Submodular Maximization

advances in distributed/streaming computing

Jiecao Chen

Indiana University Bloomington

jiecchen@indiana

March 17, 2016

Overview

Definitions of Submodularity

Definition (submodular concave)

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

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B). \tag{1}$$

An alternate equivalent definition is more interpretable in many situations.

Definition (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:

$$f(A + v) - f(A) \geq f(B + v) - f(B). \tag{2}$$

Modular Functions

Definition (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:

$$f(A+v) - f(A) = f(B+v) - f(B). \tag{3}$$

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 or proper), we have a simplified expression,

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

Monotonitcity

Definition (Monotonitcity)

A set function $f : 2^V \to \mathbb{R}$ is said to be non-decreasing if for any $A \subseteq B \subseteq V$, $f(A) \leq f(B)$. Non-increasing set functions are defined in the similar way.

When we say a submodular function is monotone, we mean it is non-decreasing.

Properties

Submodularity is closed under addition.

Property

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

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

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

Submodularity is preserved under restriction.

Property

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

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

is submodular.

Properties cont.

The following property can be useful when we show that the negative of the objective function of k-median problem is submodular.

Property

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

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

is submodular.

Constraints

Submodular Maximization Problem

A submodular maximization problem usually has the following form:

$$\arg\max_{I \in \mathcal{I}} f(I), \tag{4}$$

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.

Constraints

$\mathcal{I}$ is important!

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.

Constraints

$\mathcal{I}$ is important!

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.

Popular constraints

Some popular constraints:

- Cardinality constraint
- Knapsack constraint
- Matroid constraint
- Matching
- p-System
- ...

Constraints cont.

First we define hereditary set systems.

Definition (Hereditary)

A constraint $\mathcal{I} \subseteq 2^V$ is said to be hereditary if

$$I \in \mathcal{I} \implies J \in \mathcal{I} \text{ for any } J \subseteq I.$$

A hereditary constraint is sometimes called an independent system and each $I \in \mathcal{I}$ is called an independent set.

All constraints we will discuss are hereditary.

Constraints cont.

Cardinality

Cardinality constraint: $\mathcal{I} = \{A \subseteq V \mid |A| \leq k\}$

Knapsack

Knapsack Constraint: each $i \in V$ is assigned a weight $w_i \geq 0$, $\mathcal{I} = \{S \subseteq V \mid \sum_{i \in S} w_i \leq W\}$.

Constraints cont.

Cardinality

Cardinality constraint: $\mathcal{I} = \{A \subseteq V \mid |A| \leq k\}$

Knapsack

Knapsack Constraint: each $i \in V$ is assigned a weight $w_i \geq 0$, $\mathcal{I} = \{S \subseteq V \mid \sum_{i \in S} w_i \leq W\}$.

Matching

Matching: given a graph $G = (V, E)$, a *Matching* is a set $S \subseteq E$ such that no edges in S share common vertex.

Matroid

Matroid is the generalization of the independence concept in linear algebra; omit details here ...

p-System

p-system is very general, it includes many other constraints as special cases.

Definition of p-System

Let $(V, \mathcal{I})$ be a set system and $\mathcal{I}$ hereditary. Let $\mathcal{B}(A)$ be the collection of all bases of A.

$$\mathcal{I} = \{A \subseteq V \mid \frac{\max_{S \in \mathcal{B}(A)} |S|}{\min_{S \in \mathcal{B}(A)} |S|} \leq p\}.$$

p-System

p-system is very general, it includes many other constraints as special cases.

Definition of p-System

Let $(V, \mathcal{I})$ be a set system and $\mathcal{I}$ hereditary. Let $\mathcal{B}(A)$ be the collection of all bases of A.

$$\mathcal{I} = \{A \subseteq V \mid \frac{\max_{S \in \mathcal{B}(A)} |S|}{\min_{S \in \mathcal{B}(A)} |S|} \leq p\}.$$

Note: a base of A is the maximal independent set included in A.

p-System

p-system is very general, it includes many other constraints as special cases.

Definition of p-System

Let $(V, \mathcal{I})$ be a set system and $\mathcal{I}$ hereditary. Let $\mathcal{B}(A)$ be the collection of all bases of A.

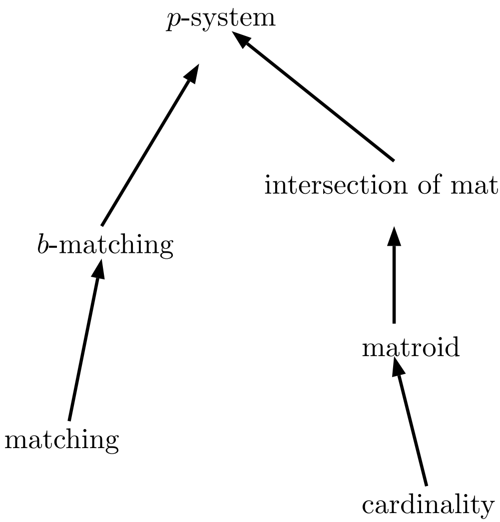
$$\mathcal{I} = \{A \subseteq V \mid \frac{\max_{S \in \mathcal{B}(A)} |S|}{\min_{S \in \mathcal{B}(A)} |S|} \leq p\}.$$

Note: a base of A is the maximal independent set included in A.

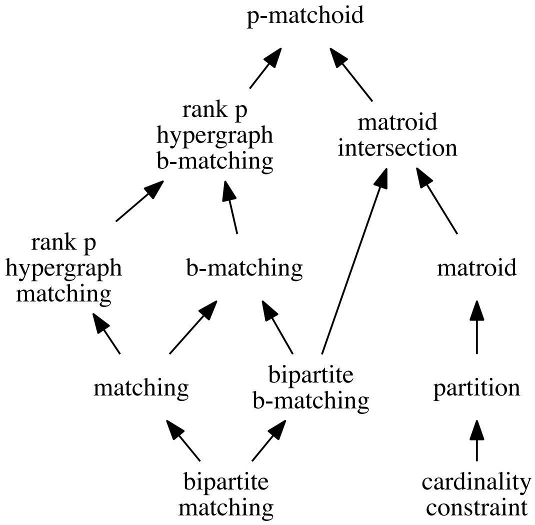
examples of p-system

- matroid is 1-system
- matching is 2-system
- intersection of p matroid is p-system
- ...

Hierarchy of constraints



Hierarchy of constraints (extended)



Notations

Some notations

- $\Delta_f(e|S) = f(S+e) - f(S)$ (or simply $\Delta(e|S)$ when f is clear from context)
- α-approximation: the returned solution S always satisfies $f(S) \geq \alpha \cdot \arg\max_{I \in \mathcal{I}} f(I)$
- When the algorithm is randomized, we normally say it guarantees α-approximation in expectation if

$$\mathbf{E}[f(S)] \geq \alpha \cdot \arg\max_{I \in \mathcal{I}} f(I).$$

The standard greedy algorithm

Algorithm 1: GREEDY algorithm for submodular maximization

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

Output: a set $S \subseteq V$

$S \leftarrow \emptyset$

while $\exists\ e \in V \backslash S$ *s.t.* $S \cup \{e\} \in \mathcal{I}$ **do**

  $e \leftarrow \arg\max_{e \in V \backslash S,\ S \cup \{e\} \in \mathcal{I}} \Delta_f(e|S)$

  $S \leftarrow S \cup \{e\}$

return S

Theorems of Algorithm 1

Theorem ([10], for cardinality constraint)

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

Theorems of Algorithm 1

Theorem ([10], for cardinality constraint)

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

Theorem ([10, 2], for p-system)

For a non-negative monotone submodular function f, given a p-system $(V, \mathcal{I})$, Algorithm 1 returns a $\frac{1}{p+1}$-approximation.

Theorems of Algorithm 1

Theorem ([10], for cardinality constraint)

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

Theorem ([10, 2], for p-system)

For a non-negative monotone submodular function f, given a p-system $(V, \mathcal{I})$, Algorithm 1 returns a $\frac{1}{p+1}$-approximation.

Theorem ([5], modular maximization s.t. p-system)

For a non-negative monotone modular function f, given a p-system $(V, \mathcal{I})$, Algorithm 1 returns a $\frac{1}{p}$-approximation.

Speedup - GreedyLazy

GreedyLazy

- Minoux [8] proposed Lazy-Greedy as a fast implementation for Algorithm 1.

Speedup - GreedyLazy

GreedyLazy

- Minoux [8] proposed Lazy-greedy as a fast implementation for Algorithm 1.
- GreedyLazy keeps an upper bound $\rho(e)$ on the marginal gain sorted in a heap.

Speedup - GreedyLazy

GreedyLazy

- Minoux [8] proposed Lazy-greedy as a fast implementation for Algorithm 1.
- GreedyLazy keeps an upper bound $\rho(e)$ on the marginal gain sorted in a heap.
- In each step, only update the top element in the heap and push it back, if this element remains in the top, include it into solution.

Speedup - GreedyLazy

GreedyLazy

- Minoux [8] proposed LAZY-GREEDY as a fast implementation for Algorithm 1.
- GREEDYLAZY keeps an upper bound $\rho(e)$ on the marginal gain sorted in a heap.
- In each step, only update the top element in the heap and push it back, if this element remains in the top, include it into solution.
- Again gives $(1 - e^{-1})$-approximation.

Speedup - StocGreedy[9]

StocGreedy

- In each round, instead of considering all $V \backslash S$ to get

$$e \leftarrow \underset{e \in V \backslash S,\ S \cup \{e\} \in \mathcal{I}}{\arg\max} \Delta_f(e|S),$$

Speedup - StocGreedy[9]

StocGreedy

- In each round, instead of considering all $V \backslash S$ to get

$$e \leftarrow \underset{e \in V \backslash S,\ S \cup \{e\} \in \mathcal{I}}{\arg\max} \Delta_f(e|S),$$

- consider only $\frac{|V|}{k} \log \frac{1}{\epsilon}$ random samples from $V \backslash S$.

Speedup - StocGreedy[9]

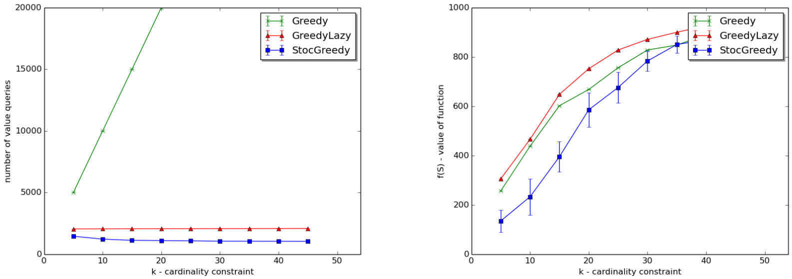
StocGreedy

- In each round, instead of considering all $V \setminus S$ to get

$$e \leftarrow \underset{e \in V \setminus S,\ S \cup \{e\} \in \mathcal{I}}{\arg\max} \Delta_f(e|S),$$

- consider only $\frac{|V|}{k} \log \frac{1}{\epsilon}$ random samples from $V \setminus S$.
- $(1 - e^{-1} - \epsilon)$-approximation in expectation.

Comparison



(a) Efficiency (b) Quality

Figure : Experiment on SYNTHETIC dataset

Summary of state of the arts

constraint	monotone	non-negative
cardinality	$1 - 1/e$ [10]	$1/e + .004$ [1]
matroid	$1 - 1/e$ [2], R	$\frac{1-\epsilon}{e}$ [3], R
matching	$\frac{1}{2+\epsilon}$ [4]	$\frac{1}{4+\epsilon}$ [4]
intersection of p matroids	$\frac{1}{p+\epsilon}$ [7]	$\frac{p-1}{p^2+\epsilon}$ [7]
p-matchoid	$\frac{1}{p+1}$ [2, 10]	$\frac{(1-\epsilon)(2-o(1))}{e \cdot p}$ [4, 11], R

Table : Best known approximation bounds for submodular maximization in RAM model. Bounds for randomized algorithms that hold in expectation are marked (R).

Overview of 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

Set Cover Problem

- Let E be a fixed set with finite size.
- $V = \{C_1, C_2, \ldots, C_n\}$ where each $C_i \subseteq E$.
- We define a function $f : 2^V \to \mathbb{R}$ such that $f(S) = |\cup_{C \in S} C|$.
- Goal: pick $S \subseteq V$ with $|S| \leq k$ that maximizes $f(S)$
- $f(S)$ is monotone submodular and this is a submodular maximization problem s.t. cardinality constraint!

Kernel Machines

The data set $V = \{x_1, x_2, \ldots, x_n\}$ is represented in a transformed space via a kernel matrix

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

where $\mathcal{K} : V \times V \to \mathbb{R}$ is the kernel function that is symmetric and positive definite.

Kernel Machines cont.

- K_V is large for large $|V|$, need to select a subset from V.
- How to measure the quality of selected subset?
- A popular way is to use *Informative Vector Machine* (IVM) introduced by Laurence et al. [6]:

$$f(S) = \frac{1}{2} \log \det \left(\mathbf{I} + \sigma^{-2} K_S \right)$$

- $f(S)$ is submodular!
- Goal:

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

The model

The ground set V is an ordered sequence of items $e_1, e_2, \ldots, e_n$. We process the items one by one and the maximum space being used should be sublinear (i.e. $o(n)$).

CHEN : *insert a graph here*

SieveStream assume OPT is known

Algorithm 2: SieveStreamOPT for submodular maximization

Input: V as data stream, f a monotone submodular function, k the size constraint, OPT the optimal value of $f(S)$ under the constraint

Output: a set $S \subseteq V$

```
S ← ∅
for each e in the data stream do
    if Δ(e|S) ≥ (OPT/2 − f(S)) / (k − |S|) and |S| < k then
        S ← S ∪ {e}
return S
```

SieveStream assume OPT is unknown

Problems with SieveStreamOPT

OPT is unknown!

So what we do?

Solution

- $m = \max_{e \in V} f(\{e\})$, for simplicity, assume $f(\emptyset) = \emptyset$
- note that $m \leq \text{OPT} \leq k \cdot m$
- if we know m, we guess OPT as $m, (1+\epsilon)m, (1+\epsilon)^2 m, \ldots \leq k \cdot m$, each guess runs an instance of SieveStreamOPT
- it runs only $O(\log_{(1+\epsilon)}) = O(\frac{k}{\epsilon})$ instances

SIEVESTREAM assume OPT is unknown, cont.

Problem again

calculating $m = \max_{e \in V} f(\{e\})$ requires an extra pass!

Solution?

Solution

- update $m \leftarrow \max(f(e_i, m))$ on the fly!
- lazy-evaluation, create an instance of SIEVESTREAMOPT only when necessary
- it runs only $O(\log_{(1+\epsilon)}) = O(\frac{k}{\epsilon})$ instances, using only 1 pass
- guarantee $(1/2 - \epsilon)$-approximation for monotone submodular maximization s.t. cardinality constraint

RandomStream

Algorithm 3: RandomStream for submodular maximization

Input: V as data stream, f a non-negative submodular function, k the cardinality constraint, ϵ a parameter

Output: a set $S \subseteq V$

```
B ← ∅, S ← ∅
for each e in the data stream do
    if |S| < k and Δ(e|S) > α then
        B ← B + e
    if |B| > k/ε then
        e ← uniformly random from B
        B ← B − e, S ← S + e
        for all e′ ∈ B s.t. Δ(e′|S) ≤ α do
            B ← B − e′
S′ ← offline algorithm on B
return arg max_{A∈{S,S′}} f(A)
```

Thank you!

N. Buchbinder, M. Feldman, J. S. Naor, and R. Schwartz.
Submodular maximization with cardinality constraints.
In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1433–1452. SIAM, 2014.

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.

M. Feldman, J. Naor, and R. Schwartz.
A unified continuous greedy algorithm for submodular maximization.
In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 570–579. IEEE, 2011.

M. Feldman, J. S. Naor, R. Schwartz, and J. Ward.
Improved approximations for k-exchange systems.
In *Algorithms–ESA 2011*, pages 784–798. Springer, 2011.

T. A. Jenkyns.
The efficacy of the greedy algorithm.
In *Proceedings of the 7th Southeastern Conference on Combinatorics, Graph Theory, and Computing, Utilitas Mathematica, Winnipeg*, pages 341–350, 1976.

N. Lawrence, M. Seeger, and R. Herbrich.
Fast sparse gaussian process methods: The informative vector machine.
In *Proceedings of the 16th Annual Conference on Neural Information Processing Systems*, number EPFL-CONF-161319, pages 609–616, 2003.

J. Lee, M. Sviridenko, and J. Vondrák.
Submodular maximization over multiple matroids via generalized exchange properties.
Mathematics of Operations Research, 35(4):795–806, 2010.

M. Minoux.

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

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

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.

J. Vondrák, C. Chekuri, and R. Zenklusen.
Submodular function maximization via the multilinear relaxation and contention resolution schemes.
In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 783–792. ACM, 2011.