{\alpha-1}.$$

By using Eq. 13 and averaging over all elements $v \in V$, we have

$$|L(S_{i-1}) - L(S_i)| = \frac{1}{|V|} \sum_{v \in V} |\min_{e \in S_{i-1}} l(e, v) - \min_{e \in S_i} l(e, v)|$$
$$\leq \alpha R^{\alpha - 1} d(e_i, e'_i).$$

Thus, for any point e_0 that satisfies

$$\max_{v' \in V} l(v, v') \leq l(v, e_0), \quad \forall v \in V \setminus S,$$

we have $L(\{e_0 \cup S\}) = L(\{S\})$ and thus

$$|f(S_{i-1}) - f(S_i)| = |L(\{e_0\}) - L(\{e_0 \cup S_{i-1}\}) - L(\{e_0\}) + L(\{e_0 \cup S_i\})|$$
$$\leq \alpha R^{\alpha - 1} d(e_i, e'_i).$$

Now, if we replace all the k points in set S with another set S' of the same size, we get

$$\big|f(S) - f(S')\big| = \left|\sum_{i=1}^{k} f(S_{i-1}) - f(S_i)\right|$$
$$= \sum_{i=1}^{k} |f(S_{i-1}) - f(S_i)|$$
$$\leq \alpha R^{\alpha - 1} \sum_{i=1}^{k} d(e_i, e'_i).$$

Therefore, for $l = d^{\alpha}$, the loss function is λ-Lipschitz with $\lambda = \alpha R^{\alpha - 1}$.

Proof of Theorem 8

In the following, we say that sets S and S' are γ-close if $|f(S) - f(S')| \leq \gamma$. First, we need the following lemma.

Lemma 19 *If for each $e_i \in A^c[k], |N_\alpha(e_i)| \geq km \log(k/\delta^{1/m})$, and if V is partitioned into sets $V_1, V_2, \ldots V_m$, where each element is randomly assigned to one set with equal probabilities, then there is at least one partition with a subset $A_i^c[k]$ such that $|f(A^c[k]) - f(A_i^c[k])| \leq \lambda\alpha k$ with probability at least $(1 - \delta)$.*

Proof By the hypothesis, the α neighborhood of each element in $A^c[k]$ contains at least $km \log(k/\delta^{1/m})$ elements. For each $e_i \in A^c[k]$, let us take a set of $m \log(k/\delta^{1/m})$ elements from its α-neighborhood. These sets can be constructed to be mutually disjoint, since each α-neighborhood contains $m \log(k/\delta^{1/m})$ elements. We wish to show that at least one of the m partitions of V contains elements from α-neighborhoods of each element.

Each of the $m \log(k/\delta^{1/m})$ elements goes into a particular V_j with a probability $1/m$. The probability that a particular V_j does not contain an element α-close to $e_i \in A^c[k]$ is $\frac{\delta^{1/m}}{k}$. The probability that V_j does not contain elements α-close to one or more of the k

elements is at most $\delta^{1/m}$ (by union bound). The probability that *each* $V_1, V_2, \ldots V_m$ does not contain elements from the α-neighborhood of one or more of the k elements is at most δ. Thus, with high probability of at least $(1-\delta)$, at least one of $V_1, V_2, \ldots V_m$ contains an $A_i^c[k]$ that is $\lambda\alpha k$-close to $A^c[k]$. ■

By lemma 19, for some V_i, $|f(A^c[k]) - f(A_i^c[k]|) \leq \lambda\alpha k$ with the given probability. Furthermore, $f(A_i^{gc}[\kappa]) \geq (1-e^{-\kappa/k}) f(A_i^c[k])$ by Lemma 18. Therefore, the result follows using arguments analogous to the proof of Theorem 4.

Proof of Theorem 9

The following lemma says that in a sample drawn from distribution over an infinite dataset, a sufficiently large sample size guarantees a dense neighborhood near each element of $A^c[k]$ when the elements are from representative regions of the data.

Lemma 20 *A number of elements:* $n \geq \dfrac{8km\log(k/\delta^{1/m})}{\beta g(\alpha)}$, *where* $\alpha \leq \alpha^*$, *suffices to have at least* $4km\log(k/\delta^{1/m})$ *elements in the* α*-neighborhood of each* $e_i \in A^c[k]$ *with probability at least* $(1-\delta)$, *for small values of* δ.

Proof The expected number of α-neighbors of an $e_i \in A^c[k]$, is $E[|N_\alpha(e_i)|] \geq 8km\log(k/\delta^{1/m})$. We now show that in a random set of samples, at least a half of this number of neighbors is realized with high probability near each element of $A^c[k]$.

This follows from a Chernoff bound:

$$P[|N_\alpha(e_i)| \leq 4km\log(k/\delta^{1/m})] \leq e^{-km\log(k/\delta^{1/m})} \leq (\delta^{1/m}/k)^{km}.$$

Therefore, the probability that some $e_i \in A^c[k]$ does not have a suitable sized neighborhood is at most $k(\delta^{1/m}/k)^{km}$. For $\delta \leq 1/k$, $k\delta^{km} \leq \delta^m$. Therefore, with probability at least $(1-\delta)$, the α-neighborhood of each element $e_i \in A^c[k]$ contains at least $4km\log(1/\delta)$ elements. ■

Lemma 21 *For* $n \geq \dfrac{8km\log(k/\delta^{1/m})}{\beta g(\frac{\varepsilon}{\lambda k})}$, *where* $\frac{\varepsilon}{\lambda k} \leq \alpha^*$, *if* V *is partitioned into sets* $V_1, V_2, \ldots V_m$, *where each element is randomly assigned to one set with equal probabilities, then for sufficiently small values of* δ, *there is at least one partition with a subset* $A_i^c[k]$ *such that* $|f(A^c[k]) - f(A_i^c[k])| \leq \varepsilon$ *with probability at least* $(1-\delta)$.

Proof Follows directly by combining Lemma 20 and Lemma 19. The probability that some element does not have a sufficiently dense $\varepsilon/\lambda k$-neighborhood with $km\log(2k/\delta^{1/m})$ elements is at most $(\delta/2)$ for sufficiently small δ, and the probability that some partition does not contain elements from the one or more of the dense neighborhoods is at most $(\delta/2)$. Therefore, the result holds with probability at least $(1-\delta)$. ■

By Lemma 21, there is at least one V_i such that $|f(A^c[k]) - f(A_i^c[k])| \leq \varepsilon$ with the given probability. And $f(A_i^{gd}[\kappa]) \geq (1-e^{-\kappa/k}) f(A_i^c[k])$ using Lemma 18. The result follows using arguments analogous to the proof of Theorem 4.

Proof of Theorem 10

Note that each machine has on the average n/m elements. Let us define Π_i the event that $n/2m < |V_i| < 2n/m$. Then based on the Chernoff bound we know that $\Pr(\neg\Pi_i) \leq 2\exp(-n/8m)$. Let us also define $\xi_i(S)$ the event that $|f_{V_i}(S) - f(S)| < \epsilon$, for some fixed $\epsilon < 1$ and a fixed set S with $|S| \leq k$. Note that $\xi_i(S)$ denotes the event that the empirical mean is close to the true mean. Based on the Hoeffding inequality (without replacement) we have $\Pr(\neq \xi_i S| \leq 2\exp(-2n\epsilon^2/m)$. Hence,

$$\Pr(\xi_i(S) \wedge \Pi_i) \geq 1 - 2\exp(-2n\epsilon^2/m) - 2\exp(-n/8m).$$

Let ξ_i be an event that $|f_{V_i}(S) - f(S)| < \epsilon$, for any S such that $|S| \leq \kappa$. Note that there are at most n^κ sets of size at most κ. Hence,

$$\Pr(\xi_i \wedge \Pi_i) \geq 1 - 2n^\kappa(\exp(-2n\epsilon^2/m) - \exp(-n/8m)). \tag{14}$$

As a result, for $\epsilon < 1/4$ we have

$$\Pr(\xi_i \wedge \Pi_i) \geq 1 - 4n^\kappa \exp(-2n\epsilon^2/m).$$

Since there are m machines, by the union bound we can conclude that

$$\Pr((\xi_i \wedge \Pi_i) \text{ on all machines}) \geq 1 - 4mn^\kappa \exp(-2n\epsilon^2/m).$$

The above calculation implies that we need to choose $\delta \geq 4mn^\kappa \exp(-2n\epsilon^2/m)$. Let n_0 be chosen in a way that for any $n \geq n_0$ we have $\ln(n)/n \leq \epsilon^2/(mk)$. Then, we need to choose n as follows:

$$n = \max\left(n_0, \frac{m\log(\delta/4m)}{\epsilon^2}\right).$$

Hence for the above choice of n, there is at least one V_i such that $|f(A^c[k]) - f(A_i^c[\kappa])| \leq \varepsilon$ with probability $1 - \delta$. Hence the solution is ϵ away from the optimum solution with probability $1 - \delta$. Now if we confine the evaluation of $f(A_i^c)$ to data point only in machine i then under the assumption of Theorem 9 we lose another ϵ. Formally, the result at this point simply follows by combining Theorem 4 and Theorem 9.

Proof of Theorem 12

The proof is similar to the proof of Theorem 3 and Theorem 4 and follows from the following lemmas.

Lemma 22 $\max_i f(A_i^c[\zeta]) \geq \frac{1}{m} f(A^c[\zeta])$.

Proof Let B_i be the elements in V_i that are contained in the optimal solution, $B_i = A^c[\zeta] \cap V_i$. Since $A^c[\zeta] \in \zeta$ and ζ is a set of hereditary constraints, we must have $B_i \in \zeta$ as well. Using submodularity of f and by the same argument as in the proof of Lemma 16, we have

$$\begin{aligned} f(A^c[\zeta]) = f(B_1 \cup \cdots \cup B_m) &= f(B_1) + f(B_2|B_1) + \cdots + f(B_m|B_{m-1}, \cdots, B_1) \\ &\leq f(B_1) + \cdots + f(B_m). \end{aligned}$$

Since $f(A_i^c[\zeta]) \geq f(B_i)$ we get

$$f(A^c[\zeta]) \leq f(A_1^c[\zeta]) + \cdots + f(A_m^c[\zeta]) \leq m \max_i f(A_i^c[\zeta]).$$

■

Lemma 23 $\max_i f(A_i^c[\zeta]) \geq \frac{1}{k} f(A^c[\zeta])$.

Proof The proof follows the outline of the proof of Lemma 17. Let $f(A^c[\zeta]) = f(\{u_1, \cdots, u_{\rho([\zeta])}\})$. Since $A^c[\zeta] \in \zeta$ and ζ is a set of hereditary constraints, we have $u_i \in \zeta$. Using submodularity of f, we have

$$f(A^c[\zeta]) \leq \sum_{i=1}^{\rho([\zeta])} f(u_i) \leq \rho([\zeta]) f(u^*).$$

where $u^* = \arg\max_i f(u_i)$. Suppose that $u^* \in V_j$, we get

$$f(\max_i f(A_i^c[\zeta])) \geq f(A_j^c[\zeta]) \geq f(u^*) \geq \frac{1}{\rho([\zeta])} f(A^c[\zeta]).$$

■

Since $f(A^d[m, \rho([\zeta])]) \geq \max_i f(A_i^c[\zeta])$; from Lemma 23 and 22 we have

$$f(A^d[m, \rho([\zeta])]) \geq \frac{1}{\min(m, \rho([\zeta]))} f(A^c[\zeta]). \tag{15}$$

For the black box algorithm X with a τ-approximation guarantee, we have

$$f(A_i^X[\zeta]) \geq \tau f(A_i^c[\zeta]).$$

Now, we generalize the definitions used in the proof of Theorem 4

$$\begin{aligned} B^{gc} &= \cup_{i=1}^m A_i^{gc}[\zeta], \\ A_{\max}^{gc}[\zeta] &= \max_i f(A_i^{gc}[\zeta]), \\ \tilde{A}[\zeta] &= \arg\max_{S \subseteq B^{gc} \& |S| \leq \rho([\zeta])} f(S). \end{aligned}$$

Then using Eq. 15 again, we obtain

$$\begin{aligned} f(A^{gd}[m, \zeta]) &\geq \max\left\{ f(A_{\max}^{gc}[\zeta]),\ \tau f(\tilde{A}[\zeta]) \right\} \\ &\geq \frac{\tau}{\min(m, \rho([\zeta]))} f(A^c[\zeta]). \end{aligned}$$

Note that since we do not use monotonicity of the submodular function in any of the proofs, the results hold in general for constrained maximization of any non-negative submodular function.

Proof of Theorem 13

Lemma 24 *If for each $e_i \in A^X[\zeta], |N_\alpha(e_i)| \geq \rho([\zeta])m\log\left(\rho([\zeta])/\delta^{1/m}\right)$, and if V is partitioned into sets $V_1, V_2, \ldots V_m$, where each element is randomly assigned to one set with equal probabilities, then there is at least one partition with a subset $A_i^X[\zeta] \in \zeta$ such that $\left|f(A^c[\zeta]) - f(A_i^X[\zeta])\right| \leq \lambda\alpha\rho([\zeta])$ with probability at least $(1-\delta)$.*

The proof is similar to the proof of Lemma 19 by taking disjoint sets of size $m\log\left(\rho([\zeta])/\delta^{1/m}\right)$ in an α-neighborhood of each $e_i \in A^c[\zeta]$ and showing that with high probability, at least one of the m partitions of V contains elements from α-neighborhoods of each element in the optimal solution. Note that now the size of the optimal solution is at most $\rho([\zeta])$. Since ζ is locally replaceable with parameter α, as elements of $A^c[\zeta]$ gets replaced by nearby elements in their α-neighborhood, the resulting set is also a feasible solution.

By Lemma 24, for some V_i, $\left|f(A^c[\zeta]) - f(A_i^X[\zeta])\right| \leq \lambda\alpha\rho([\zeta])$ with the given probability. On the other hand, for the black box algorithm X, we have $f(A_i^X[\zeta]) \geq \tau f(A_i^c[\zeta])$. Therefore, the result follows using arguments analogous to the proof of Theorem 12.

Proof of Theorem 14

We use the following Lemmas to show that in a sample drawn from a ddistribution over an infinite dataset, a sufficiently large sample size guarantees a dense neighborhood near each element of the optimal solution.

Lemma 25 *A number of elements: $n \geq \dfrac{8\rho([\zeta])m\log\left(\rho([\zeta])/\delta^{1/m}\right)}{\beta g(\alpha)}$, where $\alpha \leq \alpha^*$, suffices to have at least $4\rho([\zeta])m\log\left(\rho([\zeta])/\delta^{1/m}\right)$ elements in the α-neighborhood of each $e_i \in A^c[\zeta]$ with probability at least $(1-\delta)$, for small values of δ.*

Lemma 26 *For $n \geq \dfrac{8\rho([\zeta])m\log(\rho([\zeta])/\delta^{1/m})}{\beta g(\frac{\varepsilon}{\lambda\rho([\zeta])})}$, where $\frac{\varepsilon}{\lambda\rho([\zeta])} \leq \alpha^*$, if V is partitioned into sets $V_1, V_2, \ldots V_m$, where each element is randomly assigned to one set with equal probabilities, then for sufficiently small values of δ, there is at least one partition with a subset $A_i^c[\zeta]$ such that $|f(A^c[\zeta]) - f(A_i^c[\zeta])| \leq \varepsilon$ with probability at least $(1-\delta)$.*

The proofs follows the same arguments as in the proof of Lemma 20 and 21. Recall that, by assumption ζ is locally replaceable with parameter α. Hence, for $\varepsilon \leq \alpha\lambda\rho([\zeta])$, any set ε-close to the optimal solution is also a feasible solution.

By Lemma 26, there is at least one V_i such that $|f(A^c[\zeta]) - f(A_i^c[\zeta])| \leq \varepsilon$ with the given probability. Furthermore, for the black box algorithm X, we have $f(A_i^{gd}[\zeta]) \geq \tau f(A_i^c[\zeta])$. Thus the result follows using arguments analogous to the proof of Theorem 12.

Proof of Theorem 15

Again the proof follows the same line of reasoning as the proof of Theorem 10, except that for a constraint set ζ with $\rho([\zeta]) = \max_{S\in\zeta}|S|$, there are at most $n^{\rho([\zeta])}$ feasible solutions. Using the same definitions for Π_i and $\mathcal{E}_i$ as in the proof of Theorem 10, instead of Eq. 14 we get

$$\Pr(\xi_i \wedge \Pi_i) \geq 1 - 2n^{\rho([\zeta])}(\exp(-2n\epsilon^2/m) - \exp(-n/8m)).$$

As a result, for $\epsilon < 1/4$ and using union bound we conclude that

$$\Pr((\xi_i \wedge \Pi_i) \text{ on all machines}) \geq 1 - 4mn^{\rho([\zeta])} \exp(-2n\epsilon^2/m).$$

which implies that we need to choose $\delta \geq 4mn^{\rho}([\zeta]) \exp(-2n\epsilon^2/m)$. Now if n_0 be chosen in a way that for any $n \geq n_0$ we have $\ln(n)/n \leq \epsilon^2/(mk)$, we get $n \geq \max(n_0, m\log(\delta/4m)/\epsilon^2)$.

Bearing in mind that ζ is locally replaceable, there is at least one V_i such that the solution $A_i^c[\zeta]$ is feasible and ϵ away from the optimum solution with probability $1-\delta$. Now under the assumption of Theorem 14, if we evaluate $f(A_i^c)$ only on machine i, then we lose another ϵ. Now by combining Theorem 12 and Theorem 14 we get the desired result.

References

Yahoo! academic relations. r6a, yahoo! front page today module user click log dataset, version 1.0, 2012. URL `http://Webscope.sandbox.yahoo.com`.

Zeinab Abbassi, Vahab S Mirrokni, and Mayur Thakur. Diversity maximization under matroid constraints. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 32–40. ACM, 2013.

Mahmoudreza Babaei, Baharan Mirzasoleiman, Mahdi Jalili, and Mohammad Ali Safari. Revenue maximization in social networks through discounting. *Social Network Analysis and Mining*, 3(4):1249–1262, 2013.

Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1497–1514. SIAM, 2014.

Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 671–680. ACM, 2014.

Rafael Barbosa, Alina Ene, Huy Nguyen, and Justin Ward. The power of randomization: Distributed submodular maximization on massive datasets. *Proceedings of The 32nd International Conference on Machine Learning*, pages 1236–1244, 2015.

Guy E Blelloch, Richard Peng, and Kanat Tangwongsan. Linear-work greedy parallel approximate set cover and variants. In *Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures*, pages 23–32. ACM, 2011.

Ferenc Bodon. Kosarak dataset, 2012. URL `http://fimi.ua.ac.be/data/`.

Niv Buchbinder, Michael Feldman, Joseph Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 649–658. IEEE, 2012.

Niv Buchbinder, Moran Feldman, Joseph Seffi Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1433–1452. SIAM, 2014.

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

Flavio Chierichetti, Ravi Kumar, and Andrew Tomkins. Max-cover in map-reduce. In *Proceedings of the 19th international conference on World wide web*, pages 231–240. ACM, 2010.

Cheng Chu, Sang Kyun Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Andrew Y Ng, and Kunle Olukotun. Map-reduce for machine learning on multicore. *Advances in neural information processing systems*, 19:281, 2007.

Michele Conforti and Gérard Cornuéjols. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the rado-edmonds theorem. *Discrete applied mathematics*, 7(3):251–274, 1984.

Graham Cormode, Howard Karloff, and Anthony Wirth. Set cover algorithms for very large datasets. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 479–488. ACM, 2010.

Sven De Vries and Rakesh V Vohra. Combinatorial auctions: A survey. *INFORMS Journal on computing*, 15(3):284–309, 2003.

Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

Nan Du, Yingyu Liang, Maria Florina Balcan, and Le Song. Budgeted influence maximization for multiple products. *arXiv preprint arXiv:1312.2164*, 2013.

Delbert Dueck and Brendan J Frey. Non-metric affinity propagation for unsupervised image categorization. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

Jaliya Ekanayake, Shrideep Pallickara, and Geoffrey Fox. Mapreduce for data intensive scientific analyses. In *eScience, 2008. eScience'08. IEEE Fourth International Conference on*, pages 277–284. IEEE, 2008.

Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

Marshall L. Fisher, George L. Nemhauser, and Laurence A. Wolsey. An analysis of approximations for maximizing submodular set functions - II. *Mathematical Programming Study*, (8):73–87, 1978.

Rahul Garg, Vijay Kumar, and Vinayaka Pandit. Approximation algorithms for budget-constrained auctions. In *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, pages 102–113. Springer, 2001.

Karolien Geurts, Geert Wets, Tom Brijs, and Koen Vanhoof. Profiling of high-frequency accident locations by use of association rules. *Transportation Research Record: Journal of the Transportation Research Board*, 1840(1):123–130, 2003.

Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1098–1116. SIAM, 2011.

Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, pages 427–486, 2011.

Daniel Golovin, Matthew Faulkner, and Andreas Krause. Online distributed sensor selection. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 220–231. ACM, 2010.

Manuel Gomez Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1019–1028. ACM, 2010.

Andrew Guillory and Jeff Bilmes. Active semi-supervised learning using submodular functions. In *Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11)*, pages 274–282. AUAI, 2011.

Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *Internet and Network Economics*, pages 246–257. Springer, 2010.

Jason Hartline, Vahab Mirrokni, and Mukund Sundararajan. Optimal marketing strategies over social networks. In *Proceedings of the 17th international conference on World Wide Web*, pages 189–198. ACM, 2008.

Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for mapreduce. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 938–948. Society for Industrial and Applied Mathematics, 2010.

Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

Chun-Wa Ko, Jon Lee, and Maurice Queyranne. An exact algorithm for maximum entropy sampling. *Operations Research*, 43(4):684–691, 1995.

Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability: Practical Approaches to Hard Problems*, 3:19, 2012.

Andreas Krause and Ryan G Gomes. Budgeted nonparametric learning from data streams. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 391–398, 2010.

Andreas Krause and Carlos Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, page 5, 2005a.

Andreas Krause and Carlos Guestrin. A note on the budgeted maximization on submodular functions. Technical Report CMU-CALD-05-103, Carnegie Mellon University, 2005b.

Andreas Krause and Carlos Guestrin. Submodularity and its applications in optimized information gathering. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(4):32, 2011.

Alex Kulesza. Determinantal point processes for machine learning. *Machine Learning*, 5 (2-3):123–286, 2012.

Ariel Kulik, Hadas Shachnai, and Tami Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 545–554. Society for Industrial and Applied Mathematics, 2009.

Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. Fast greedy algorithms in mapreduce and streaming. In *Proceedings of the 25th ACM symposium on Parallelism in algorithms and architectures*, pages 1–10. ACM, 2013.

Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in mapreduce. In *Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures*, pages 85–94. ACM, 2011.

Jon Lee, Vahab S Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 323–332. ACM, 2009a.

Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 244–257. Springer, 2009b.

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429. ACM, 2007.

Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics, 2011.

Odile Macchi. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, pages 83–122, 1975.

Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, pages 234–243. Springer, 1978.

Vahab Mirrokni and Morteza Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC '15, pages 153–162. ACM, 2015.

Baharan Mirzasoleiman, Mahmoudreza Babaei, and Mahdi Jalili. Immunizing complex networks with limited budget. *EPL (Europhysics Letters)*, 98(3):38004, 2012.

Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems*, pages 2049–2057, 2013.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrak, and Andreas Krause. Lazier than lazy greedy. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015a.

Baharan Mirzasoleiman, Amin Karbasi, Ashwinkumar Badanidiyuru, and Andreas Krause. Distributed submodular cover: Succinctly summarizing massive data. In *Advances in Neural Information Processing Systems*, 2015b.

Ramasuri Narayanam and Amit A Nanavati. Viral marketing for product cross-sell through social networks. In *Machine Learning and Knowledge Discovery in Databases*, pages 581–596. Springer, 2012.

George L Nemhauser and Leonard A Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3):177–188, 1978.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1): 265–294, 1978.

Tore Opsahl and Pietro Panzarasa. Clustering in weighted networks. *Social networks*, 31 (2):155–163, 2009.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

Matthew Streeter, Daniel Golovin, and Andreas Krause. Online learning of assignments. In *Advances in Neural Information Processing Systems*, pages 1794–1802, 2009.

Maxim Sviridenko. A note on maximizing a submodular set function subject to knapsack constraint. *Operations Research Letters*, 32, 2004.

Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.

Athanasios Tsanas, Max Little, Patrick E McSharry, Lorraine O Ramig, et al. Enhanced classical dysphonia measures and sparse regression for telemonitoring of parkinson's disease progression. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 594–597. IEEE, 2010.

Kai Wei, Yuzong Liu, Katrin Kirchhoff, and Jeff Bilmes. Using document summarization techniques for speech data subset selection. In *Proceedings of NAACL-HLT*, pages 721–726, 2013.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. Fast multi-stage submodular maximization. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1494–1502, 2014.