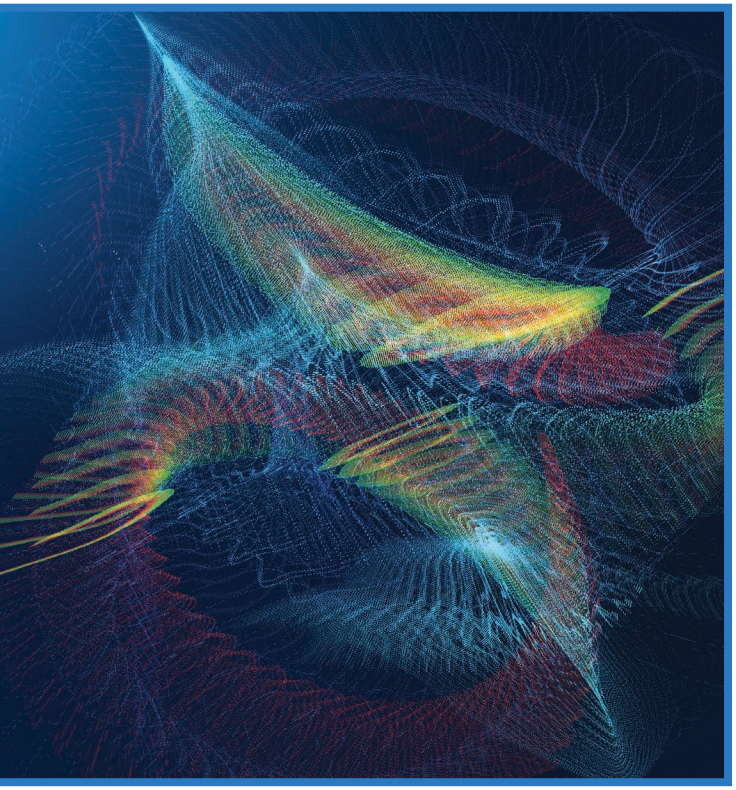
Ehsan Tohidi, Rouhollah Amiri, Mario Coutino, David Gesbert, Geert Leus, and Amin Karbasi

Submodularity in Action

From machine learning to signal processing applications



©ISTOCKPHOTO.COM/IN-FUTURE

Digital Object Identifier 10.1109/MSP.2020.3003836
Date of current version: 2 September 2020

Submodularity is a discrete domain functional property that can be interpreted as mimicking the role of well-known convexity/concavity properties in the continuous domain. Submodular functions exhibit strong structure that lead to efficient optimization algorithms with provable near-optimality guarantees. These characteristics, namely, efficiency and provable performance bounds, are of particular interest for signal processing (SP) and machine learning (ML) practitioners, as a variety of discrete optimization problems are encountered in a wide range of applications. Conventionally, two general approaches exist to solve discrete problems: 1) relaxation into the continuous domain to obtain an approximate solution or 2) the development of a tailored algorithm that applies directly in the discrete domain. In both approaches, worst-case performance guarantees are often hard to establish. Furthermore, they are often complex and thus not practical for large-scale problems. In this article, we show how certain scenarios lend themselves to exploiting submodularity for constructing scalable solutions with provable worst-case performance guarantees. We introduce a variety of submodular-friendly applications and elucidate the relation of submodularity to convexity and concavity, which enables efficient optimization. With a mixture of theory and practice, we present different flavors of submodularity accompanying illustrative real-world case studies from modern SP and ML. In all of the cases, optimization algorithms are presented along with hints on how optimality guarantees can be established.

Introduction

Discrete optimization

Discrete optimization is a notoriously challenging problem that occurs in countless engineering applications and particularly in SP and ML. Discrete optimization usually involves finding a solution in some finite or countably infinite set of potential solutions that maximizes/minimizes an objective function. A counterintuitive phenomenon is that discrete problems are sometimes more difficult than their continuous counterparts. This phenomenon was perfectly illustrated by Welsh

[1]: "mathematical generalization often lays bare the important bits of information about the problem at hand."

In general, discrete optimization problems are tackled in two common ways: 1) building a continuous relaxation or 2) working out a tailored algorithm. The first relaxes the original problem to a continuous one so as to apply tools from continuous optimization. The continuous solution is then made discrete by a rounding technique to obtain an approximate feasible solution for the original problem. The second approach develops a customized algorithm for use directly in the discrete domain. Beyond the fact that we often need creative endeavors to design these algorithms, both approaches have the following shortcomings: 1) for most problems, it is hard to have a gap bound between the approximate and optimal solutions (OPTs), i.e., approximation guarantees; and 2) although the proposed algorithms are usually solvable in polynomial time, they are not necessarily scalable.

The submodularity property enables striking algorithm-friendly features and is observed in a good number of application scenarios.

Submodularity: A useful property for optimization

In this article, we depict a family of optimization scenarios and seek solutions that circumvent the aforementioned fundamental limitations. Although it is challenging to find optimality bounds and provide low-complexity methods that will address all of the discrete optimization scenarios, in this article, we highlight a particular structure that is relevant to a surprisingly large class of problems. Submodularity is a functional property that has recently gained significant attention. The submodularity property enables striking algorithm-friendly features and is observed in a good number of application scenarios. In fact, due to their specific connections with convexity, the exact minimization of submodular functions can be done efficiently while greedy algorithms can obtain near-optimality guarantees in submodular-maximization problems thanks to their relationship with concavity. These benefits have drawn attention in many different contexts [2]–[4]. Sample applications include sensor selection [5], detection [6], resource allocation [7], active learning [8], interpretability of neural networks (NNs) [9], and adversarial attacks [10].

This article takes an illustrative approach to explain the concept of submodularity and its promise in modern SP and ML applications. The relationship of submodularity with the celebrated properties of convexity/concavity is introduced to motivate the algorithmic landscape for optimizing submodular functions. Starting with basic structures, we give a presentation of different aspects of submodular functions and cover current extensions of submodularity. Also, to clarify the importance of submodularity in the context of SP and ML, various applications are showcased as examples and their connections to different aspects of submodularity are highlighted.

Submodularity essentials

In this section, we explain the mathematical formulation of submodularity through the following illustrative example.

Motivating example: Sensing coverage problem

We illustrate the concept of submodularity through the so-called sensing coverage problem. Assume that there exists a set of candidate locations of sensors—called the *ground set*—with effective areas that determine the area around each sensor from where fruitful field measurements can be obtained (see Figure 1). Because these sensors are costly, we should optimize the deployment locations to maximize their effectiveness. Therefore, the optimization problem turns into a coverage problem where the objective function is to maximize the covered area of the sensing field subject to a limited number of selected sensors. Here, we investigate one important property of this objective function.

Figure 1 depicts two sample subsets of the ground set (i.e., the blue circles) where the sensors in the smaller subset are all included in the larger one. We have added the same sensor (i.e., the red circle) to both subsets. As shown in Figure 1, the new covered area due to the newly added sensor is smaller for the larger subset. We often call this phenomenon *diminishing returns* (*DRs*) or *decreasing gain*. For this specific example, this property implies that the new covered area added due to the addition of a new sensor may decrease when the size of the original subset increases. As will be seen next, this example carries the essential intuition behind submodularity.

Submodular set functions

A set function $f: 2^{\mathcal{N}} \to \mathbb{R}$ defined over a ground set $\mathcal{N}$ is submodular if, for every $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{N}$ and $e \in \mathcal{N} \backslash \mathcal{B}$, it holds that

$$f(e \mid \mathcal{A}) \geq f(e \mid \mathcal{B}), \tag{1}$$

where $f(e \mid \mathcal{S}) = f(\mathcal{S} \cup \{e\}) - f(\mathcal{S})$ is defined as the discrete derivative (or marginal gain) of the set function f at $\mathcal{S}$ with respect to e. Although there are several equivalent definitions

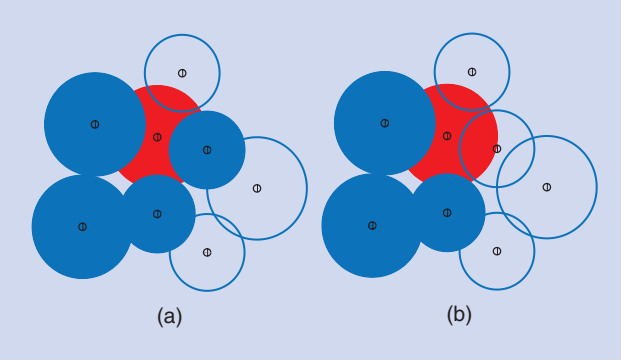


FIGURE 1. An illustration of submodularity for the sensing coverage problem. The figure depicts an example of adding the same sensor (i.e., the red circle) to (a) a large subset and (b) a smaller subset, where the selected subsets of sensors are shown by the blue circles. The amount of added coverage is larger in (b).

of submodularity, we stick to the definition in (1) as it naturally displays the DRs' property of submodular set functions. Moreover, function f is supermodular if $-f$ is submodular, and a function is modular if it is both submodular and supermodular.

Aside from submodularity, another common property of some set functions is monotonicity. A set function $f: 2^{\mathcal{N}} \to \mathbb{R}$ is monotone if for every $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{N}$, $f(\mathcal{A}) \leq f(\mathcal{B})$. This property is exhibited by many commonly encountered set functions in practice. For instance, in the previous example, the coverage is monotone in the number of sensors.

Note that although there is a substantial body of literature that addresses submodular functions with nonmonotone behavior [11], in this article, we mainly consider normalized monotone submodular functions for ease of exposition. Here, a normalized set function implies $f(\varnothing) = 0$. In the following sections, we present crucial properties linked with submodularity as well as extensions that find applications in both SP and ML.

When we add elements to a set, the elements that are added in the beginning are more important than the elements that are included later.

DRs and concavity: Maximization

In this section, we focus on the link between submodularity and concavity, which can be leveraged for designing algorithms for the maximization of submodular functions.

The definition (1) exhibits the notion of diminishing marginal gains, which naturally leads to considering submodularity as a discrete analog of concavity. As depicted in Figure 2(a), the derivative of a concave function does not increase by increasing the input variable. Although less explicit, a similar characteristic appears in submodular functions. In Figure 2(b), a hypercube is shown where each vertex corresponds to a subset of the ground set, and a line connects two vertices if one subset contains the other. Here, the arrows represent the direction from the smaller to the larger subset. Similar to the concavity in the continuous domain, moving along an arrow, the discrete derivative does not increase.

Greedy maximization of submodular functions

Consider now the problem of submodular function maximization, which can be written in its most general form as

$$\max_{\mathcal{S} \subseteq \mathcal{N}} \; f(\mathcal{S}) \text{ subject to some constraints on } \mathcal{S}, \tag{2}$$

where $f(\cdot)$ is a submodular function, and the common valid constraints are cardinality, knapsack, multipartition, and matroid (more details of which are given later). Equation (2) appears in many applications; for example, consider the previous sensing coverage problem (compare Figure 1). In that setting, our goal is to deploy up to K sensors, i.e., $|\mathcal{S}| \leq K$, such that the coverage area, i.e., $f(\mathcal{S})$, is maximized. Finding the OPT of such a problem is NP-hard [12]. Therefore, it is reasonable to look for efficient algorithms that achieve a near-OPT.

Greedy algorithm

A simple approach for solving the problem of normalized monotone submodular function maximization under a cardinality constraint is the greedy algorithm, which starts with an empty subset $\mathcal{S}$, and in each iteration, adds element e, which maximizes the marginal gain $f(e \mid \mathcal{S})$, i.e.,

$$\mathcal{S} \leftarrow \mathcal{S} \cup \{\operatorname*{argmax}_{e \in \mathcal{N} \setminus \mathcal{S}} f(e \mid \mathcal{S})\}. \tag{3}$$

This simple algorithm is guaranteed to achieve the OPT with a factor of $1 - 1/e$ [12]. It is also known that one cannot achieve a better approximation guarantee under natural computational-theoretic assumptions. Even though obtaining this performance guarantee with such a simple algorithm is surprising, there is a good explanation. The definition (1) is based on DRs. Simply put, when we add elements to a set, the elements that are added in the beginning are more important than the elements that are included later. Now, a greedy algorithm only optimizes over the next element and does not consider later elements. As such, it gives the largest weight to the next element. This observation intuitively explains that the basis of a greedy algorithm is matched with the concept of submodularity, and this is the main reason for achieving a noticeable performance by employing such an algorithm with a low computational complexity.

Submodularity and concavity

One way to establish a connection between submodularity and concavity is through the use of a so-called multilinear extension [13]. For set function $f: 2^{\mathcal{N}} \to \mathbb{R}$, its multilinear extension $F_M: [0,1]^N \to \mathbb{R}$ with $|\mathcal{N}| = N$ is defined by

$$F_M(\mathbf{x}) = \mathbb{E}\{f(\mathcal{X})\} = \sum_{\mathcal{S} \subseteq \mathcal{N}} f(\mathcal{S}) \prod_{i \in \mathcal{S}} x_i \prod_{j \in \mathcal{N} \setminus \mathcal{S}} (1 - x_j), \tag{4}$$

with $\mathcal{X}$ being a random set where elements appear independently with probabilities x_i. It is proved that if $f(\cdot)$ is submodular, then $F_M(\cdot)$ is concave along positive directions. Namely, if $\mathbf{x} \leqslant \mathbf{y}$, then $\nabla F_M(\mathbf{x}) \geqslant \nabla F_M(\mathbf{y})$, where the inequality should be considered element-wise [13], and ∇ stands for the gradient operation.

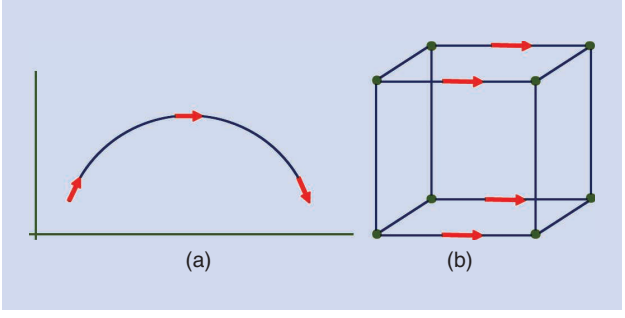


FIGURE 2. An illustration of decreasing derivatives in both concave and submodular functions. (a) and (b) depict a concave function in the continuous domain and a submodular function in the discrete domain, respectively.

Using the restricted concavity of the multilinear extension, that is, the continuous version of the greedy algorithm [compare (3)], the so-called continuous greedy can solve certain types of constrained-maximization problems near optimally [14].

Constrained submodular maximization

In many applications, the elements of the ground set may have nonuniform costs, e.g., some sensors might be more expensive to deploy than others, and the problem may be constrained on a budget that the total cost cannot exceed or, a certain structure has to be enforced in the solution set. For example, consider a set of heterogeneous sensors (such as acoustic, optical, and radar), which provide a given sensing coverage and have different operative/deployment costs. We aim to deploy a number of these sensors to maximize the total coverage while meeting a budgetary requirement. Such a problem is called a *knapsack problem* and appears often in SP and ML applications. This problem can be formulated as

$$\max_{\mathcal{S}\subseteq\mathcal{N}} \quad f(\mathcal{S}) \quad \text{s.t.} \quad \sum_{e\in\mathcal{S}} c_e \le B, \tag{5}$$

where c_e is the cost of element $e\in\mathcal{N}$ and B is a given budget, determining the maximum sum-cost of elements in $\mathcal{S}$.

Cost-weighted greedy algorithm

The optimization problem (5) is well solved with a modified version of the greedy algorithm that takes cost into account. This algorithm greedily generates a solution, substituting the update selection rule (3) by

$$\mathcal{S} \leftarrow \mathcal{S} \cup \left\{ \operatorname*{argmax}_{e\in\mathcal{J}} \frac{f(e\,|\,\mathcal{S})}{c_e} \right\}, \tag{6}$$

where $\mathcal{J}=\{e\,|\,e\in\mathcal{N}\backslash\mathcal{S}, c_e \le B - c_{\mathcal{S}}\}$. Next to this set, another greedy set is constructed using the rule (3). It is shown that constructing both sets and selecting the best one provides a $(1-1/e/2)$-approximation guarantee [15]. A later study showed that a more involved version of this procedure achieves a $(1-1/\sqrt{e})$-approximation guarantee [16]. Further, if the partial enumeration of all the feasible sets of cardinality one or two is performed, a version of the cost-weighted greedy algorithm leads to a $(1-1/\sqrt{e})$-approximation guarantee [17].

Although knapsack-type constraints are pervasive, they only model the weights (or cost) associated with the elements of the ground set. Fortunately, there exist other structures that are able to capture more complex and practical constraints while allowing for the near-optimal maximization of submodular functions. In the following sections, we introduce these structures along with their corresponding greedy algorithms.

Matroids: Useful combinatorial structures

A matroid is a useful combinatorial structure that generalizes the concept of linear independence in linear algebra to set theory. Matroids and submodular functions are closely related. Specifically, each matroid corresponds to a submodular rank function.

A matroid is defined as a pair $(\mathcal{N},\mathcal{I})$ in which $\mathcal{N}$ is a finite set and $\mathcal{I}\subseteq 2^{\mathcal{N}}$ comprises any subset of $\mathcal{N}$, which satisfies the following properties:

- $\mathcal{A}\subseteq\mathcal{B}\subseteq\mathcal{N}$ and $\mathcal{B}\in\mathcal{I}$ imply $\mathcal{A}\in\mathcal{I}$.
- $\mathcal{A},\mathcal{B}\in\mathcal{I}$ and $|\mathcal{B}|>|\mathcal{A}|$ imply that $\exists e\in\mathcal{B}\backslash\mathcal{A}$ such that $\mathcal{A}\cup\{e\}\in\mathcal{I}$.

Based on this definition, the sets in $\mathcal{I}$ are called *independent*.

Examples of matroids

In the following, we list some examples of commonly encountered matroids in SP and ML applications.

- *Graphic matroid*: Consider a graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$ with a vertex set $\mathcal{V}$ and an edge set $\mathcal{E}$, and let $\mathcal{I}(\mathcal{G})$ be the set of all edge subsets that do not contain a cycle of $\mathcal{G}$; that is, the elements of the independent sets are the forests of the graph. Then, $(\mathcal{E},\mathcal{I}(\mathcal{G}))$ forms a matroid.
- *Uniform matroid*: Another common example of a matroid is given by a cardinality constraint, i.e., $|\mathcal{S}|\le K$. The related matroid is constructed by considering as independent sets all the subsets of $\mathcal{N}$ with at most k elements, i.e., $\mathcal{I}=\{\mathcal{S}\subset\mathcal{N}:|\mathcal{S}|\le k\}$. This matroid is referred to as the *uniform matroid* of rank k, $\mathcal{U}_N^k$.
- *Partition matroid*: Given a collection of I disjoint sets $\{C_i\subset\mathcal{N}\}_{i=1}^I$ and integers $\{b_i\}_{i=1}^I$ such that $0\le b_i\le|C_i|, \forall i\in\{1,\ldots,I\}$. Then, the independent sets of a partition matroid are given by $\mathcal{I}=\{\mathcal{S}\subseteq\mathcal{N}: |\mathcal{S}\cap C_i|\le b_i, \forall i\, 1\le i\le I\}$.

Matroid-aware greedy algorithm

Given the matroid $(\mathcal{N},\mathcal{I})$, the constrained submodular-maximization problem $\max_{\mathcal{S}\in\mathcal{I}} f(\mathcal{S})$ can be near-optimally solved by constructing a solution using the rule

$$\mathcal{S} \leftarrow \mathcal{S} \cup \{ \operatorname*{argmax}_{e\notin\mathcal{S}\,:\,\mathcal{S}\cup\{e\}\in\mathcal{I}} f(e\,|\,\mathcal{S})\}, \tag{7}$$

until there is no more candidate element that can be added to form a feasible solution. It is shown that the matroid-aware greedy method can achieve a one-half near-optimality guarantee [12].

The $(1-1/e)$-approximation guarantee can be achieved if the continuous greedy algorithm in [14] is used instead. This method constructs a solution by appropriately rounding the solution of a continuous relaxation using the multilinear extension of the original problem. In the following sections, we present an application that makes use of matroids to model commonly encountered constraints.

Resource selection for parameter estimation in multiple-input, multiple-output radars

The application of multiple-input, multiple-output (MIMO) radar systems becomes pervasive because of their enormous advantages over conventional radars. Such large-scale MIMO systems are, however, very expensive to implement in practice due to the high increase in hardware cost regarding the deployment of multiple sensors, their power consumption for multipulse transmissions, and their processing complexity. To reduce

the aforementioned costs while at the same time guaranteeing a given estimation accuracy level, it is important to select only a limited set of transmitters, pulses, and receivers (as shown in the MIMO radar configuration of Figure 3) that are the most informative for the parameter estimation task. Such a problem is known as *resource selection* in the literature. In [5], the problem of resource selection in an MIMO radar is formulated as maximizing a submodular function subject to a partition matroid $(\mathcal{P} \cup \mathcal{R}, \mathcal{I})$ whose independent sets are defined as

$$\mathcal{I} = \{\mathcal{S} : |\mathcal{S} \cap \mathcal{P}| \leq K_P, |\mathcal{S} \cap \mathcal{R}| \leq K_R\}, \tag{8}$$

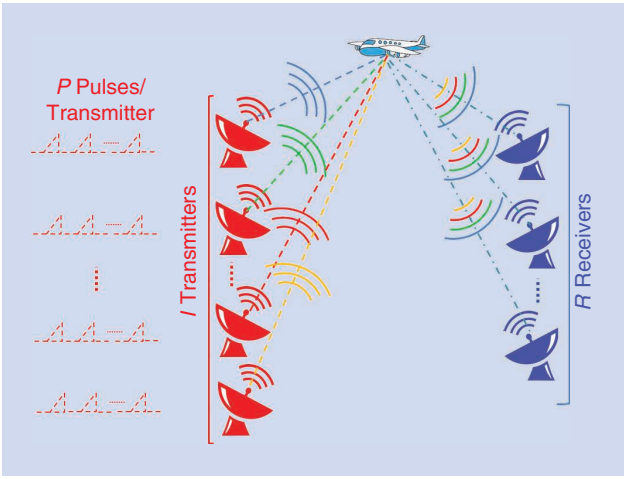


FIGURE 3. The configuration of a MIMO radar system.

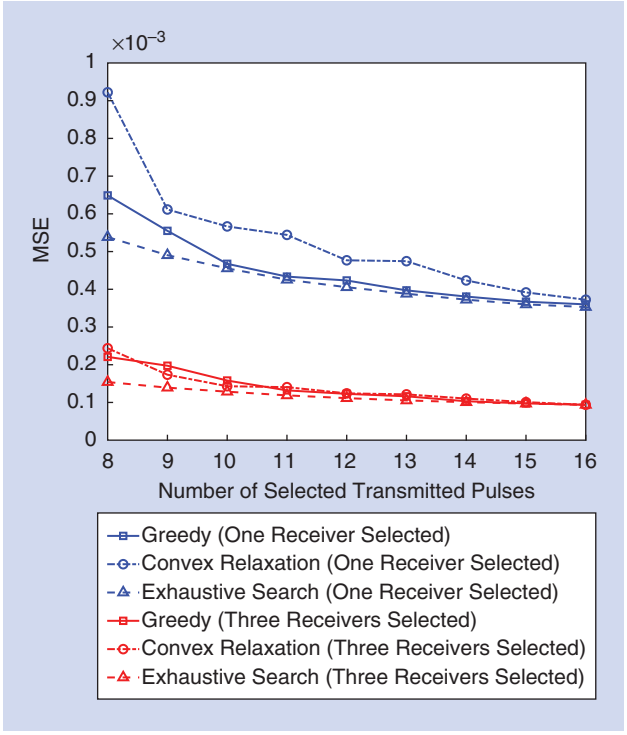


FIGURE 4. The MSE as a function of the selected transmitted pulses for two different cases with one and three selected receivers [5].

where $\mathcal{P}$ and $\mathcal{R}$ are the ground sets of all the transmitted pulses and receivers, respectively. Variables K_P and K_R stand for cardinality of the selected sets of pulses and receivers, respectively.

To illustrate this application, we consider a simulation scenario with four receivers, four transmitters, and four pulses per transmitter. To evaluate the performance of the greedy selection algorithm in comparison with the convex method and the optimum mean square error (MSE) obtained through an exhaustive search, the estimation MSE is plotted in Figure 4 as a function of the number of selected transmitted pulses. The results are presented for two cases where one and three receivers should be selected. As shown in Figure 4, the estimation accuracy of the greedy algorithm is very close to the optimal value. Furthermore, its performance is the same or better than its convex counterpart while having a much lower complexity.

The key to the problem is the objective function, which should be related to the estimation accuracy. Because MSE is neither convex nor submodular and makes the optimization task difficult, in [5], a surrogate objective function is incorporated that measures the orthogonality between the vectors of a frame. This measurement is an appropriate submodular proxy for the MSE in nonlinear estimation problems.

Multiway partitions

In addition to matroids, multiway partitions are amenable structures for submodular optimization. Multiway partitioning arises in a diverse range of combinatorial optimization problems in areas including communications and SP. This problem is defined as partitioning a given set $\mathcal{S}$ into k disjoint subsets $\mathcal{S}_1, \ldots, \mathcal{S}_k, \mathcal{S}_i \cap \mathcal{S}_j = \emptyset, \cup_{i=1}^{k} \mathcal{S}_i = \mathcal{S}$ such that $\Sigma_{i=1}^{k} f_i(\mathcal{S}_i)$ is maximized, where the f_is are arbitrary submodular functions.

Multiway greedy partition algorithm

To solve the multiway-partitioning problem, a greedy algorithm can be used that allocates each element e to the best subset at that point, i.e., $\mathcal{S}_{j^*} \leftarrow \mathcal{S}_{j^*} \cup \{e\}$, where $j^* = \text{argmax}_j f_j(\mathcal{S}_j \cup \{e\})$. If all of the involved functions in the multiway-partitioning problem are nonnegative, monotone, and submodular, then the greedy algorithm provides a one-half-approximation guarantee [12]. In the following, we illustrate this problem via a resource-allocation application.

Water filling-based resource allocation

Consider an orthogonal frequency-division multiple access (OFDMA) communication system with a set of n orthogonal subcarriers, denoted by C. The considered problem is to allocate a disjoint subset of C to each user so as to maximize the sum-rate criterion [7]. Denoting the set of allocated subcarriers to user i by $\mathcal{A}_i \subseteq C$, the resource-allocation problem can be formulated as the following partitioning problem [7]:

$$\max_{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m} \sum_{i=1}^{m} R_i(\mathcal{A}_i),$$
$$\text{s.t.} \bigcup_{i=1}^{m} \mathcal{A}_i = C, \mathcal{A}_i \cap \mathcal{A}_j = \emptyset, \tag{9}$$

where

$$R_i(\mathcal{A}_i) = \max_{\substack{\sum_{j\in\mathcal{A}_i} P_{i,j}\le P_i \\ P_{i,j}\ge 0}} \sum_{j\in\mathcal{A}_i} \log\left(1+\frac{P_{i,j}}{N_{i,j}}\right), \tag{10}$$

with P_i being the sum-power constraint for user i, $P_{i,j}$ the power allotted by user i to subcarrier j, and $N_{i,j}$ the corresponding channel noise level. The rate (10) has a similar form as that of the water-filling function [7] and thus, the power for each user can be allocated locally via a water-filling algorithm. Furthermore, it is proved that the water-filling function is submodular [7] (because a sum of submodular functions is a submodular function [4], the objective function in (9) is submodular) and thus, a greedy algorithm can be employed to efficiently solve the subcarrier allocation problem with a theoretical guarantee [12].

For the presented application, an illustrative example is provided in Figure 5 to demonstrate how a greedy resource-allocation algorithm works. As presented in Figure 5(a)–(f), starting from the first subcarrier and going through them one by one, at each step, the subcarrier is allocated to the user with the maximum marginal gain, and power is reallocated afterward for the new set of subcarriers based on the water-filling algorithm.

Approximate (weak) submodularity

When a cost function f is not submodular, greedy algorithms can still be useful; yet, they do not necessarily provide any theoretical guarantees. Fortunately, for cases where f is close to submodular, it can be shown that greedy algorithms degrade gracefully [18].

To measure how far function f is from being submodular, the concept of weak submodularity is introduced in [18]. This notion of approximate submodularity is linked with other notions like ϵ-submodularity (see, e.g., [19]) and to properties such as restricted strong convexity [20]. Mathematically, weak submodularity is defined through the submodularity ratio, γ, as follows: a monotone nonnegative set function $f: 2^N \to \mathbb{R}_+$ is called *γ-weakly submodular* for an integer r if

$$\gamma \le \gamma_r := \min_{\mathcal{L},\mathcal{S}\subseteq\mathcal{N}:|\mathcal{L}|,|\mathcal{S}\setminus\mathcal{L}|\le r} \frac{\sum_{j\in\mathcal{S}\setminus\mathcal{L}} f(\{j\}\mid\mathcal{L})}{f(\mathcal{S}\mid\mathcal{L})}, \tag{11}$$

where 0/0 is defined as 1. This definition generalizes submodularity by relaxing the DRs' property. It can be easily shown that function f is submodular if and only if $\gamma_r \ge 1$ for all r. Whenever γ is bounded away from 0, the greedy algorithm guarantees a solution with a $(1-e^{-\gamma})$-approximation guarantee under a cardinality constraint of size r, which is the best achievable performance, as discussed in [21]. To explore other variants and guarantees for the greedy maximization of approximately submodular functions, we refer the reader to [18].

Subset selection for regression

To show how the approximate submodularity framework can be useful in practice, we consider the problem of subset selection for regression [18], [20]. That is, given a set of n regressors, select a subset of k regressors that best predicts the variables of interest. The applications of this problem range from feature selection to sparse learning in both SP and ML. The advantage of using the natural combinatorial formulation of the problem—based on weak submodularity—over traditional convex relaxations [22] is that it provides direct control over the sparsity level k and avoids the tuning of regularization parameters.

Consider a set of observation variables $\mathbf{x} = [x_1, \ldots, x_n]^{\mathrm{T}}$, and a predictor variable z. Further, let $[\mathbf{C}]_{i,j} = \mathrm{Cov}(x_i, x_j)$ and $[\mathbf{b}]_i = \mathrm{Cov}(x_i, z)$ be the covariances among the observations and between the predictor and observation, respectively. Then, the square multiple correlation with respect to a subset of variables $\mathcal{S}$ is given by

$$R^2_{z,\mathcal{S}} := \mathbf{b}_{\mathcal{S}}^{\top}\mathbf{C}_{\mathcal{S}}^{-1}\mathbf{b}_{\mathcal{S}}, \tag{12}$$

where the subscript indicates either that only the entries or row and column indices in $\mathcal{S}$ are retained. Hence, given both $\mathbf{C}$ and $\mathbf{b}$ and k, the subset selection problem is posed as the maximization of $R^2_{z,\mathcal{S}} : |\mathcal{S}| \le k$. This setting is similar to that of [6], where the approximate submodularity of the signal-to-noise ratio [the same functional form as (12)] is leveraged during sensor selection for the detection of signals under Gaussian noise.

Using approximate submodularity, it can be shown that celebrated greedy algorithms, such as forward regression (FR) and orthogonal matching pursuit (OMP), obtain near-optimality guarantees for this family of problems. For these two algorithms, their submodularity ratios are given by

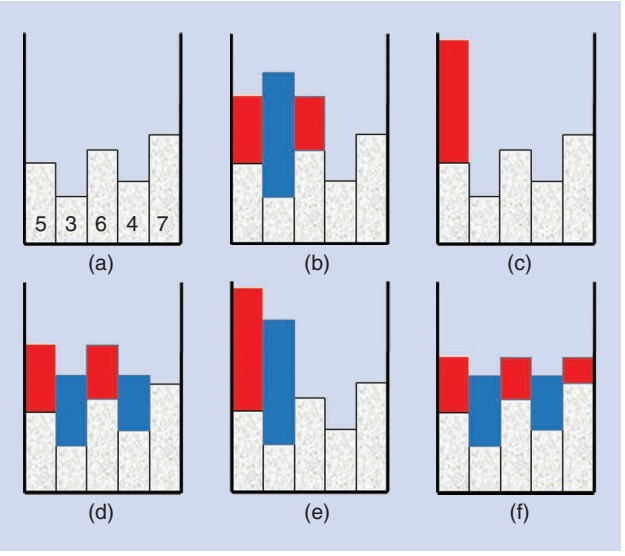


FIGURE 5. An example of an uplink OFDMA subcarrier and power-allocation problem with two users and five subcarriers. We consider 8 W for the sum-power constraint of each user and both users are assumed to experience the same noise variance in each subcarrier. (a) The noise variance of different subcarriers. Starting from the first channel and allocating the channels one by one, (b)–(f) present the procedure of subcarrier allocation to users, where the red/blue bars represent the allocated power by the first/second user to each subcarrier, respectively.

$\gamma_{\mathrm{FR}} = \lambda(4\mathbf{C}, 2k)$ and $\gamma_{\mathrm{OMP}} = \lambda(\mathbf{C}, 2k)^2$, respectively. Here, $\lambda(\mathbf{C}, k) := \min_{\mathcal{S}:|\mathcal{S}|=k} \lambda_{\min}(\mathbf{C}_{\mathcal{S}})$, where $\lambda_{\min}(\mathbf{A})$ is the minimum eigenvalue of the matrix $\mathbf{A}$.

To illustrate this application, a comparison of these two algorithms is conducted against the OPT and the oblivious greedy and Lasso algorithms. The theoretical results (compare the submodularity ratios) predict that FR should outperform OMP in most of the cases. In Figure 6, the $R^2_{z,\mathcal{S}}$ values for the selected subsets by different methods for sizes $k \in \{2, \ldots, 8\}$ are displayed for two different data sets.

(Weakly) adaptive submodularity

Several problems in SP and ML require making decisions in a sequence, such as terrain exploration or movie recommendations based on a user's feedback. The difficulty of such problems lies not only in the size of the search space but also in the partial knowledge of the process, i.e., decisions are made under the uncertainty of the future. As tackling these problems without considering any further structure is either challenging or intractable, the concept of submodularity has been extended to this setting to find tractable methods with strong theoretical guarantees [8].

To model adaptability, we can make use of a (directed) graph $G = (\mathcal{V}, \mathcal{E})$ whose vertices relate to the elements of the ground set, i.e., there is a bijection between the vertices of G and the elements of set $\mathcal{N}$; and its weighted (directed) edges capture the probabilities of selecting one of the elements after the other (if this information is available). Adaptability arises when nodes (or edges) are allowed to have states. That is, their properties, e.g., the cost, signals on the nodes (edges), and so on can, change as a selection procedure (the creation of a subset of elements) progresses [23].

For instance, let us consider recommending websites to a user. Here, set $\mathcal{N}$ represents websites, and graph G (and its weights) can be constructed based on the traffic (cross references) between these websites. Further, each website is assumed to have, at any given time, one of the states $\mathcal{Q} := \{\text{unvisited}, \text{visited}\}$. Finally, a function $h : 2^{|\mathcal{V}|} \times \mathcal{Q}^{|\mathcal{V}|} \mapsto \mathbb{R}_+$ is selected to measure user satisfaction, e.g., the affinity for visited websites with dynamic user preferences. Here, adaptability is required because at each new recommendation, the states of the websites change.

To extend submodularity notions to an adaptive setting, the conditional marginal gain of a set $\mathcal{A} \subseteq \mathcal{N}$, with respect to the function h, is defined as

$$\Delta(\mathcal{A} \,|\, \psi) = \mathbb{E}\left[h(\mathrm{dom}(\psi) \cup \mathcal{A}, \phi) - h(\mathrm{dom}(\psi), \phi) \,\middle|\, \psi\right], \quad (13)$$

where ψ denotes a partial realization, i.e., a mapping disclosing the states of a subset of nodes (edges), and the expectation is taken over all of the full realizations ϕ, that is, a complete disclosure of node (edges) states such that $\mathrm{dom}(\psi) \subseteq \mathrm{dom}(\phi)$. Here, $\mathrm{dom}(\psi)$ denotes the list of items whose state is known. Using this definition, set function h is weakly adaptive and set submodular with parameter γ if, for all sets $\mathcal{A} \subseteq \mathcal{V}$, we have

$$\gamma \leq \frac{\sum_{e \in \mathcal{A}} \Delta(e \,|\, \psi)}{\Delta(\mathcal{A} \,|\, \psi')} \; \forall \psi \subseteq \psi', \quad (14)$$

where $\psi \subseteq \psi'$, if and only if $\mathrm{dom}(\psi) \subseteq \mathrm{dom}(\psi')$ and they are equal in the domain of ψ.

This notion is a natural extension of the submodularity ratio [compare (11)] to the adaptive setting. Notice that instead of conditioning only on sets, conditioning on realizations is

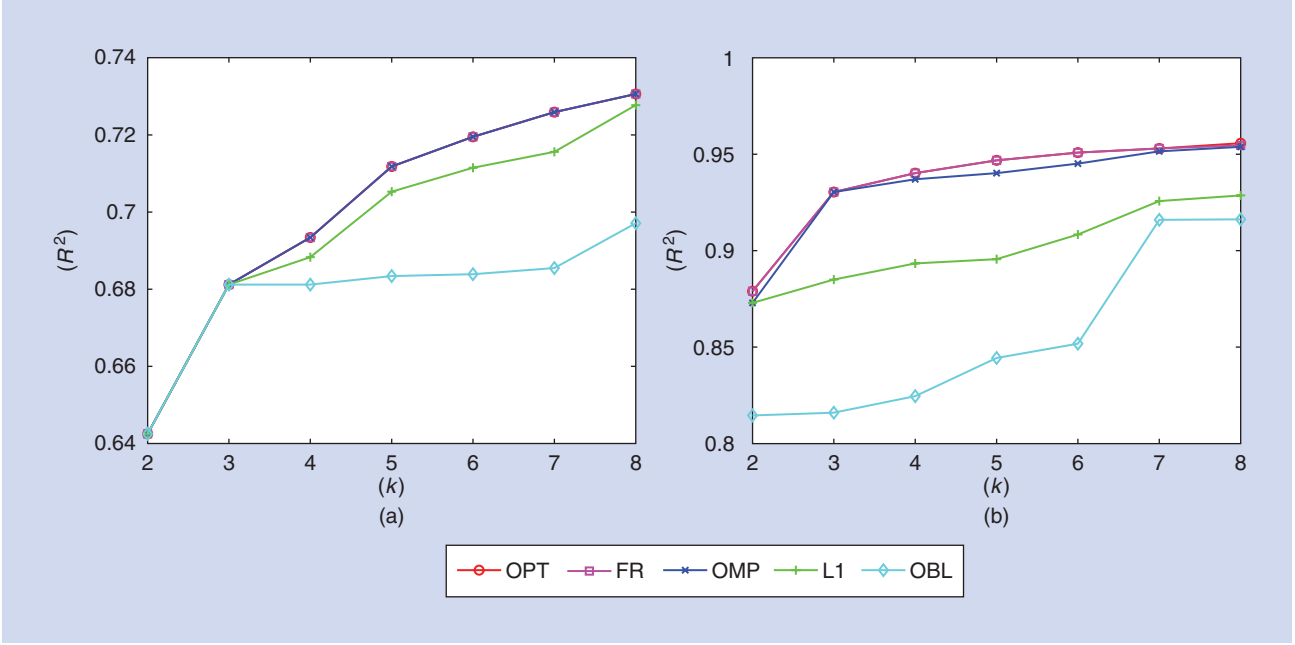


FIGURE 6. The regression results for two different data sets. (a) The Boston Housing data set $R^2_{z,\mathcal{S}}$ and (b) the World Bank data set $R^2_{z,\mathcal{S}}$ [18]. OBL: oblivious greedy algorithm; L1: Lasso algorithm.

needed to account for the element states. Note that an edge function can be defined in a similar way to h.

Under this setting, it has been shown that an adaptive version of the greedy algorithm [23], that is, the adaptive sequence greedy (ASG) method, returns set σ_{ASG} such that

$$f_{\text{avg}}(\sigma_{\text{ASG}}) \geq \frac{\gamma}{2d_{\text{in}}+\gamma} f_{\text{avg}}(\sigma^*), \tag{15}$$

where $f_{\text{avg}}(\sigma) := \mathbb{E}[h(\mathcal{E}(\sigma), \phi)]$, d_{in} is the largest in-degree of input graph G, and σ^* is the set obtaining the highest expected value [compare $f_{\text{avg}}(\cdot)$].

Wikipedia link search

To illustrate an application of adaptive sequence submodularity, we present the problem of an adaptive-article sequence recommendation. Here, a user is surfing Wikipedia toward some target article. And given her history of previously visited links, we aim to guide her to the target article. Because the order in which she visits the articles is critical, a set of articles does not suffice as an answer, and an ordered sequence is needed.

The ASG method comes with provable guarantees on its performance and does not require hyperparameter tuning nor retraining.

Under this setting, the sequence value is encoded through the weights of a directed graph $G = (\mathcal{V}, \mathcal{E})$ where each element of the ground set is represented by a vertex in $\mathcal{V}$. The probability of moving from the ith to the jth link is captured by the weight w_{ij} of the corresponding directed edge. As a result, a sequence of elements $\sigma := \{\sigma_1, \ldots, \sigma_i, \sigma_{i+1}, \ldots\}$, i.e., an ordered set of elements, induces a set of edges $\mathcal{E}(\sigma) := \{(\sigma_i, \sigma_j) \mid (\sigma_i, \sigma_j) \in \mathcal{E}, i \leq j\}$. In addition, each node is assumed to have two states: "1" if the user visits a page and "0" if the user does not want to visit it. This last feature, plus the fact that the decision must be made based on the current page that the user is visiting, is what makes adaptability necessary. For this problem, the probabilistic coverage utility function [23], i.e.,

$$h(\mathcal{E}(\sigma)) = \sum_{j \in \mathcal{V}} \left[1 - \prod_{(i,j) \in \mathcal{E}(\sigma)} (1 - w_{ij})\right], \tag{16}$$

is used to guide the selection problem.

A comparison of the ASG method with deep learning-based alternatives is presented in Figure 7. The results report the relevance of the final output page to the true target page, i.e., a higher relevance is related to a lower score. Notice that the ASG method outperforms the deep learning alternatives, as under this setting, we suffer from data scarcity. Also, the ASG method comes with provable guarantees on its performance and does not require hyperparameter tuning nor retraining. These are in contrast with deep learning approaches, which do not have theoretical guarantees; however, they do require parameter tuning and, when the ground set is changed, need to be retrained.

Distributed submodular maximization

As explained thus far, although submodularity enables us to employ conceptually low-complexity algorithms with theoretical approximation bounds, classical approaches of submodular optimization require access to the full data set, which is impractical in large-scale problems. MapReduce is a fruitful programming model used for reliable and efficient parallel processing that has demonstrated a promising approach for designing parallel, submodular optimization algorithms, particularly for forming a small representative subset from a large data set [24], [25].

In a distributed setting, we assume m machines are given to carry out the submodular optimization problem with two considerations: 1) the optimality of the returned solution and 2) the communication complexity, i.e., the number of synchronizations among machines. One can imagine that, without any constraint on the number of synchronizations, we can technically perform a centralized scenario. Consequently, the following three questions arise when tackling distributed optimization: 1) how to distribute items among machines, 2) what algorithms to run across different machines in a parallel fashion, and 3) how to merge/synchronize the results of different machines [24].

A two-round parallel protocol that provides efficient responses to these three questions is proposed in [24]. Here, we briefly explain the algorithm and its results for a monotone submodular function with a cardinality constraint (for more general cases, see [24]). In the initialization phase, the data set is arbitrarily partitioned into m sets—one set for each machine. Next, in the first round, given K as the cardinality constraint, each machine executes a greedy algorithm over its own set to achieve a subset with K elements. Then, in the second round, the m subsets obtained from all of the machines are shared with a central node to form a superset with mK elements. Running a standard greedy algorithm over this superset leads to a new subset with K elements. Finally, among the $m + 1$ subsets with K elements, the one that maximizes the utility function is selected. It is shown that this algorithm provides a $\left((1 - 1/\text{e})/\sqrt{\min(m, K)}\right)$-approximation guarantee [26]. Figure 8 depicts an illustration of the two-round algorithm in [24]. Active set selection in Gaussian processes and large-scale exemplar-based clustering are two instances of applications where the size of the data sets often requires a distributed method for a given submodular-maximization problem [24].

Submodularity and convexity: Minimization

Up to this point, we have only discussed the maximization of submodular functions. In this section, we highlight an interesting connection between submodular set functions and convexity, which allows for the efficient minimization of submodular set functions.

A convex extension of submodular functions

Consider a set function $f : 2^{\mathcal{N}} \to \mathbb{R}$ with $|\mathcal{N}| = N$. Associate every element of the set $2^{\mathcal{N}}$ to a vertex of the hypercube

$\{0,1\}^N$. That is, each $\mathcal{S} \subseteq \mathcal{N}$ corresponds uniquely to a binary vector of length N where the nth entry is 1 if $n \in \mathcal{S}$ and 0 otherwise.

A set function f can be extended from the discrete domain (vertices of the hypercube) to the continuous domain (the complete hypercube) through the Lovász extension. The Lovász extension $F_L : \mathbb{R}^N \to \mathbb{R}$, is defined as follows: given a vector $\mathbf{x} \in [0,1]^N$ and a scalar $\theta \in [0,1]$, define $\mathcal{T}_\theta = \{e \in \mathcal{N} \,|\, x_e \geq \theta\}$, where x_e is the eth entry of $\mathbf{x}$. Then, $F_L(\mathbf{x})$ is obtained through the following equation [27]:

$$F_L(\mathbf{x}) = \mathop{\mathbb{E}}_{\theta \in [0,1]} \{f(\mathcal{T}_\theta)\}. \tag{17}$$

Figure 9 illustrates an example of how the hypercube is formed, and it is divided into six parts corresponding to the six possible orderings of the input vector for the Lovász extension.

The Lovász extension of a submodular function has two main properties [27]: first, it is convex; and second, its minimizer resides at the vertices of the hypercube. Hence, convex optimization can be employed to efficiently find a feasible minimizer of the original discrete problem.

We note that, in both Lovász and multilinear extensions, the distribution defining the continuous function at $\mathbf{x}$ is independent of the set function $f(\cdot)$. Also, these extensions are similar in the sense that both are obtained by taking the expectation of the function, however, with respect to different probability measures.

Unconstrained submodular minimization

Consider an unconstrained submodular minimization problem, i.e., $\min_{S \subseteq \mathcal{N}} f(\mathcal{S})$. If f is extended to F_L, the continuous function is convex and exact (i.e., we can recover an optimal set of the original problem from an OPT of the

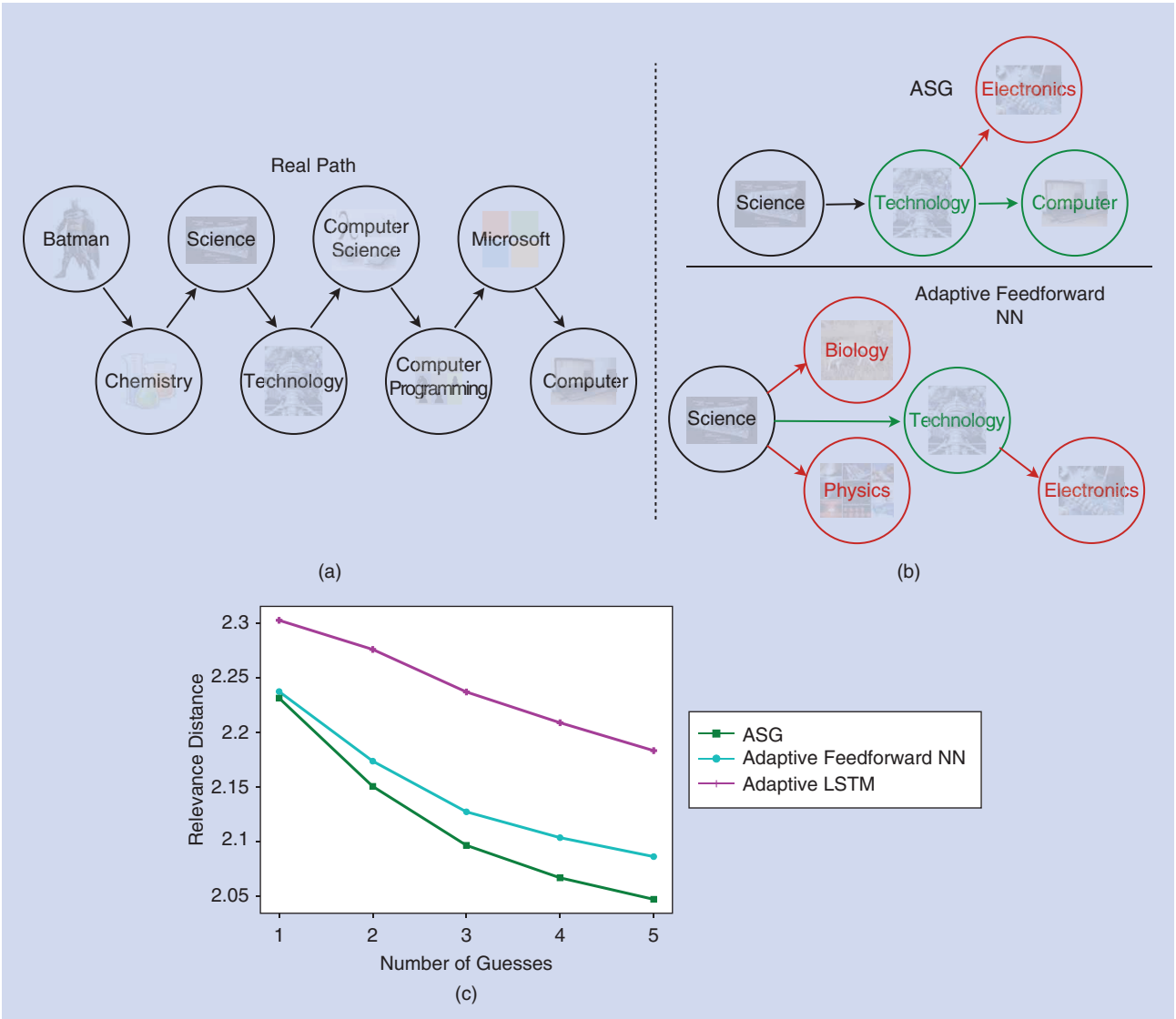


FIGURE 7. (a) The real and (b) predicted page paths. (c) The overall performance of the compared methods using the relevance distance [23]. LSTM: long short-term memory.

extended problem). Therefore, solving $\min_{\mathbf{x}\in[0,1]^N} F_L(\mathbf{x})$ by a subgradient method, we can retrieve the OPT for the original discrete problem.

Alternatively, using the dual formulation of the continuous problem, we can obtain a minimum norm problem that can be solved by the Frank–Wolfe algorithm [28]. This algorithm, also known as the *conditional gradient method*, is an iterative first-order method that considers a linear approximation of the objective function and moves toward its minimizer. The Frank–Wolfe algorithm is a projection-free method, and it is well known for keeping the sparsity of the proposed solution. These aspects make this algorithm

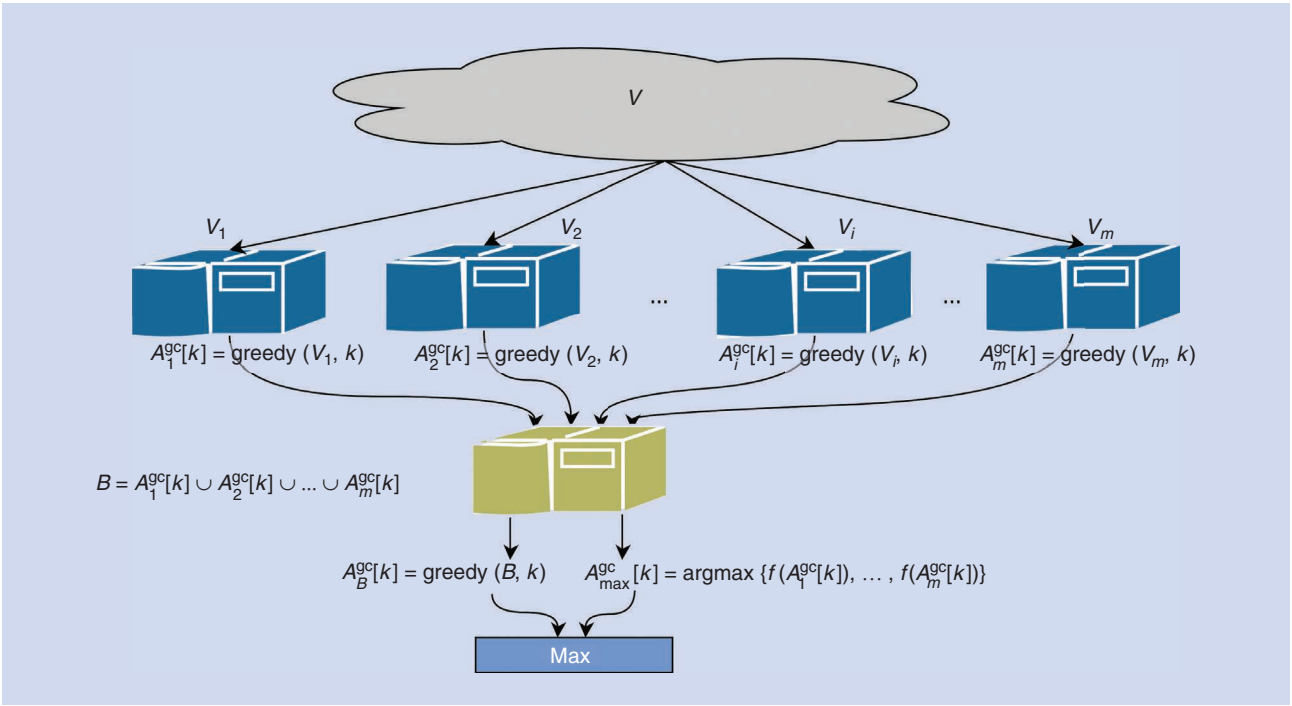


FIGURE 8. An illustration of the distributed two-round algorithm. V is the ground set and $V_1, \ldots, V_m$ are the initial partitions distributed among different machines with m being the number of machines. Moreover, $A_1^{\text{gc}}, \ldots, A_m^{\text{gc}}$ are the greedy solutions obtained locally on machines in the first round where the one with the maximum utility is denoted as $A_{\text{max}}^{\text{gc}}$. Then, in the second round, m subsets obtained in all the machines are shared with a central node to form a superset B with mK elements, i.e., $B = A_1^{\text{gc}} \cup \cdots \cup A_m^{\text{gc}}$. Furthermore, running a standard greedy algorithm over the superset B leads to A_B^{gc}. Finally, the maximum of $A_{\text{max}}^{\text{gc}}$ and A_B^{gc} is returned as the solution [24].

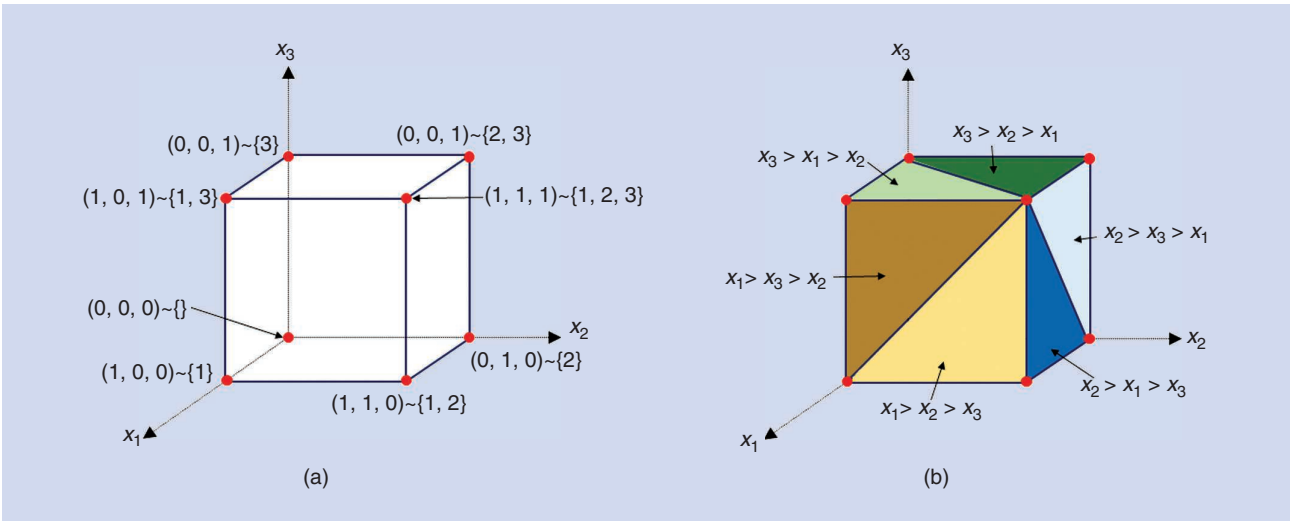


FIGURE 9. From subsets to vertices in the hypercube for a sample ground set $\mathcal{N} = \{1,2,3\}$ with $N = 3$. Each subset has a corresponding vertex in the hypercube. (a) A depiction of how the hypercube is created. (b) The six possible orderings of the input vector for the Lovász extension [3].

attractive for sparse and large-scale constrained optimization problems, for instance, for optimizing over atomic domains [29].

Hardness of constrained submodular minimization

Although there exist polynomial-time algorithms for minimizing any unconstrained submodular function, constrained submodular minimization becomes challenging to approximate under simple constraints. For instance, in [30], it is shown that for problems such as submodular load balancing [given a monotone submodular function f and a positive integer M, find a partition of $\mathcal{N}$ into M sets, $\mathcal{N}_1,\ldots,\mathcal{N}_M$, so as to minimize $\max_m f(\mathcal{N}_m)$] and submodular sparsest cut [given a set of unordered pairs $\{\{x_m,y_m\}\,|\,x_m,y_m\in\mathcal{N}\}$, each with a demand of $p_m>0$, find a subset $\mathcal{S}\subseteq\mathcal{N}$ minimizing $(f(\mathcal{S})/\Sigma_{m:|\mathcal{S}\cap\{x_m,y_m\}|=1}p_m)$], the approximation guarantees cannot be better than $O(\sqrt{N/\ln N})$, where N is the size of the ground set.

Therefore, even though polynomial-time algorithms are available for the minimization of submodular functions, we must be aware of the following issues: 1) the high computational complexity of algorithms required to minimize unconstrained problems, i.e., the polynomial orders are typically larger than three, making the exact minimization challenging for large-scale problems; and 2) dealing with constraints makes the problem extremely hard.

Optimization of general set functions

At this point, readers may wonder: Is it possible to develop a suitable greedy algorithm for any arbitrary set function? Unfortunately, the answer is no, in general. However, using the fact that any set function can be expressed as a difference of two submodular set functions [31], greedy algorithms inspired by optimization methods for the difference of two convex functions [32] can be developed.

The maximization of any set function $f:2^{\mathcal{N}}\to\mathbb{R}$ defined over a ground set $\mathcal{N}$ can be expressed as the difference of two submodular set functions $g:2^{\mathcal{N}}\to\mathbb{R}$ and $h:2^{\mathcal{N}}\to\mathbb{R}$:

$$\max_{\mathcal{S}\subseteq\mathcal{N}} f(\mathcal{S}) \equiv \max_{\mathcal{S}\subseteq\mathcal{N}} [g(\mathcal{S})-h(\mathcal{S})]. \tag{18}$$

This formulation allows for drawing parallels with convex optimization techniques and to devise a greedy algorithm to approximate the solution. Specifically, when $h(\mathcal{S})$ is modular, a recent result shows that we can get a $(1-1/\mathrm{e})\,g(\mathrm{opt})-h(\mathrm{opt})$ approximation guarantee [21].

Supermodular-submodular procedure

Similar to maximizing the difference of convex functions, we can consider an approach that approximates the solution of the original problem by a sequence of submodular-maximization problems. Recall that in the convex-concave procedure [32], the concave function is approximated at every step by its first-order Taylor expansion.

Following this idea, the problem of maximizing the difference of submodular set functions is cast as the sequential maximization of submodular functions. This is done by substituting the second submodular set function in (18) with its modular upper bound in each iteration [33]. A number of tight, modular upper bounds $\mathcal{J}^h_{\mathcal{S}}$ are suggested in [12], e.g.,

$$\mathcal{J}^h_{\mathcal{S}}(\mathcal{X}) = h(\mathcal{S}) - \sum_{e\in\mathcal{S}\setminus\mathcal{X}} h(\{e\}\,|\,\mathcal{S}\setminus\{e\}) + \sum_{e\in\mathcal{X}\setminus\mathcal{S}} h(\{e\}\,|\,\varnothing). \tag{19}$$

This method, called a *supermodular-submodular* (*SupSub*) procedure, starts with an empty set $\mathcal{S}_0$, and in each iteration, k tries to solve the following problem (specialized for a K-cardinality constraint):

$$\mathcal{S}_{k+1} = \operatorname*{argmax}_{\mathcal{S}\subseteq\mathcal{N},|\mathcal{S}|=K} g(\mathcal{S}) - \mathcal{J}^h_{\mathcal{S}_k}(\mathcal{S}) \tag{20}$$

until a convergence condition is satisfied.

Note that to solve the maximization problem at each step of this algorithm, which is NP-hard in general, the greedy heuristic can be used to obtain a near-OPT. Despite the fact that near-optimality guarantees are not available for the SupSub procedure, similar to the convex-concave procedure, this method is guaranteed to converge to a local minima [33].

Feature selection for classification

We consider the binary classification problem as a nonsubmodular example to which the SupSub procedure can be applied. Note that exploiting more features does not necessarily reduce the classification error; however, we should find a way to select the most informative features for a given data set as quickly as possible. Thus, a greedy feature-selection procedure can be considered an attractive solution.

Consider the binary classification problem, modeled as a binary hypothesis test given by

$$\begin{cases}\mathcal{H}_0:\mathbf{y}_{\mathcal{S}}\sim\mathcal{N}(\boldsymbol{\theta}_{0,\mathcal{S}},\boldsymbol{\Sigma}_{0,\mathcal{S}})\\ \mathcal{H}_1:\mathbf{y}_{\mathcal{S}}\sim\mathcal{N}(\boldsymbol{\theta}_{1,\mathcal{S}},\boldsymbol{\Sigma}_{1,\mathcal{S}})\end{cases}, \tag{21}$$

where $\mathcal{S}\subseteq\mathcal{N}$ is the subset of selected features from the ground set $\mathcal{N}=\{1,\ldots,N\}$. The mean vectors of the selected data under $\mathcal{H}_0$ and $\mathcal{H}_1$ are denoted by $\boldsymbol{\theta}_{0,\mathcal{S}}$ and $\boldsymbol{\theta}_{1,\mathcal{S}}$ and the second-order statistics by $\boldsymbol{\Sigma}_{0,\mathcal{S}}$ and $\boldsymbol{\Sigma}_{1,\mathcal{S}}$, respectively.

We consider the Kullback–Leibler (KL) divergence, $\mathcal{K}(\mathcal{H}_1\,\|\,\mathcal{H}_0)$, as the performance measurement for the classification task, which is a distance measuring how far the two hypotheses $\mathcal{H}_0$ and $\mathcal{H}_1$ are. Unfortunately, the KL divergence is not submodular. To solve this problem, the KL divergence was decomposed in [6] as the difference of two submodular set functions, which means the SupSub procedure can be employed.

Now, let us assess the performance of the greedy method for feature selection. In this example, we consider two classes described by a Gaussian distribution with second-order statistics $\mathbf{\Sigma}_0$ and $\mathbf{\Sigma}_1$ and mean vectors $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$, respectively. Here, the covariance matrices are assumed to be Toeplitz matrices (a common structure in SP problems). The total number of features is 50, and the trained classifier is the quadratic discriminant classifier (QDC). In Figure 10, the classification soft error for different feature-selection methods is depicted versus the cardinality of the selected feature set. The feature-selection method in the MATLAB pattern recognition toolbox (PRTools) Toolbox is considered here as a baseline [6]. As shown in this figure, the method based on the SupSub procedure provides a desirable performance superior to the PRTools baseline result. Furthermore, the KL greedy, i.e., directly applying the greedy heuristic to the KL function, outperforms the other methods in most cases. However, it can get stuck sometimes and has no near-optimality guarantees (for further discussions, see [6]).

Submodularity and continuous domain optimization

As discussed in previous sections, submodularity is a useful property of functions defined in a discrete domain, which admits a guaranteed approximate optimization with efficient algorithms. The reader may expect that an extension of submodularity to the continuous domain provides similar benefits for continuous optimization problems.

In this regard, in [34], the notion of continuous submodularity is defined on subsets of $\mathbb{R}^N$ so that function $f:\mathcal{X}\to\mathbb{R}$, where $\mathcal{X}=\prod_{i=1}^{N}\mathcal{X}_i$ with each $\mathcal{X}_i$ is an interval, is continuous submodular if and only if for all $(\mathbf{x},\mathbf{y})\in\mathcal{X}\times\mathcal{X}$,

$$f(\mathbf{x})+f(\mathbf{y})\geq f(\mathbf{x}\vee\mathbf{y})+f(\mathbf{x}\wedge\mathbf{y}), \tag{22}$$

where $\vee$ and $\wedge$ stand for the coordinate-wise maximum and minimum operators, respectively. When $f(\cdot)$ is twice-differentiable, this function is submodular iff all nondiagonal elements of its Hessian are nonpositive [34].

The class of continuous submodular functions covers a subset of both convex and concave functions. As an example, a function of the form $f_{i,j}(x_i-x_j)$ for a convex $f_{i,j}$ is both submodular and convex; or, an indefinite quadratic function of the form $f(\mathbf{x})=(1/2)\mathbf{x}^T\mathbf{A}\mathbf{x}+\mathbf{b}^T\mathbf{x}+c$ with all nondiagonal elements of $\mathbf{A}$ nonpositive is a submodular, but nonconvex/nonconcave, function.

Analogous to the discrete domain, the DRs' property is generalized to functions defined over $\mathcal{X}$ (see [35]). It is clear that for set functions, the DR property is equivalent to submodularity, however, for general continuous domain functions, submodularity does not necessarily imply the DR property. In other words, the DR property is stronger than submodularity in general. If a continuous submodular function is coordinate-wise concave, it satisfies the DR property [35], which defines a subclass of submodular functions called *DR-submodular*. Being twice-differentiable, DR-submodularity is equivalent to the nonpositivity of all Hessian entries.

One can exploit the well-known gradient ascent algorithm to maximize a continuous submodular function, which achieves a one-half-approximation guarantee [36]. To gain a superior guarantee, in [35], a variant of the Frank–Wolfe algorithm for maximizing a monotone DR-submodular continuous function under down-closed convex constraints has been proposed, providing a $(1-1/e)$-approximation guarantee. Recently, in [37], a stochastic, continuous greedy algorithm has been developed, achieving a $(1-1/e)$-approximation guarantee, which deals with maximizing a similar optimization problem subject to a general convex body constraint. Maximizing nonmonotone continuous DR-submodular functions has also been studied in [35], [38], and [39]. Furthermore, the problem of submodular continuous function minimization has been considered in [34], which proved that efficient techniques from convex optimization can be employed for this task.

Nonconvex/nonconcave quadratic function maximization

Nonconvex/nonconcave quadratic programming under general convex constraints occurs in various applications, including price optimization, scheduling, graph theory, and free boundary problems. A special class of such problems is submodular quadratic programming, which can be tractably optimized. In this example, a monotone DR-submodular quadratic program is generated under the positive polytope constraint $\mathcal{P}=\{\mathbf{x}\in\mathbb{R}^N\,|\,\mathbf{A}\mathbf{x}\leq\mathbf{b},\mathbf{0}\leq\mathbf{x}\leq 1\}$, where $\mathbf{A}$ has uniformly distributed entries in the interval $[0,1]$, $\mathbf{b}=b\mathbf{1}$ and $N=100$. In Figure 11, the value of the objective function obtained by the Frank–Wolfe variant [35] is compared as a function of b with that of the random and empirically tuned projected gradient method [36] used for three different step sizes. It is noteworthy that the Frank–Wolfe variant provides provable performance guarantees without any tuning requirement, while the performance of the projected gradient is sensitive to parameter tuning.

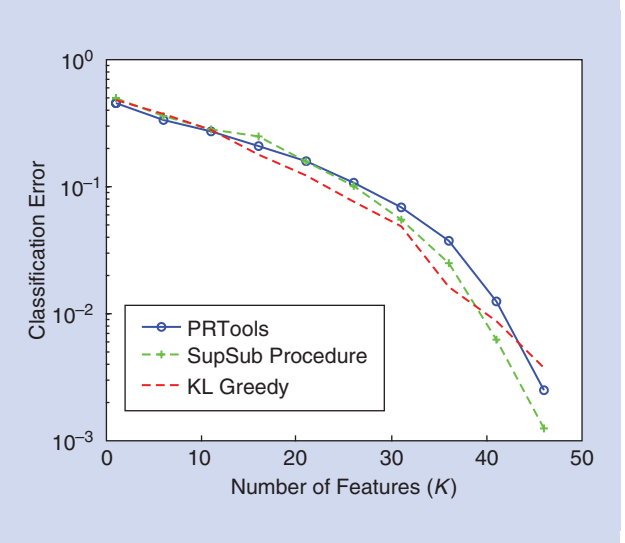


FIGURE 10. A classification soft error when using a quadratic discriminant classifier for a Gaussian binary classification problem [6].

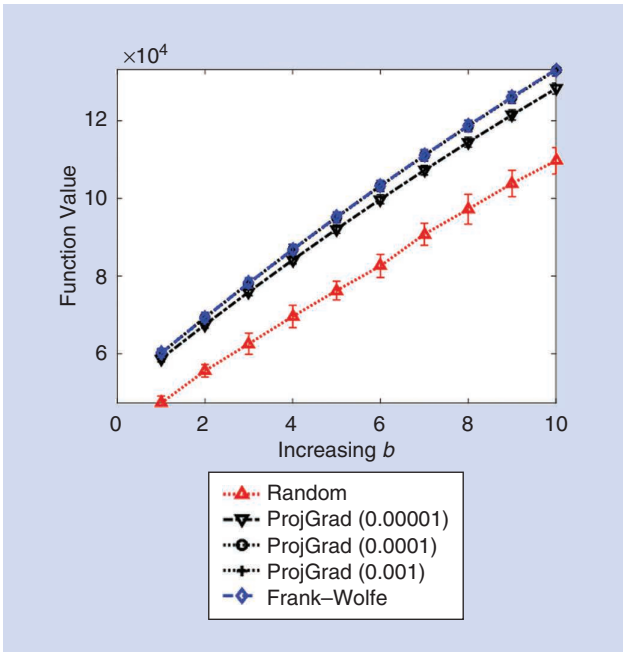


FIGURE 11. The objective function value as a function of different values of the upper-bound b [35].

Conclusions and research trends

In this article, we explained the concept of submodularity, provided the intuition for how it works, and illustrated some properties. The connection with the convexity in the continuous domain was discussed and the minimization problem was briefly explained. Also, the concavity aspect of submodularity was demonstrated along with low-computational complexity algorithms to maximize submodular functions where the corresponding theorems that guarantee a near-OPT were presented. Moreover, several applications in SP and ML have been covered to transfer the flavor of submodularity to practice. However, it should be pointed out that there is vast literature on submodularity with a wide variety of applications that, for the sake of conciseness, were not covered in this article. Continuous submodularity is one of the ongoing research directions that finds applications in robust resource allocation [40]. Online submodular optimization is another research trend that opens up opportunities for many applications, such as experimental design [41]. Finally, it is worth mentioning that submodular optimization is an active research area that is growing rapidly, not only through proposing new algorithms and theories but also by introducing new applications.

Authors

Ehsan Tohidi (tohidi@eurecom.fr) received his B.Sc., M.Sc., and Ph.D. degrees in electrical engineering (with a specialization on communications and signal processing) from Sharif University of Technology, Tehran, Iran, in 2011, 2013, and 2018, respectively. From September 2016 to March 2017, he was a visiting Ph.D. student with the Department of Electrical Engineering, Mathematics, and Computer Science, Delft University of Technology, The Netherlands. He is currently a postdoctoral researcher at EURECOM, Biot, France. His research interests include signal processing, discrete and continuous optimization, resource allocation, and machine learning.

Rouhollah Amiri (amiri_rouhollah@ee.sharif.edu) received his B.Sc. degree from the University of Tehran, Iran, and his M.Sc. and Ph.D. degrees from Sharif University of Technology, Tehran, Iran, in 2012, 2014, and 2018, respectively, all in electrical engineering. He is currently a postdoctoral researcher at Sharif University of Technology. His research interests include statistical signal processing, submodular and convex optimization, localization, and machine learning.

Mario Coutino (m.a.coutinominguez-1@tudelft.nl) received his M.Sc. degree (cum laude) in electrical engineering from Delft University of Technology, The Netherlands, in 2016, where he has been working toward his Ph.D. degree in signal processing. He has held positions at Thales Nederlands in 2015 and Bang & Olufsen during 2015–2016. He received a Best Student Paper Award for his publication at the IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing in 2017 and was a visiting researcher with RIKEN Center for Advanced Intelligence Project and the University of Minnesota in 2018 and 2019, respectively. His research interests include signal processing on networks, optimization, and numerical linear algebra. He is a Student Member of the IEEE.

David Gesbert (gesbert@eurecom.fr) received his Ph.D. degree from TelecomParis, France, in 1997. He is the head of the Communication Systems department at EURECOM. From 1997 to 1999, he was with the Information Systems Laboratory, Stanford University. He was a founding engineer of Iospan Wireless, Inc. (now Intel). He has published approximately 300 papers and been awarded 25 patents. He was a technical program cochair for the IEEE International Conference on Communications in 2017. He was named a Thomson–Reuters Highly Cited Researcher in computer science. Since 2015, he has held the European Research Council's Advanced grant PERFUME. He is a board member for the OpenAirInterface Software Alliance. Since early 2019, he has been heading the Huawei-funded Chair on Advanced Wireless Systems towards 6G networks. In 2020, he was awarded funding by the French Interdisciplinary Institute on Artificial Intelligence for a chair in the area of artificial intelligence for the future Internet of Things.

Geert Leus (g.j.t.leus@tudelft.nl) received his M.Sc. and Ph.D. degrees in electrical engineering from KU Leuven, Belgium, in June 1996 and May 2000, respectively. He is currently a full professor with the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, The Netherlands. He was a Member-at-Large of the Board of Governors of the IEEE Signal Processing Society, the chair of the IEEE Signal Processing for Communications and Networking Technical Committee, and the editor-in-chief of *EURASIP Journal on Advances in Signal Processing*. Currently, he is the chair of the EURASIP Technical Area Committee on Signal Processing for

Multisensor Systems and the editor-in-chief of *EURASIP Signal Processing*. He is a Fellow of IEEE and of EURASIP.

Amin Karbasi (amin.karbasi@yale.edu) received his Ph.D. degree from École Polytechnique Fédérale de Lausanne and was a postdoctoral researcher at ETH Zurich. He is currently an associate professor of electrical engineering, computer science, and statistics and data science at Yale University. He received the National Science Foundation Career Award (2019), the U.S. Office of Naval Research Young Investigator Award (2019), the U.S. Air Force Office of Scientific Research Young Investigator Award (2018), the DARPA Young Faculty Award (2016), the National Academy of Engineering Grainger Award (2017), the Amazon Research Award (2018), the Google Faculty Research Award (2016), the Simons Research Fellowship (2017), and the ETH Research Fellowship (2013).

References

[1] D. J. A. Welsh, "Matroid theory," *London Math. Soc. Monographs*, vol. 8. London: Academic, 1976.

[2] A. Krause and D. Golovin, "Submodular function maximization," in *Tractability: Practical Approaches to Hard Problems*. Cambridge, U.K.: Cambridge Univ. Press, Feb. 2014.

[3] F. R. Bach, "Learning with submodular functions: A convex optimization perspective," *Found. Trends Mach. Learning*, vol. 6, nos. 2–3, pp. 145–373, 2013. doi: 10.1561/2200000039.

[4] S. Fujishige, *Submodular Functions and Optimization*, vol. 58. Amsterdam, The Netherlands: Elsevier, 2005.

[5] E. Tohidi, M. Coutino, S. P. Chepuri, H. Behroozi, M. M. Nayebi, and G. Leus, "Sparse antenna and pulse placement for colocated MIMO radar," *IEEE Trans. Signal Process.*, vol. 67, no. 3, pp. 579–593, Feb. 2019. doi: 10.1109/TSP.2018.2881656.

[6] M. Coutino, S. P. Chepuri, and G. Leus, "Submodular sparse sensing for Gaussian detection with correlated observations," *IEEE Trans. Signal Process.*, vol. 66, no. 15, pp. 4025–4039, Aug. 2018. doi: 10.1109/TSP.2018.2846220.

[7] K. Thekumparampil, A. Thangaraj, and R. Vaze, "Combinatorial resource allocation using submodularity of waterfilling," *IEEE Trans. Wireless Commun.*, vol. 15, no. 1, pp. 206–216, Jan. 2016. doi: 10.1109/TWC.2015.2469291.

[8] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *J. Artificial Intell. Res.*, vol. 42, pp. 427–486, 2011.

[9] E. Elenberg, A. G. Dimakis, M. Feldman, and A. Karbasi, "Streaming weak submodularity: Interpreting neural networks on the fly," in *Proc. Advances Neural Information Processing Systems*, 2017, pp. 4044–4054.

[10] V. Tzoumas, K. Gatsis, A. Jadbabaie, and G. J. Pappas, "Resilient monotone submodular function maximization," in *Proc. IEEE 56th Annu. Conf. Decision and Control (CDC)*, Dec. 2017, pp. 1362–1367. doi: 10.1109/CDC.2017.8263844.

[11] U. Feige, V. S. Mirrokni, and J. Vondrák, "Maximizing non-monotone submodular functions," *SIAM J. Comput.*, vol. 40, no. 4, pp. 1133–1153, 2011. doi: 10.1137/090779346.

[12] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions–I," *Math. Programming*, vol. 14, no. 1, pp. 265–294, 1978. doi: 10.1007/BF01588971.

[13] J. Vondrák, "Optimal approximation for the submodular welfare problem in the value oracle model," in *Proc. 40th Annu. ACM Symp. Theory Computing*, 2008, pp. 67–74.

[14] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM J. Comput.*, vol. 40, no. 6, pp. 1740–1766, 2011. doi: 10.1137/080733991.

[15] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proc. 13th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2007, pp. 420–429. doi: 10.1145/1281192.1281239.

[16] H. Lin and J. Bilmes, "Multi-document summarization via budgeted maximization of submodular functions," in *Human Language Technologies: The 2010 Annu. Conf. North American Chapter Association Computational Linguistics*, 2010, pp. 912–920.

[17] M. Sviridenko, "A note on maximizing a submodular set function subject to a knapsack constraint," *Oper. Res. Lett.*, vol. 32, no. 1, pp. 41–43, 2004. doi: 10.1016/S0167-6377(03)00062-2.

[18] A. Das and D. Kempe, "Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection," *J. Mach. Learn. Res.*, vol. 19, no. 1, pp. 74–107, 2018.

[19] A. Krause and V. Cevher, "Submodular dictionary selection for sparse representation," in *Proc. Int. Conf. Machine Learning (ICML)*, 2010, pp. 567–574.

[20] E. R. Elenberg, R. Khanna, A. G. Dimakis, and S. Negahban, "Restricted strong convexity implies weak submodularity," *Ann. Statist.*, vol. 46, no. 6B, pp. 3539–3568, 2018. doi: 10.1214/17-AOS1679.

[21] C. Harshaw, M. Feldman, J. Ward, and A. Karbasi, Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. 2019. [Online]. Available: arXiv:1904.09354

[22] S. Joshi and S. Boyd, "Sensor selection via convex optimization," *IEEE Trans. Signal Process.*, vol. 57, no. 2, pp. 451–462, 2008. doi: 10.1109/TSP.2008.2007095.

[23] M. Mitrovic, E. Kazemi, M. Feldman, A. Krause, and A. Karbasi, Adaptive sequence submodularity. 2019. [Online]. Available: arXiv:1902.05981

[24] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause, "Distributed submodular maximization," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 8330–8373, 2016.

[25] R. Kumar, B. Moseley, S. Vassilvitskii, and A. Vattani, "Fast greedy algorithms in MapReduce and streaming," *ACM Trans. Parallel Computing (TOPC)*, vol. 2, no. 3, pp. 1–22, 2015. doi: 10.1145/2809814.

[26] B. Mirzasoleiman, A. Karbasi, A. Badanidiyuru, and A. Krause, "Distributed submodular cover: Succinctly summarizing massive data," in *Proc. Advances Neural Information Processing Systems*, 2015, pp. 2881–2889.

[27] L. Lovász, "Submodular functions and convexity," in *Mathematical Programming: The State of the Art*. A. Bachem, M. Grotschel, and B. Korte, Eds. Berlin: Springer-Verlag, 1983, pp. 235–257.

[28] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Res. Logistics Quart.*, vol. 3, nos. 1–2, pp. 95–110, 1956. doi: 10.1002/nav.3800030109.

[29] M. Jaggi, "Revisiting Frank–Wolfe: Projection-free sparse convex optimization," in *Proc. 30th Int. Conf. Machine Learning*, 2013, pp. 427–435.

[30] Z. Svitkina and L. Fleischer, "Submodular approximation: Sampling-based algorithms and lower bounds," *SIAM J. Comput.*, vol. 40, no. 6, pp. 1715–1737, 2011. doi: 10.1137/100783352.

[31] M. Narasimhan and J. A. Bilmes, A submodular-supermodular procedure with applications to discriminative structure learning. 2012. [Online]. Available: arXiv:1207.1404

[32] A. L. Yuille and A. Rangarajan, "The concave–convex procedure," *Neural Comput.*, vol. 15, no. 4, pp. 915–936, 2003. doi: 10.1162/08997660360581958.

[33] R. Iyer and J. Bilmes, Algorithms for approximate minimization of the difference between submodular functions, with applications, 2012. [Online]. Available: arXiv:1207.0560

[34] F. Bach, "Submodular functions: From discrete to continuous domains," *Math. Program.*, vol. 175, nos. 1–2, pp. 419–459, 2019. doi: 10.1007/s10107-018-1248-6.

[35] A. Bian, B. Mirzasoleiman, J. M. Buhmann, and A. Krause, "Guaranteed non-convex optimization: Submodular maximization over continuous domains," in *Proc. Int. Conf. Artificial Intelligence and Statistics*, 2017, pp. 111–120.

[36] H. Hassani, M. Soltanolkotabi, and A. Karbasi, "Gradient methods for submodular maximization," in *Proc. Advances Neural Information Processing Systems*, 2017, pp. 5841–5851.

[37] A. Mokhtari, H. Hassani, and A. Karbasi, Conditional gradient method for stochastic submodular maximization: Closing the gap. 2017. [Online]. Available: arXiv:1711.01660

[38] A. Bian, K. Levy, A. Krause, and J. M. Buhmann, "Continuous DR-submodular maximization: Structure and algorithms," in *Proc. Advances Neural Information Processing Systems*, 2017, pp. 486–496.

[39] A. Mokhtari, H. Hassani, and A. Karbasi, Stochastic conditional gradient methods: From convex minimization to submodular maximization. 2018. [Online]. Available: arXiv:1804.09554

[40] M. Staib and S. Jegelka, "Robust budget allocation via continuous submodular functions," in *Proc. 34th Int. Conf. Machine Learning*, 2017, vol. 70, pp. 3230–3240.

[41] L. Chen, H. Hassani, and A. Karbasi, "Online continuous submodular maximization," in *Int. Conf. Artificial Intelligence and Statistics*, 2018, pp. 1896–1905.

SP