Learning and Testing Submodular Functions

Sofya Raskhodnikova, Grigory Yaroslavtsev
Pennsylavania State Univeristy

Submodularity

- Discrete analog of concavity, captures law of diminishing returns
- Applications: matroids, valuations in AGT, etc.

Definitions (for functions $f: 2^X \to R$)

Discrete derivative:

$$\partial_x f(S) = f(S \cup \{x\}) - f(S) \qquad for\ S \subseteq X, x \notin S$$

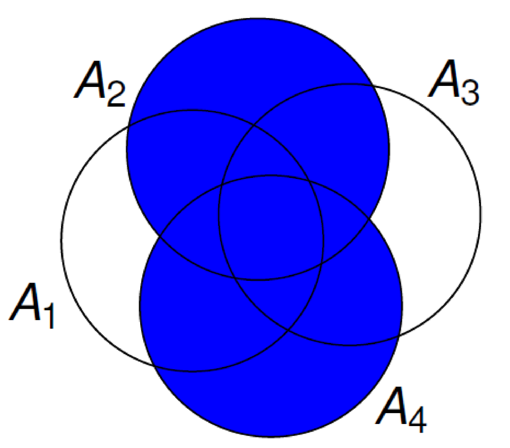
Submodular function:

$$\partial_x f(S) \geq \partial_x f(T) \qquad for\ all\ S \subseteq T \subseteq X$$

Examples:

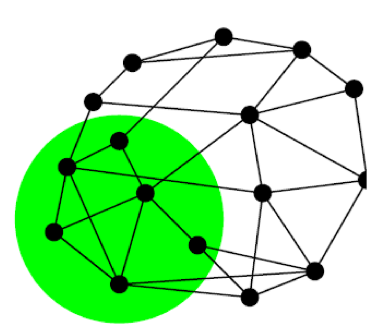
Coverage function:
Given $A_1, \ldots, A_n \subset U$,

$$f(S) = \left|\bigcup_{j \in S} A_j\right|.$$



Cut function:

$$\delta(T) = |e(T, \overline{T})|$$



Pseudo-Boolean k-DNF ($\vee \to max,\ A_i \in \{0, \ldots, k\}$):

- $max_i\ (A_i \cdot (x_{i_1} \wedge \overline{x_{i_2}} \ldots \wedge x_{i_k}))$

Main results

Structural result: Every **monotone** submodular function $f: 2^X \to \{0, \ldots, k\}$ can be represented as a **monotone** pseudo-Boolean k-DNF.

Learning and testing: polynomial query complexity for k = o(log n / log log n) for **monotone** functions, where n = $|X|$.

Previous work

Property testing
[Seshadhri, Vondrak, ICS'11]:
- Upper bound $(1/\epsilon)^{O(\sqrt{n})}$.
- Lower bound: $\Omega(n)$

$\}$ **Gap in query complexity**

Special case: coverage functions [Chakrabarty, Huang, ICALP'12].

Learning everywhere (with membership queries) [Goemans, Harvey, Iwata, Mirrokni, SODA'09]: $\tilde{\Theta}(\sqrt{n})$-approximation.

PAC-like learning Lipschitz submodular functions (under uniform/product distributions)
- Multiplicative error [Balcan, Harvey, STOC'11]
- Additive error [Gupta, Hardt, Roth, Ullman, STOC'11]

SQ-learning submodular functions with additive error [Cheraghchi, Klivans, Kothari, Lee, SODA'11]

How about bounded integral range? (Open problem from the workshop on sublinear algorithms in Bertinoro'11)

Let $f: 2^X \to \{0, \ldots, k\}$, where k is a constant.

Case study: k = 1 (Boolean functions)
- Monotone submodular = $x_{i_1} \vee x_{i_2} \vee \cdots \vee x_{i_a}$ (monomial)
- Submodular = $(x_{i_1} \vee \cdots \vee x_{i_a}) \wedge (\overline{x_{j_1}} \vee \cdots \vee \overline{x_{j_b}})$ (2-term CNF)

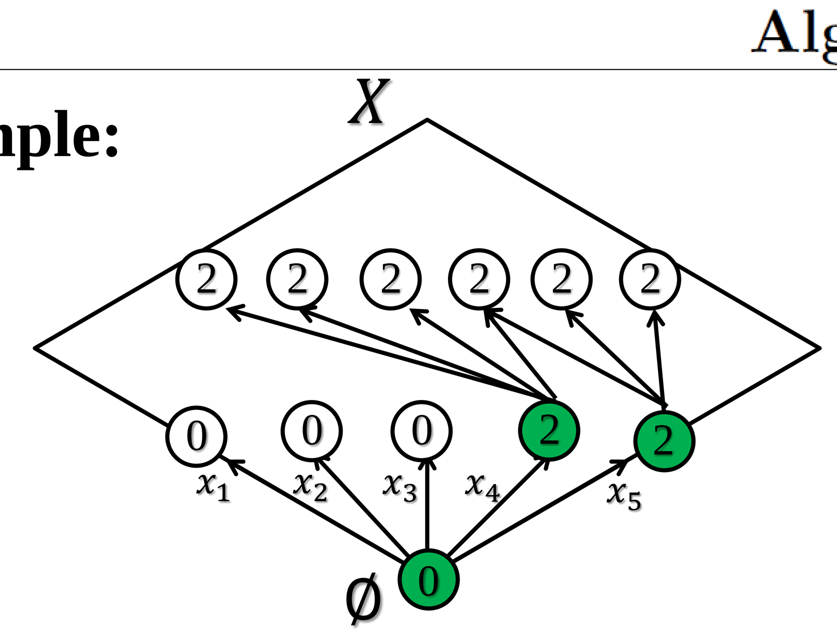
Theorem: Every **monotone** submodular function $f: 2^X \to \{0, \ldots, k\}$ can be represented as a **monotone** pseudo-Boolean k-DNF.

Proof : Run Build-DNF($f, \emptyset$).

```
input  : Function f: 2^X → {0,...,k}, argument Y ∈ 2^X.
output: Collection C of monotone clauses of width at most k.
C ← {f(Y) · ⋀_{i∈Y} x_i}
for j ∉ Y do
    if f(Y ∪ {j}) > f(Y) then
        C ← C ∪ BUILD-DNF(f, Y ∪ {j}).
    end
end
return C
```

Algorithm 1: BUILD-DNF(f, Y).

Example:



$\max(2x_4, 2x_5)$

Corollaries for monotone submodular functions:
- $O(n^k)$ query complexity exact learning (trivial, matches previous work).
- $O(n\, k^{k \log \frac{1}{\epsilon}})$ query complexity PAC-learning under uniform distribution (we can generalize Kushilevitz-Mansour learning algorithm to pseudo-Boolean k-DNF, requires **nontrivial modification of Hastad's switching lemma**).
- $O(n\, k^{k \log \frac{1}{\epsilon}})$ query complexity property tester via transformation from a learning algorithm [Goldrecich, Goldwasser, Ron '98]. Running time not preserved, because learning is not proper.

What's next?

- Improve algorithms for general range?
- Give testing algorithms for bounded range without proper learning?
- Prove better lower bounds via communication complexity/information-theoretic arguments?