Submodular Maximization and its Applications,
with Recent Advances in Streaming/Distributed Computing

Jiecao Chen

February 27, 2016

1 Introduction

Submodularity is a property of set functions with deep theoretical and practical consequences. Submodular functions occur in a variety of applications, including representative skyline selection [23], network structure learning [10], influence maximization [12], document summarization [15], image segmentation [4, 13] and many others.

There is a large body of research in submodular optimization which we can not cover thoroughly here. In this survey, we focus on the recent advances in optimizing (mostly, maximizing) submodular functions in distributed and streaming setting.

Through Lovász' extension, a submodular minimization problem can be solved via techniques from convex optimization. Therefore most work focuses on submodular maximization. In particular, almost all results for distributed and streaming submodular optimization are for maximization problems. In this survey, we also focus on submodular maximization, but will briefly mention minimization problems as well when necessary.

1.1 Notations

Through out this survey,

we use V to represent the ground set we consider.

2^V is the power set of V (i.e. the set of all subsets of V).

In general, A^V is the collection of maps from V to A.

For simplicity, we sometime write $A \cup \{x\}$ as $A + x$.

2 Submodularity

In this section, we first give several equivalent definitions of submodularity, and then we introduce several fundamental properties of submodular functions. We also discuss various constraints that occur frequently in submodular optimization problems. In the last part of this section, we cover algorithms that solve constrained submodular maximization problems with theoretical approximation guarantee.

2.1 Definitions

There are many equivalent definitions, and we will discuss three of them in this section.

Definition 1 (submodular concave) A function $f : 2^V \to \mathbb{R}$ is submodular if for any $A, B \subseteq V$, we have that:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B). \quad (1)$$

An alternate equivalent definition is more interpretable in many situations,

Definition 2 (diminishing returns) A function $f : 2^V \to \mathbb{R}$ is submodular if for any $A \subseteq B \subset V$, and $v \in V \backslash B$, we have that:

$$f(A + v) - f(A) \geq f(B + v) - f(B). \quad (2)$$

Intuitively, this definition requires that the incremental "gain" of adding a new element v decreases (diminishes) as the base set grows from A to B. We will see that this property is actually shared by many real-world phenomenons.

It turns out that a stronger but equivalent statement can also serve as the definition of a submodular function,

Definition 3 (group diminishing returns) A function $f : 2^V \to \mathbb{R}$ is submodular if for any $A \subseteq B \subset V$, and $C \subseteq V \backslash B$, we have that:

$$f(A \cup C) - f(A) \geq f(B \cup C) - f(B). \quad (3)$$

2.2 Modularity and Supermodularity

We also briefly mention modularity and supermodularity here. These two concepts are closely related to submodularity.

A function $f : 2^V \to \mathbb{R}$ is modular if we replace inequality by equality in Definition 2 (or any of other two). Formally,

Definition 4 (Modularity) A function $f : 2^V \to \mathbb{R}$ is modular if for any $A \subseteq B \subset V$, and $v \in V \backslash B$, we have that:

$$f(A + v) - f(A) = f(B + v) - f(B). \quad (4)$$

Notably, a modular function f can always be written as

$$f(S) = f(\emptyset) + \sum_{v \in S} (f(\{v\}) - f(\emptyset))$$

for any $S \subseteq V$.

If we further assume $f(\emptyset) = 0$ (in this case, we call f normalized), we have a simplified expression,

$$f(S) = \sum_{v \in S} f(\{v\}).$$

Modularity can be useful in our discussion of submodularity, because one can use modular functions to construct submodular functions with desired properties in their applications. Examples can be found in e.g. [15, 16].

A supermodular function is defined by flipping the inequality sign in the definition of a submodular function. Formally,

Definition 5 (Supermodularity) A function $f : 2^V \to \mathbb{R}$ is modular if for any $A \subseteq B \subset V$, and $v \in V \backslash B$, we have that:

$$f(A + v) - f(A) \leq f(B + v) - f(B). \tag{5}$$

We will focus on submodular functions because a function is supermodular if and only if its negative is submodular.

2.3 Properties

Like convex and concave functions, submodular functions have many nice properties. Lovász's description of convex functions [17] can be viewed as accurate comments on submodularity:

- Convex functions occur in many mathematical models in economy, engineering, and other sciences. Convexity is a very natural property of various functions and domains occurring in such models; quite often the only non-trivial property which can be stated in general.

- Convexity is preserved under many natural operations and transformations, and thereby the effective range of results can be extended, elegant proof techniques can be developed as well as unforeseen applications of certain results can be given.

- Convex functions and domains exhibit sufficient structure so that a mathematically beautiful and practically useful theory can be developed.

- There are theoretically and practically (reasonably) efficient methods to find the minimum of a convex function.

We survey several useful properties which can be useful in our later section. More properties of submodularity can be found in e.g. [3, 9].

Submodularity is close under addition,

Property 1 *Let $f_1, f_2 : 2^V \to \mathbb{R}$ be two submodular functions. Then*

$$f : 2^V \to \mathbb{R} \; with \; f(A) = \alpha f_1(A) + \beta f_2(A)$$

is submodular for any fixed $\alpha, \beta \in \mathbb{R}^+$.

Adding a modular function does not break submodularity,

Property 2 *Let $f_1, f_2 : 2^V \to \mathbb{R}$, f_1 is submodular and f_2 is modular. Then*

$$f : 2^V \to \mathbb{R} \; with \; f(A) = f_1(A) + \alpha f_2(A)$$

is submodular for any fixed $\alpha \in \mathbb{R}$.

Submodularity is preserved under restriction,

Property 3 *Let $f : 2^V \to \mathbb{R}$ be a submodular function. Let $S \subseteq V$ be a fixed set. Then*

$$f' : 2^V \to \mathbb{R} \; with \; f'(A) = f(A \cap S)$$

is submodular.

As a direct implication of Property 1 and Property 3, we have the following more general result,

Property 4 *Let $f : 2^V \to \mathbb{R}$ be a submodular function, $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ be a collection of subsets of V (i.e. each $C_i \subseteq V$). Then*

$$f' : 2^V \to \mathbb{R} \; with \; f'(A) = \sum_{C \in \mathcal{C}} f(A \cap C)$$

is submodular.

This property can be useful in graphical models and image processing. CHEN : *TODO: show examples*

Following property can be useful when we show that the objective function of k-medoid problem is supermodular,

Property 5 *Consider V as a set of indices. Let $\mathbf{c} \in \mathbb{R}^V$ be a fixed vector, c_i be its ith coordinate. Then*

$$f : 2^V \to \mathbb{R} \; with \; f(A) = \max_{j \in A} c_i$$

is submodular.

We can use non-negative modular function and a concave function to construct submodular functions,

Property 6 *Let $m : 2^V \to \mathbb{R}^+$ be a modular function, and f a concave function over $\mathbb{R}$. Then*

$$f : 2^V \to \mathbb{R} \; with \; f(A) = g(m(A))$$

is submodular.

Before introducing the next property, we define the monotonitcity of set function,

Definition 6 (Monotonitcity) An set function $f : 2^V \to \mathbb{R}$ is said to be monotone non-decreasing if for any $A \subseteq B \subseteq V$, $f(A) \leq f(B)$. Monotone non-increasing function can be defined similarly.

Property 7 *Let $f, g : 2^V \to \mathbb{R}$ be two submodular functions. If $(f - g)(\cdot)$ is either monotone non-decreasing or monotone non-increasing, then $f : 2^V \to \mathbb{R}$ with*

$$f(A) = \min(f(A), g(A))$$

is submodular.

2.4 Constraints

Now we discuss the constraints in submodular optimization problems. A submodular maximization problem usually has the following form,

$$\arg\max_{I \in \mathcal{I}} f(I) \tag{6}$$

where f is a submodular function and $\mathcal{I} \subseteq 2^V$ is the collection of all feasible solutions. We call $\mathcal{I}$ the constraint of the optimization problem. The structure of $\mathcal{I}$ plays a crucial role in submodular optimization. Different constraints have different hardness results, normally the difficulty increases when the constraint becomes more general. We will introduce cardinality constraint, knapsack constraint, matroid, and p-matchoid.

2.4.1 Cardinality Constraint

Cardinality constraint is perhaps the most straightforward constraint we would discuss in this survey. Efficient algorithms have been developed for finding or approximating the optimal solution of (6). There are also a lot of discussions on optimization subject to cardinality constraint, in both streaming and distributed setting.

A cardinality constraint is parameterized with a fixed constant $k \in \mathbb{Z}^+$. It is simply defined as $\mathcal{I} = \{A \subseteq V \mid |V| \leq k\}$, i.e. all subsets of V with size no larger than k. Cardinality constraint is arguably the most popular constraint, and it occurs everywhere. For example, in k-medoid clustering, we want to find a set S of *at most* k points, that minimizes the total distance of all points to S.

2.4.2 Knapsack Constraint

Knapsack constraint generalizes cardinality constraint by assigning each element in V a weight. Given a budget $W > 0$ and assume that each $i \in V$ is assigned a weight $w_i \geq 0$, a knapsack constraint can be defined as $\mathcal{I} = \{S \subseteq V \mid \sum_{i \in S} w_i \leq W\}$.

2.4.3 Matroid

Informally, a Matroid is the abstraction of the *independence* concept in linear algebra. In fact there are so many results around Matroid and the Matroid itself becomes a subfield of algebra. We cover some basics of Matroid theory and from which readers can easily see how powerful this concept is.

Before discussing the concept of a matroid, we briefly review the independence concept from linear algebra. For simplicity, let us just consider $\mathbb{R}^d$ instead of a general linear space. A subset S of $\mathbb{R}^d$ is said to be *independent* if there does not exist any $\mathbf{x} \in S$ such that $\mathbf{x}$ can be represented by linear combination of vectors in $S \backslash \{\mathbf{x}\}$.

Let $\mathcal{I} = \{S \subseteq \mathbb{R}^d \mid S \text{ is independent }\}$, i.e. only a independent set can be considered feasible. From what we learn in college linear algebra course, we know $\mathcal{I}$ has the following propertie: 1) $\emptyset \in \mathcal{I}$; 2) if $I \in \mathcal{I}$, any of I's subsets is also in $\mathcal{I}$; 3) if $J, I \in \mathcal{I}$ and J has smaller size than I, we must be able to find an element $\mathbf{x} \in I \backslash J$ such that $J \cup \{\mathbf{x}\} \in \mathcal{I}$.

Even for the "trivial" size function $f : 2^V \to \mathbb{Z}$ with $f(A) = |A|$, optimizing $\arg\max_{I \in \mathcal{I}} f(S)$ would have tremendous applications because its optimal solution is a base of the vector space.

We will see shortly how this optimization problem has direct connection with *Maximum Spanning Tree* problem. If we somehow generalize the definition of independence, we may be able to model a much more larger class of problems into the form (6).

It turns out that the properties of $\mathcal{I}$ we just described are sufficient to give a meaningful definition for Matroid. Formally,

Definition 7 (Matroid) A set system $(V, \mathcal{I})$ is a Matroid if it has the following properties,

1. $\emptyset \in \mathcal{I}$
2. $\forall I \in \mathcal{I}, J \subseteq I \implies J \in \mathcal{I}$
3. $\forall I, J \in \mathcal{I}$, with $|I| = |J| + 1$, then $\exists\, x \in I \backslash J$ such that $J \cup \{x\} \in \mathcal{I}$

Note that, unlike in the $\mathbb{R}^d$ case, we restrict on a finite set V.CHEN : *is it necessary?*

Finally, we generalize the concept of rank in linear algebra. Let $(V, \mathcal{I})$ be a matroid, we define the rank function $r : 2^V \to \mathbb{Z}$ as $f(S) = \max_{I \subseteq S, I \in \mathcal{I}} |I|$, i.e. the rank of $S \subseteq V$ is the maximum possible size of S's subsets that are also members of $\mathcal{I}$ (or in other words, independent). Our definition of rank is consistent with what we have in linear algebra (in that case $\mathcal{I}$ is the collection of all independent sets). A rank function is submodular (as you may expect).

2.4.4 Matchoids

Let $\mathcal{M}_1 = (V_1, \mathcal{I}_1), \ldots, \mathcal{M}_q = (V_q, \mathcal{I}_q)$ be q matroids where $V = V_1 \cup \ldots V_q$. Let $\mathcal{I} = \{S \subseteq V \mid S \cap V_i \in \mathcal{I}_i \text{ for all } i\}$. The finite set system $(V, \mathcal{I})$ is a p-matchoid if for every $e \in V$, e is a member of V_i for at most p indices $i \in [q]$. The concept of p-matchoid generalizes the intersection of matroids (taking $p = q$ and $V_i = V$ for all i).

2.5 Algorithms for Submodular Maximization

There are a lot of results for submodular maximization in the centralized setting where the data can fit into the RAM. Those results normally assume the oracle model: one is given a value oracle and a membership oracle. Given $S \subseteq V$, the membership oracle answers if $S \in \mathcal{I}$ and the value oracle returns $f(S)$. We cover several classical results which serve as the building blocks for distributed/streaming algorithms for submodular maximization.

We introduce the notation for marginal gain: $\Delta_f(e|S) = f(S+e) - f(S)$. The following algorithm shows a popular greedy strategy for submodular optimization.

Algorithm 1: GREEDY algorithm for submodular maximization

Input: V the ground set, f the submodular function, $\mathcal{I}$ the constraint
Output: a set $S \subseteq V$

```
S ← ∅
while ∃ e ∈ V\S s.t. S ∪ {e} ∈ I do
    e ← arg max_{e∈V\S, S∪{e}∈I} Δ_f(e|S)
    S ← S ∪ {e}
return S
```

2.5.1 Algorithms for Cardinality Constraint

A celebrated result of [21] shows that,

Theorem 1 ([21]) *For a non-negative monotone non-decreasing submodular function $f : 2^V \to \mathbb{R}$, let $\mathcal{I}$ be the cardinality constraint, Algorithm 1 returns a $(1 - 1/e)$-approximation to $\arg\max_{I \in \mathcal{I}} f(S)$.*

For several classes of submodular functions, this result is actually the best one can expect for any efficient algorithm. In fact the hardness in [7, 21] shows that any algorithm that is only allowed sub-exponential number of value queries can not achieve better than $(1 - 1/e)$-approximation (for a large class of submodular functions).

There are several papers improving the running time of Algorithm 1 under cardinality constraint of size k (under which the membership oracle is trivial). Minoux [19] proposed LAZY-GREEDY as a fast implementation for Algorithm 1. Instead of computing $\Delta_f(e|S)$ for each $e \in V \backslash S$ in Line 3, LAZY-GREEDY keeps an upper bound $\rho(e)$ (initially $+\infty$) on the marginal gain sorted in decreasing order (or kept in a heap). In each iteration, the LAZY-GREEDY algorithm evaluates the element on top of the heap and updates its upper bound $\rho(e) \leftarrow \Delta(e|S)$. If the updated $\rho(e) \geq \rho(e')$ for all other e', submodularity guarantees that e is the element with the largest marginal gain. The exact number of value queries consumed by LAZY-GREEDY is unknown because it heavily relies on both f and V, experimental study however shows that the LAZY-GREEDY algorithm is in order of magnitude faster than the naive implementation of Algorithm 1.

Wei et al. [26] improved the running time of LAZY-GREEDY by approximating the underlying submodular function with a set of (sub)modular functions. Badanidiyuru et al. [2] proposed a different approach that uses only $O(\frac{|V|}{\epsilon} \log \frac{1}{\epsilon})$ number of value queries and guarantees $(1 - 1/e - \epsilon)$-approximation. This result was improved by Mirzasoleiman et al. [20] recently where they proposed a randomized algorithm (STOCHASTIC-GREEDY) that reduce the number of queries to $O(|V| \log \frac{1}{\epsilon})$ and the approximation guarantee is $(1 - 1/e - \epsilon)$, in expectation. The key observation made in [20] is that, instead of considering all $e \in V \backslash S$, one can only consider $O(\frac{|V|}{k} \log \frac{1}{\epsilon})$ random samples from $V \backslash S$.

2.5.2 Algorithms for Matroid Constraint

Now we consider the matroid constraint. Let $f(S) = \sum_{i \in S} w_i$, where $S \subseteq V$ and each element $i \in V$ is assigned a non-negative weight w_i. A notable result shows the deep connection between Algorithm 1 and the concept of matroid,

Theorem 2 (see e.g. [3, 9]) *Let $(V, \mathcal{I})$ be a set system we consider, $\mathcal{I}$ is a matroid* if and only if *for* any *modular function f defined in above way, Algorithm 1 leads to a optimal solution for $\arg\max_{I \in \mathcal{I}} f(I)$.*

Note that Algorithm 1 actually includes many greedy algorithms as special cases (e.g. maximum weighted spanning tree algorithm). The statement of Theorem 2 is so strong that it provides a complete characterization of a large class of problems.

Algorithm 1 can also be used to handle the matroid constraint. In particular, we have the following guarantee,

Theorem 3 ([21]) *For a non-negative non-decreasing submodular function f, given a matroid $(V, \mathcal{I})$, Algorithm 1 returns a $1/2$-approximation to the optimal solution.*

Based on the idea of continuous greedy process [25] and pipage rounding [1], Calinescu et al. [5] improved the approximation ratio to $(1 - 1/e)$ (in expectation) for monotone submodular maximization under matroid constraint. Filmus et al. [8] presented a randomized combinatorial $(1-1/e-\epsilon)$-approximation algorithm using only $O(|V|r^3\epsilon^{-3}\log r)$ number of value queries, where r is the size of returned set. Their method is based on local-search and is conceptually much simpler than [5]. Badanidiyuru et al. [2] gave an algorithm that runs in $O(\frac{r|V|}{\epsilon^4}\log^2\frac{r}{\epsilon})$ queries by using a variant of the continuous greedy algorithm.

Non-monotone submodular is normally more difficult to efficiently optimize. Some results from Feldman ..

3 Applications

In this section, we first show a list of possible applications and discuss several representative applications in detail. We will see from those examples that submodularity is such a natural property that many real-world problems can be cast in to the framework of submodular optimization (maximization).

3.1 List of Possible Applications

- Combinatorial Problems: set cover, max k coverage, vertex cover, edge cover, graph cut problems etc.
- Networks: social networks, viral marketing, diffusion networks etc.
- Graphical Models: image segmentation, tree distributions, factors etc.
- NLP: document summarization, web search, information retrieval
- Machine Learning: active/semi-supervised learning etc.
- Economics: markets, economies of scale

3.2 Classical Problems Revisited

We first show that several well-known problems actually fit into our standard submodular maximization framework.

Exemplar Based Clustering Clustering is one of the most important tasks in the area of data mining. In the k-medoid problem [11] one tries to minimize the sum of pairwise dissimilarities/distances between exemplars and the elements of the dataset. Let $d : V \times V \rightarrow \mathbb{R}^+ \cup \{0\}$ be a function that measures the pairwise dissimilarity, we define the k-medoid loss function as following,

$$L(S) = \sum_{e \in V} \min_{v \in S} d(e, v).$$

It is quite straightforward (by Property 1, 5) to show that $-L(S)$ is submodular. By introducing an auxiliary element e_0, we can transform L into a non-negative monotone submodular function,

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

A k-medoid problem can then be formulated as a submodular maximization problem subject to a cardinality constraint,

$$\arg\max_{S \subseteq V : |S| \leq k} f(S).$$

Set Cover Problem The *set cover problem* is an important problem in combinatorial optimization where we are given a collection of subsets of a set E, i.e. $V = \{C_1, C_2, \ldots, C_n\}$ where each $C_i \subseteq E$. We define a function $f : 2^V \to \mathbb{R}$ such that $f(S) = |\cup_{C \in S} C|$. We can interpret f as follow: given S as a subset of V, the value of $f(S)$ is the number of distinct elements covered by the sets in S.

One can easily verify that f satisfies the diminishing return property thus is a submodular function. Furthermore, it is clear that f is non-decreasing. Now given the cardinality constraint, we want to solve the following,

$$\arg\max_{S \subseteq V : |S| \leq k} f(S).$$

We may also assign each $C \in V$ a non-negative cost $w(C)$ (e.g. the size of C), and given a total budget W, our goal is to find a solution of the following,

$$\arg\max_{S \subseteq V : w(S) \leq W} f(S),$$

where $w(S) = \sum_{C \in S} w(C)$. This is a monotone submodular maximization problem under the knapsack constraint.

Maximum Spanning Forest Let us consider a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. In this case we consider E as the ground set and define

$$\mathcal{I} = \{S \subseteq E \mid \text{edge-induced graph } G(V, S) \text{ does not contain a circle}\}.$$

One can verify (via definition) that $(E, \mathcal{I})$ is a matriod. The rank function of $(E, \mathcal{I})$ can be interpreted as the size of the maximum spanning forest of an edge-induced graph, i.e. given $S \subseteq E$, $r(S)$ is the size of maximum spanning forest (in terms of number of edges) of $G(V, S)$.

Now assume that we assign each $e \in E$ a weight $w_e \geq 0$. Let $f : 2^E \to \mathbb{R}$ with $f(S) = \sum_{e \in S} w_e$ be the objective function we want to maximize. Clearly f is monotone and (sub)modular. we consider the following optimization problem,

$$\arg\max_{S \in \mathcal{I}} f(S).$$

This is exactly the *Maximum Spanning Forest* problem and by Theorem 2, we can solve it efficiently (and exactly) using Algorithm 1.

Maximum Cut in Graphs Given an undirected graph $G = (V, E)$ and a non-negative capacity function $c : E \to \mathbb{R}^+ \cup \{0\}$, the cut capacity function $f : 2^V \to \mathbb{R}$ defined by $f(S) = \sum_{e \in \delta(S)} c(e)$ is submodular, where $\delta(S) = \{e \in E \mid e \text{ has exactly one vertex in } S\}$ i.e. the set of edges crossing S and $E \backslash S$. To shows why f is submodular, we introduce an auxiliary function $f_e : 2^{\{u,v\}} \to \mathbb{R}$ for each $e = \{u, v\} \in E$ which is defined on the graph $G_e = (\{u, v\}, \{e\})$. We define,

- $f_e(\{u, v\}) = f_e(\emptyset) = 0$
- $f_e(\{u\}) = f_e(\{v\}) = w_e$

Then f_e is submodular. We have,

$$f(S) = \sum_{e \in E} f_e(S \cap \{u, v\}).$$

The submodularity of f follows Property 1 and Property 3.

An interesting optimization problem can then be formulated as following,

$$\arg\max_{S \subseteq E : |S| \leq k} f(S).$$

3.3 Applications to NLP

Summarization In the task of summarization, we are given a ground set V and we want to find a subset of V which maximizes some quality measurement under certain constraints. One popular formulation is that, given a function $f : 2^V \to \mathbb{R}$ that measures the quality of a summarization, we try to solve,

$$\arg\max_{S \in \mathcal{I}} f(S)$$

where $\mathcal{I}$ is a knapsack constraint.

Lin et al. [15] pointed out that a lot of existing work for document summarization task fit the knapsack optimization framework. Furthermore the quality measurement functions being used are usually submodular. Lin et al. also proposed a class of submodular functions that outperform previous work in many aspects. Each of those functions combines two terms, one that encourages the summary to be representative of the corpus, and the other positively rewards diversity. In the task of speech summarization, several submodular functions were discussed by Lin [14]. More discussion on the applications of submodular optimization to summarization can be found in [14].

Word Alignment Word alignment is a key component in most statistical machine translation systems. Unlike classical approaches that utilize graphical models, Lin et al. [16] viewed word alignment problem as submodular maximization problem under matroid constraints.

Suppose that we are given a source language (English) string $e_1, e_2, \ldots, e_n$ and a target language (French) string $f_1, f_2, \ldots, f_m$. Let $V = \{(i, j) \mid i \in [n], j \in [m]\}$ be the ground set. The goal is to find a match $S \subseteq V$ that maximizes a certain quality function under some constraints.

Let $P_1, \ldots, P_m$ be a partition of V where $P_j = [n] \times \{j\}$. The *fertility* restriction of a word f_j then requires that f_j can only match at most k_j words in $e_1, \ldots, e_n$, or equivalently the match $S \subseteq V$ satisfies $|S \cap P_j| \leq k_i$ for all $j \in [m]$. Such a constraint is called *partition constraint*, which is an important instance of matroid. Lin et al. then proposed a submodular function as the quality function, which is a composition of a concave function and a modular function. We refer readers to [16] for details.

3.4 Applications to Social Networks

Influence Maximization Domingos and Richardson [6,22] posed a fundamental algorithmic problem: if we can try to convince a subset of individuals (at most k) to adopt a new product or innovation, and the goal is to trigger a large cascade of further adoptions, which set of individuals should we target? Kempe et al. [12] considered such question as a discrete optimization problem. Let $G = (V, E, W)$ a social network (a directed graph with non-negative weights for each edge), two basic models of influence spreading were considered in [12],

- Linear Threshold Mode: after initializing $A \subset V$ as the set of active nodes, at each time step, a node v will turn from inactive to active if the total weights of its active neighbors is above some threshold θ_v. Here θ_v may or may not be a random variable.

- Independent Cascade Model: after initializing the active set $A \subset V$, at each time step each active node v will activate its neighbor u (for all inactive neighbors) with probability $w_{v,u}$ (which is the parameter of this network).

Now let $\sigma(A)$ be the expected number of active nodes after T time steps (T is usually set to be large). [12] proved that for both models, $\sigma : 2^V \to \mathbb{Z}$ is a monotone submodular function.

In our application, one is unlikely to able to efficiently compute the exact value of $\sigma(A)$. In fact, one can extend the result of Theorem 1 to show that, by using $(1 + O(\epsilon))$-approximate values for the function to be optimized, we obtain a $(1 - 1/e - \epsilon)$-approximation of the optimum.

Network Structure Inference Gomez Rodriguez et al. [10] first introduced submodular maximization to the context of network structure learning. They considered the problem of learning the network structure in a influence network: given a hidden directed network G^* (vertices are known while edges are not), we observe multiple cascades spreading over it. Each cascade c can be considered as a series of triples. Each triple has the form $(u_i, t_i, \phi_i)_c$ which presents the event that at time t_i, node u_i is reached by c with feature ϕ_i. The goal is to infer the structure of G^* based on a collection of cascades observed (denoted as C).

To model this process, they assumed that in each cascade, each node can only be influenced by at most one other node. So the influence structure of a cascade c is given by a directed tree $T \subseteq G$. Let $P(c|T)$ be the probability of c propagating in a particular tree pattern T, $P(c|G)$ the probability that cascade c occurs in a network G. Let $\mathcal{T}(G)$ be set of all spanning trees of G. [10] then proposed the model

$$P(c|G) = \max_{T \in \mathcal{T}(G)} P(c|T) = \max_{T \in \mathcal{T}(G)} \Pi_{(u,v) \in T} P_c(u, v)$$

where $P_c(u, v) \propto e^{-(t_v - t_u)}$. The structure of the network can be approximated by solving

$$\arg\max_{|G| \le k} P(C|G) = \arg\max_{|G| \le k} \Pi_{c \in C} P(c|G).$$

Consider the improvement of log-likelihood for cascade c under graph G over an empty graph K:

$$F_c(G) = \max_{T \in \mathcal{T}(G)} \log P(c|T) - \max_{T \in \mathcal{T}(K)} P(c|T).$$

It turns out that $F_c(G)$ can be expressed as following,

$$F_c(G) = \max_{T \in \mathcal{G}} \sum_{(u,v) \in T} w_c(u,v)$$

where $w_c(u,v)$ is a non-negative weight. Optimizing $P(C|G)$ is equivalently to optimize $F_C(G) = \sum_{c \in C} F_c(G)$ and the latter is shown to be submodular (the ground set is all possible graphs defined over the given vertices set).

3.5 Applications to Machine Learning

Kernel Machines Kernel machines [24] are powerful non-parametric learning techniques. Those approaches use kernel trick to reduce non-linear problems to linear tasks that have been well studied. The data set $V = \{x_1, x_2, \ldots, x_n\}$ is represented in a transformed space via a kernel matrix

$$K = \begin{pmatrix} \mathcal{K}(x_1, x_2) & \ldots & \mathcal{K}(x_1, x_n) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(x_n, x_1) & \ldots & \mathcal{K}(x_n, x_n) \end{pmatrix}$$

here $\mathcal{K} : V \times V \to \mathbb{R}$ is the kernel function that is symmetric and positive definite. For large-scale problem, even representing

3.6 More Applications

- [18] distributed k-centers using submodular optimization.

4 Streaming Submodular Maximization

5 Distributed Submodular Optimization

References

[1] A. A. Ageev and M. I. Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(3):307–328, 2004.

[2] A. Badanidiyuru and J. 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.

[3] J. Bilmes. Lecture notes on submodular optimization. `http://j.ee.washington.edu/~bilmes/classes/ee596b_spring_2014/`, 2014.

[4] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE, 2001.

[5] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[6] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66. ACM, 2001.

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

[8] Y. Filmus and J. Ward. Monotone submodular maximization over a matroid via non-oblivious local search. *SIAM Journal on Computing*, 43(2):514–542, 2014.

[9] S. Fujishige. *Submodular functions and optimization*, volume 58. Elsevier, 2005.

[10] M. Gomez Rodriguez, J. Leskovec, and A. 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.

[11] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[12] D. Kempe, J. Kleinberg, and É. 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.

[13] P. Kohli, M. P. Kumar, and P. H. Torr. P^3 & beyond: Move making algorithms for solving higher order functions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(9):1645–1656, 2009.

[14] H. Lin. *Submodularity in Natural Language Processing: Algorithms and Applications*. PhD thesis, 2012.

[15] H. Lin and J. 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.

[16] H. Lin and J. Bilmes. Word alignment via submodular maximization over matroids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 170–175. Association for Computational Linguistics, 2011.

[17] L. Lovász. Submodular functions and convexity. In *Mathematical Programming The State of the Art*, pages 235–257. Springer, 1983.

[18] G. Malkomes, M. J. Kusner, W. Chen, K. Q. Weinberger, and B. Moseley. Fast distributed k-center clustering with outliers on massive data. In *Advances in Neural Information Processing Systems*, pages 1063–1071, 2015.

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

[20] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrak, and A. Krause. Lazier than lazy greedy. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[21] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

[22] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70. ACM, 2002.

[23] A. D. Sarma, A. Lall, D. Nanongkai, R. J. Lipton, and J. Xu. Representative skylines using threshold-based preference distributions. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 387–398. IEEE, 2011.

[24] B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[25] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 67–74. ACM, 2008.

[26] K. Wei, R. Iyer, and J. Bilmes. Fast multi-stage submodular maximization. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1494–1502, 2014.