Learning and Testing Submodular Functions

Grigory Yaroslavtsev

http://grigory.us

Slides at
http://grigory.us/cis625/lecture3.pdf

CIS 625: Computational Learning Theory

Penn
University of Pennsylvania

Submodularity

- Discrete analog of convexity/concavity, “law of diminishing returns”
- Applications: combintorial optimization, AGT, etc.

Let $\boldsymbol{f}: 2^X \to [0, R]$:

- **Discrete derivative:**

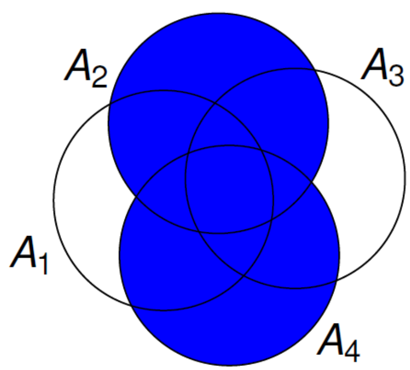
$$\partial_x \boldsymbol{f}(S) = \boldsymbol{f}(S \cup \{x\}) - \boldsymbol{f}(S), \qquad for\ S \subseteq X, x \notin S$$

- **Submodular function:**

$$\partial_x \boldsymbol{f}(S) \geq \partial_x \boldsymbol{f}(T), \quad \forall\, S \subseteq T \subseteq X, x \notin T$$

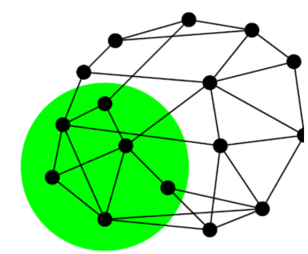
Coverage function:
Given $A_1, \ldots, A_n \subset U$,

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



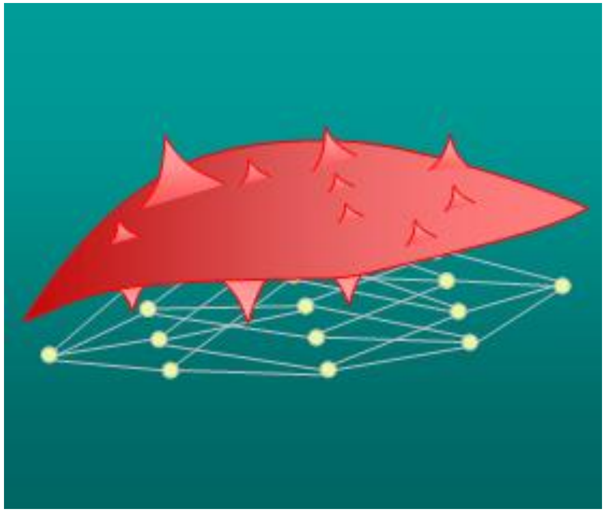
Cut function:

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



Approximating everywhere

- **Q1:** Approximate a submodular $\boldsymbol{f}: 2^X \to [0, R]$ **for all arguments** with only poly(|X|) queries?
- **A1:** Only $\tilde{\Theta}\left(\sqrt{|X|}\right)$-approximation (multiplicative) possible [Goemans, Harvey, Iwata, Mirrokni, SODA'09]



- **Q2:** Only for $(1-\epsilon)$-fraction of arguments (PAC-style learning **with membership queries** under uniform distribution)?

$$\Pr_{randomness\ of\ \boldsymbol{A}}\left[\Pr_{\boldsymbol{S}\sim U(2^X)}[\boldsymbol{A}(\boldsymbol{S}) = \boldsymbol{f}(\boldsymbol{S})] \geq 1-\epsilon\right] \geq \frac{1}{2}$$

- **A2:** Almost as hard [Balcan, Harvey, STOC'11].

Approximate learning

- **PMAC**-learning (**M**ultiplicative), with poly(|X|) queries :

$$\Pr_{rand.\,of\,\boldsymbol{A}}\left[\Pr_{\boldsymbol{S}\sim U(2^X)}\left[\frac{\boldsymbol{1}}{\boldsymbol{\alpha}}\boldsymbol{f}(\boldsymbol{S}) \le \boldsymbol{A}(\boldsymbol{S}) \le \boldsymbol{\alpha}\boldsymbol{f}(\boldsymbol{S})\right] \ge 1-\epsilon\right] \ge \frac{1}{2}$$

$\Omega\left(|X|^{\frac{1}{3}}\right) \le \boldsymbol{\alpha} \le O\left(\sqrt{|X|}\right)$ [Balcan, Harvey '11]

- **PAAC**-learning (**A**dditive)

$$\Pr_{rand.\,of\,\boldsymbol{A}}\left[\Pr_{\boldsymbol{S}\sim U(2^X)}\left[\,|\boldsymbol{f}(\boldsymbol{S}) - \boldsymbol{A}(\boldsymbol{S})| \le \boldsymbol{\beta}\right] \ge 1-\epsilon\right] \ge \frac{1}{2}$$

  - Running time: $|X|^{O\left(\frac{R}{\beta}\right)^2 \log(\frac{1}{\epsilon})}$ [Gupta, Hardt, Roth, Ullman, STOC'11]
  - Running time: $\mathrm{poly}\left(|X|^{\left(\frac{R}{\beta}\right)^2},\ \log\frac{1}{\epsilon}\right)$ [Cheraghchi, Klivans, Kothari, Lee, SODA'12]

Learning $f: 2^X \to [0, R]$

- For all algorithms $\epsilon = const.$

	Goemans, Harvey, Iwata, Mirrokni	Balcan, Harvey	Gupta, Hardt, Roth, Ullman	Cheraghchi, Klivans, Kothari, Lee	Raskhodnikova, Y.
Learning	$\tilde{O}\left(\sqrt{\lvert X \rvert}\right)$-approximation Everywhere	PMAC **M**ultiplicative $\boldsymbol{\alpha}$ $\boldsymbol{\alpha} = O\left(\sqrt{\lvert X \rvert}\right)$	PAAC **A**dditive $\boldsymbol{\beta}$		**PAC** $\boldsymbol{f: 2^X \to \{0, \dots, R\}}$ (bounded integral range $R \leq \lvert X \rvert$)
Time	Poly(|X|)	Poly(|X|)	$\lvert X \rvert^{O\left(\frac{R}{\beta}\right)^2}$		$\lvert X \rvert^3 R^{O(R \cdot \log R)}$ Polylog(|X|) $R^{O(R \cdot \log R)}$ queries
Extra features		Under arbitrary distribution	Tolerant queries	SQ-queries, Agnostic	

Learning: Bigger picture

Subadditive

$\subseteq$

XOS = Fractionally subadditive

$\subseteq$

Submodular

$\subseteq$

Gross substitutes

$\subseteq$

OXS

$\subseteq$ $\subseteq$

Additive (linear) Coverage (valuations)

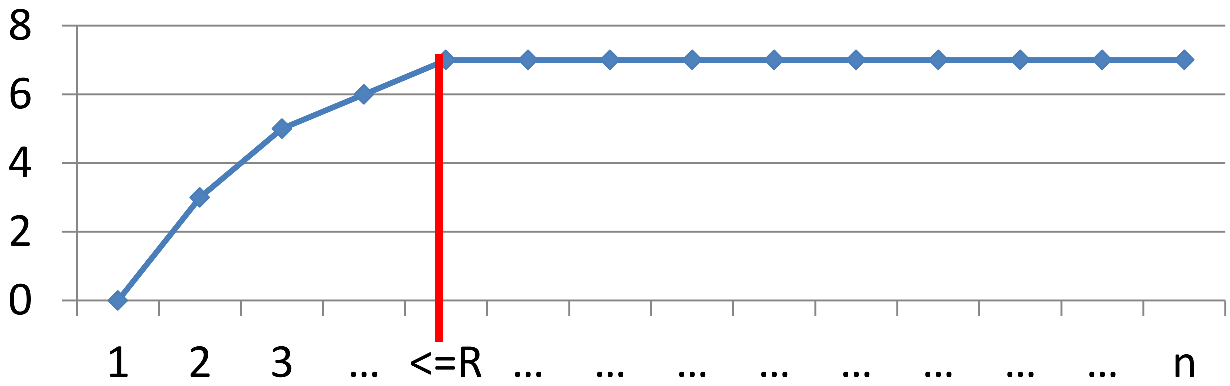
} [Badanidiyuru, Dobzinski, Fu, Kleinberg, Nisan, Roughgarden,SODA'12]

Other positive results:

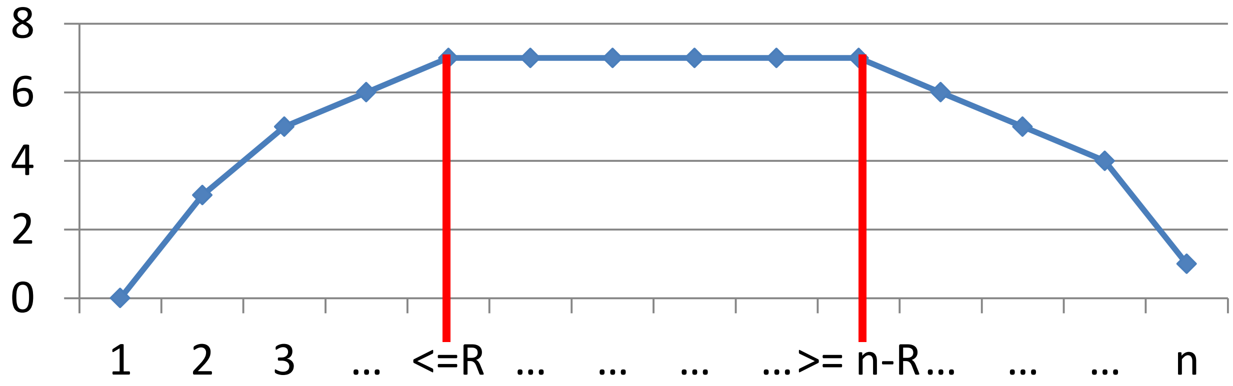
- Learning valuation functions [Balcan, Constantin, Iwata, Wang, COLT'12]
- $(1 + \epsilon)$ PMAC-learning (sketching) coverage functions [BDFKNR'12]
- $(1 + \epsilon)$ PMAC learning Lipschitz submodular functions [BH'10] (concentration around average via Talagrand)

Discrete convexity

- Monotone convex $f: \{1, \ldots, n\} \to \{0, \ldots, R\}$



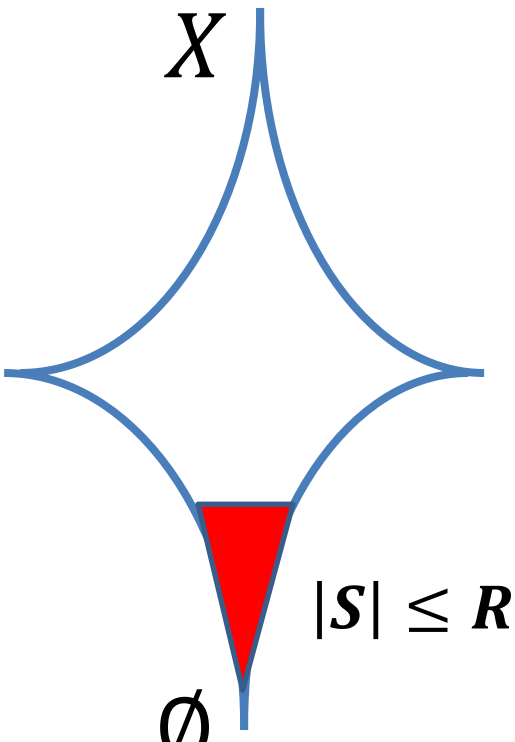
- Convex $f: \{1, \ldots, n\} \to \{0, \ldots, R\}$



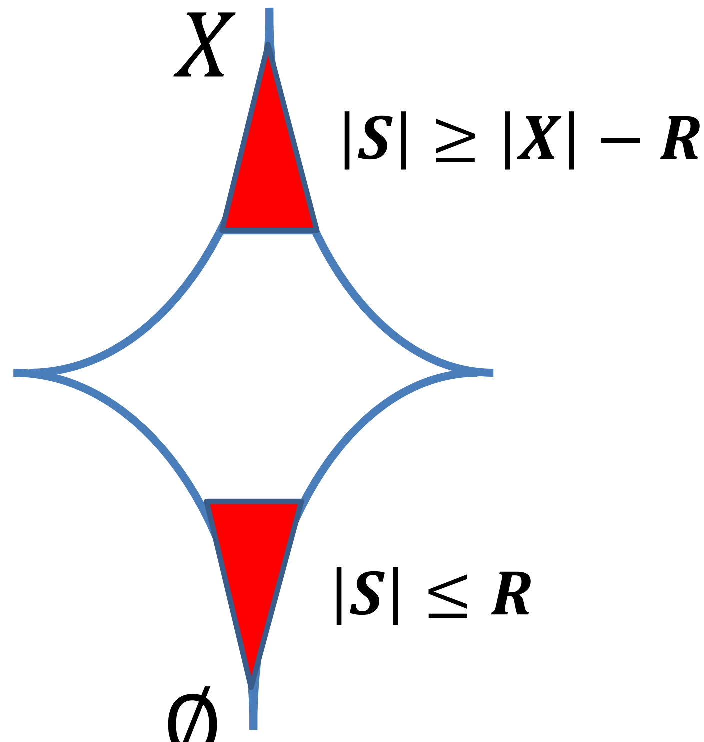
Discrete submodularity $f: 2^X \to \{0, \ldots, R\}$

- **Case study**: R = 1 (Boolean submodular functions $f: \{0,1\}^n \to \{0,1\}$)
  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)

- Monotone submodular

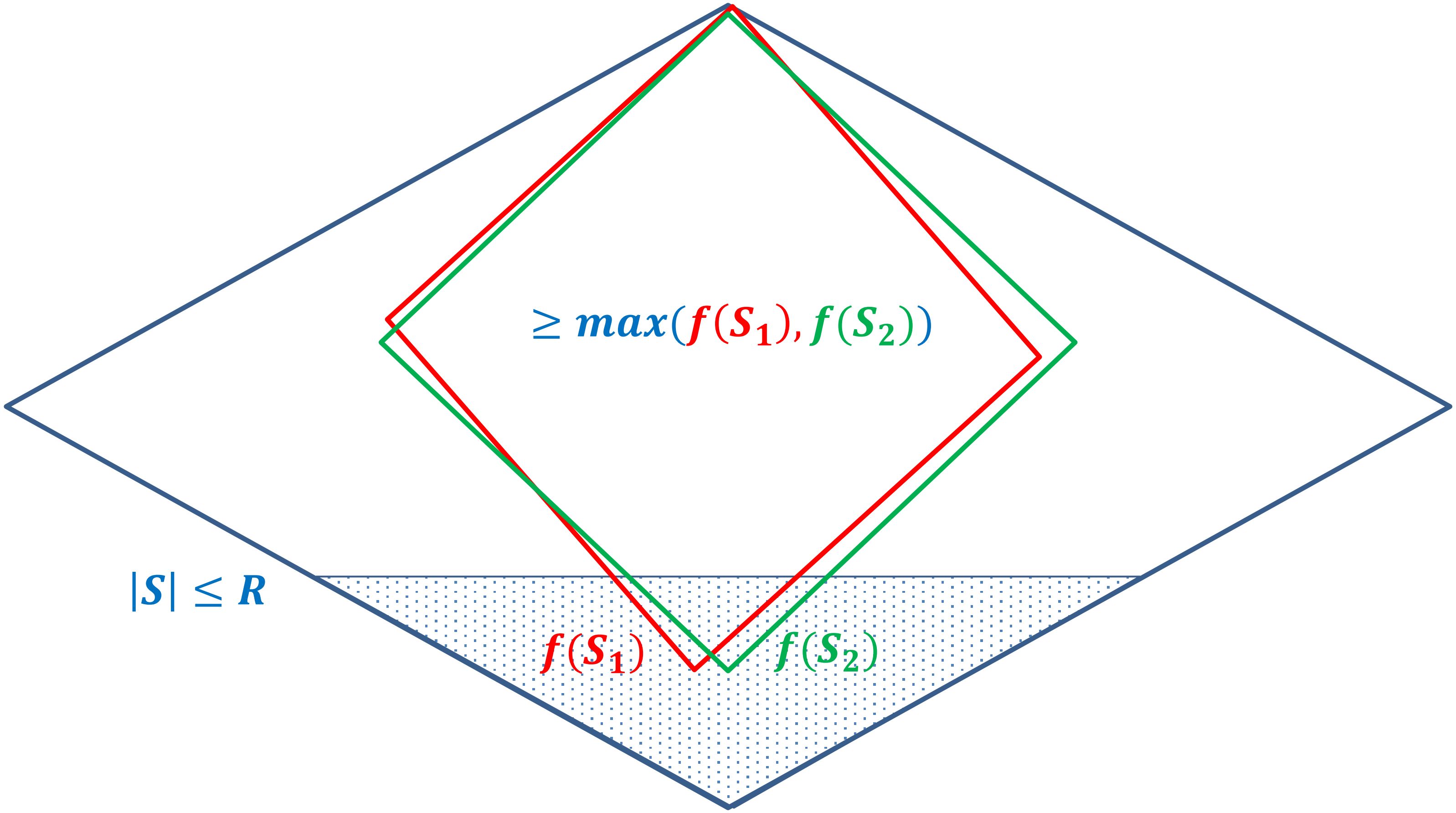


- Submodular



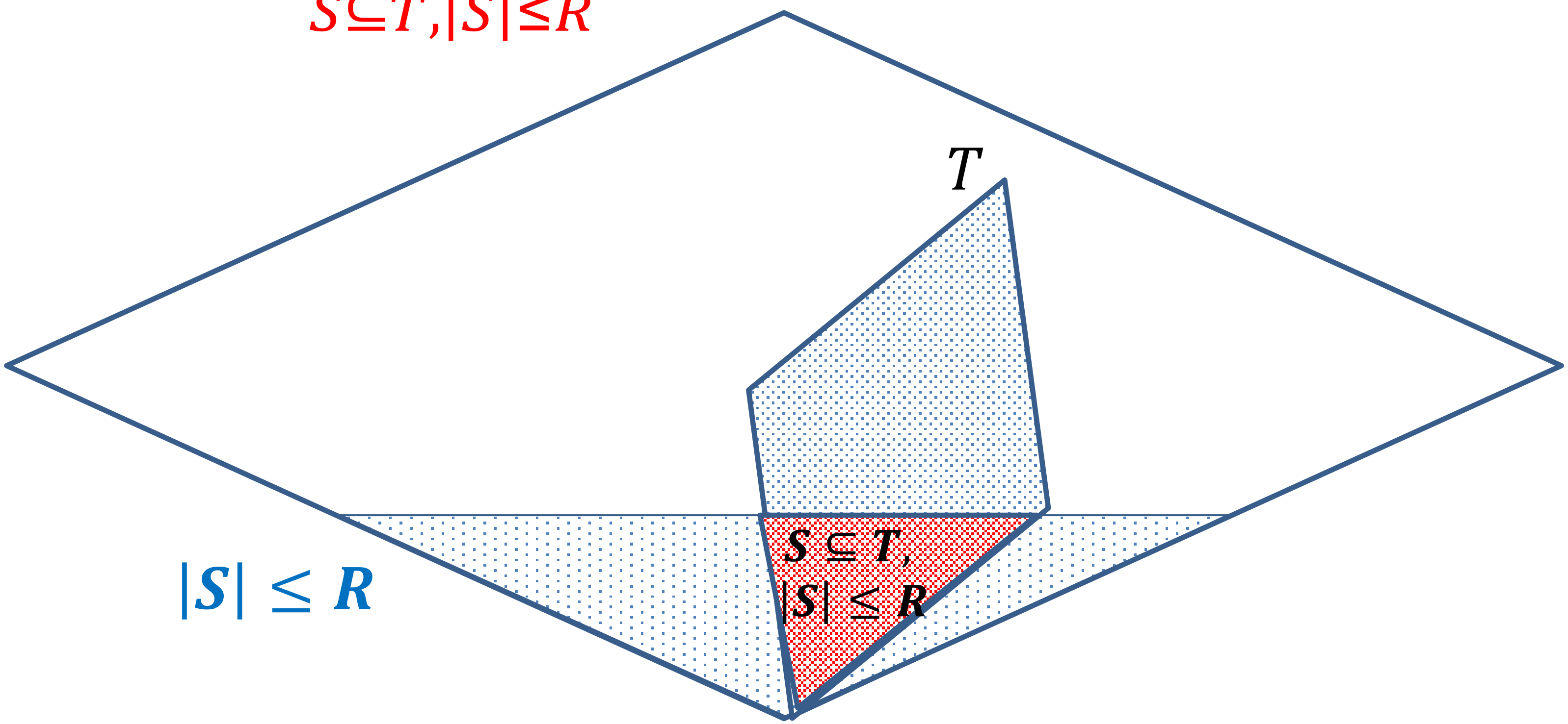
Discrete monotone submodularity

- Monotone submodular $f: 2^X \to \{0, \ldots, R\}$



Discrete monotone submodularity

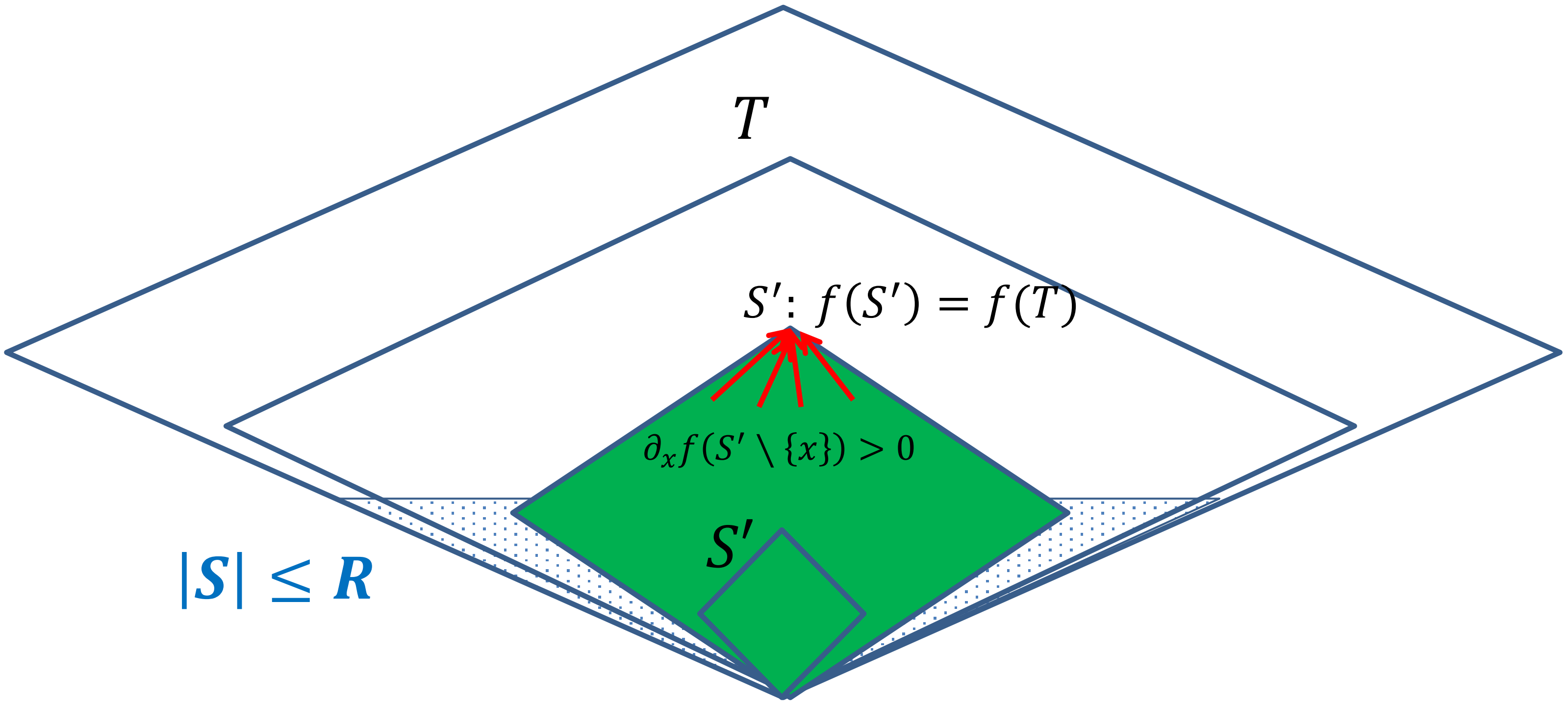
- **Theorem**: for **monotone** submodular $f: 2^X \to \{0, \ldots, R\}$ for all T: $f(T) = \max_{S \subseteq T, |S| \leq R} f(S)$
- $f(T) \geq \max_{S \subseteq T, |S| \leq R} f(S)$ (by monotonicity)



Discrete monotone submodularity

- $f(T) \le \max_{S \subseteq T, |S| \le R} f(S)$
- S’ $=$ **smallest** subset of T such that $f(T) = f(S')$
- $\forall x \in S'$ we have $\partial_x f(S' \setminus \{x\}) > 0$ =>

Restriction of f on $2^{S'}$ is **monotone increasing** =>$|S'| \le R$



Representation by a formula

- **Theorem**: for **monotone** submodular $f: 2^X \to \{0, \ldots, R\}$ for all T:

$$f(T) = \max_{S \subseteq T, |S| \leq R} f(S)$$

- Alternative notation: $|X| \to n$, $2^X \to (x_1, \ldots, x_n)$
- **Boolean k–DNF =** $\bigvee \left(x_{i_1} \wedge \overline{x_{i_2}} \wedge \cdots \wedge x_{i_k}\right)$
- **Pseudo–Boolean k–DNF** ($\vee \to \boldsymbol{max}$, $A_i = 1 \to \boldsymbol{A_i} \in$ R):

$\boldsymbol{max_i} \left[\boldsymbol{A_i} \cdot \left(x_{i_1} \wedge \overline{x_{i_2}} \wedge \cdots \wedge x_{i_k}\right)\right]$ (**Monotone, if no negations**)

- **Theorem (restated)**:

Monotone submodular $f(x_1, \ldots, x_n) \to \{0, \ldots, \boldsymbol{R}\}$ can be represented as a **monotone** pseudo-Boolean $\boldsymbol{R}$-DNF with constants $A_i \in \{0, \ldots, \boldsymbol{R}\}$.

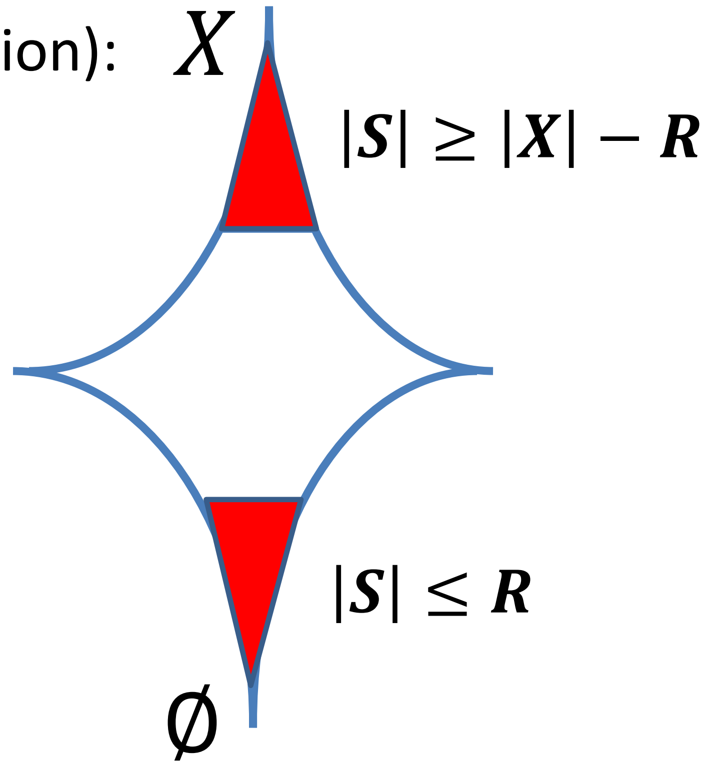
Discrete submodularity

- Submodular $f(x_1, \ldots, x_n) \to \{0, \ldots, \boldsymbol{R}\}$ can be represented as a pseudo-Boolean **2R**-DNF with constants $A_i \in \{0, \ldots, \boldsymbol{R}\}$.
- Hint [Lovasz] (Submodular monotonization):

Given submodular $\boldsymbol{f}$, define

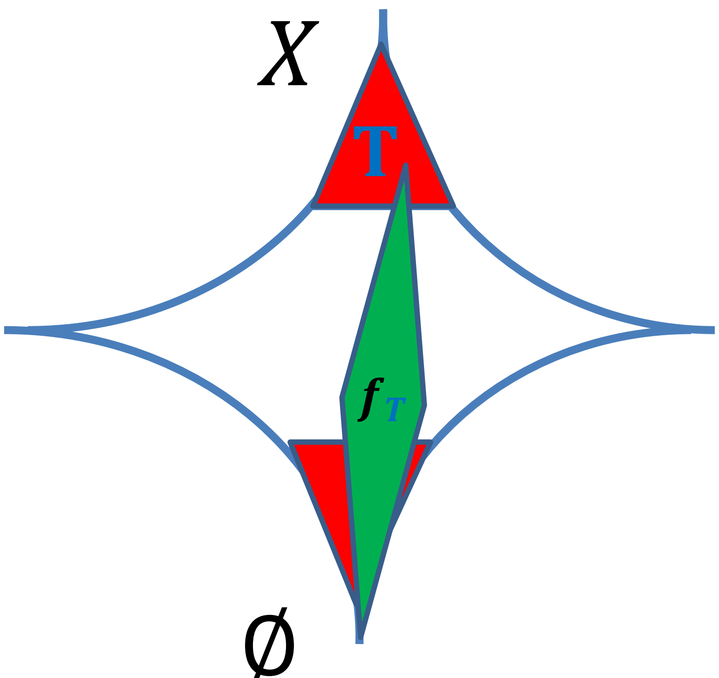
$$\boldsymbol{f^{mon}(S) = min_{S \subseteq T}\, f(T)}$$

Then $\boldsymbol{f^{mon}}$ is monotone and **submodular**.



Proof

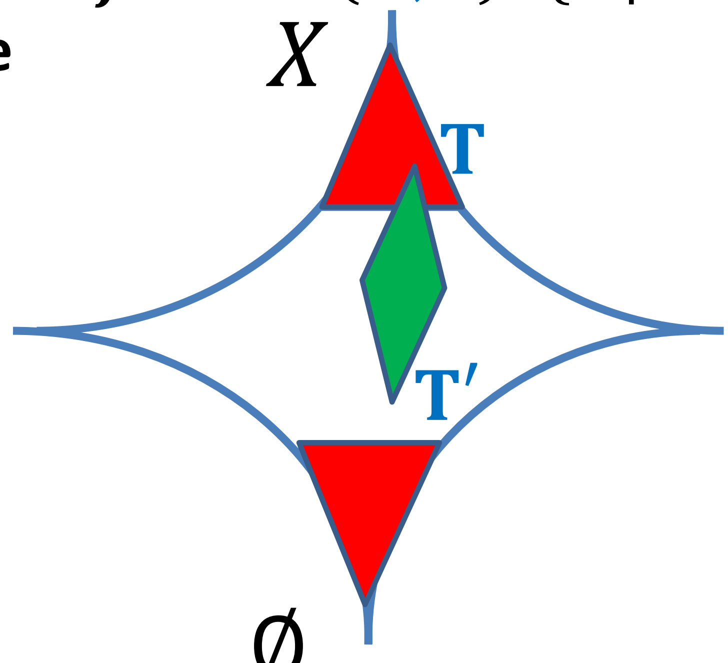
- We're done if we have a **coverage** $\boldsymbol{C} \subseteq 2^X$:
  1. All $\mathbf{T} \in \boldsymbol{C}$ have large size: $|\mathbf{T}| \geq |\boldsymbol{X}| - \boldsymbol{R}$
  2. For all $\boldsymbol{S} \in 2^{\boldsymbol{X}}$ there exists $\mathbf{T} \in \boldsymbol{C} : \boldsymbol{S} \subseteq \boldsymbol{T}$
  3. For every $\mathbf{T} \in \boldsymbol{C}$ restriction $\boldsymbol{f}_{\boldsymbol{T}}$ of $\boldsymbol{f}$ on $2^{\boldsymbol{T}}$ is **monotone**



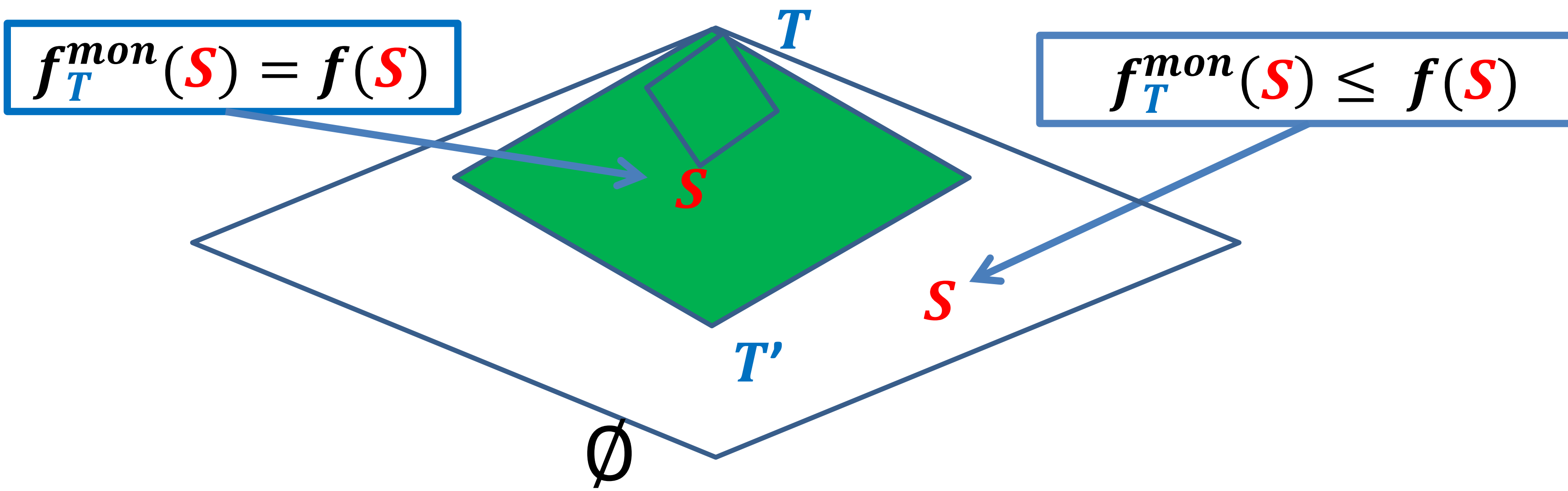
- Every $\boldsymbol{f}_{\boldsymbol{T}}$ is a monotone pB **R**-DNF **(3)**
- Add at most **R** negated variables to every clause to restrict to $2^{\boldsymbol{T}}$ **(1)**
- $\boldsymbol{f}(S) = \max_{\boldsymbol{T} \in \boldsymbol{C}} \boldsymbol{f}_{\boldsymbol{T}}(S)$ **(2)**

Proof

- There is no such coverage => relaxation [GHRU'11]
  - All $\mathbf{T} \in \boldsymbol{C}$ have large size: $|\mathbf{T}| \geq |\boldsymbol{X}| - \boldsymbol{R}$
  - For all $\boldsymbol{S} \in 2^X$ there exists a pair $\mathbf{T}' \subseteq \boldsymbol{T} \in \boldsymbol{C}$:
  $$\mathbf{T}' \subseteq \boldsymbol{S} \subseteq \boldsymbol{T}$$
  - Restriction of $\boldsymbol{f}$ on all $r(\boldsymbol{T}', \boldsymbol{T})$: $\{\boldsymbol{S} \mid \mathbf{T}' \subseteq \boldsymbol{S} \subseteq \boldsymbol{T}\}$ is **monotone**



Coverage by monotone lower bounds



- Let f_T^{mon} be defined as $f_T^{mon}(S) = \min_{S \subseteq S' \subseteq T} f(S')$
  - f_T^{mon} is monotone submodular [Lovasz]
  - For all $S \subseteq T$ we have $f_T^{mon}(S) \leq f(S)$
  - For all $\mathbf{T}' \subseteq S \subseteq T$ we have $f_T^{mon}(S) = f(S)$
- $f(S) = \max_{T \in C} f_T^{mon}(S)$ (where f_T^{mon} is a monotone pB **R**-DNF)

Learning pB-formulas and k-DNF

- $DNF^{\boldsymbol{k},\boldsymbol{R}}$ = class of pB $\boldsymbol{k}$-DNF with $A_i \in \{0, \dots, \boldsymbol{R}\}$
- **i-slice** $\boldsymbol{f}_i(x_1, \dots, x_n) \to \{0,1\}$ defined as

$$\boldsymbol{f}_i(x_1, \dots, x_n) = 1 \quad \textbf{iff} \quad \boldsymbol{f}(x_1, \dots, x_n) \geq i$$

- If $\boldsymbol{f} \in DNF^{\boldsymbol{k},\boldsymbol{R}}$ its **i-slices** $\boldsymbol{f}_i$ are $\boldsymbol{k}$-DNF and:

$$\boldsymbol{f}(x_1, \dots, x_n) = \max_{1 \leq i \leq \boldsymbol{R}} (i \cdot \boldsymbol{f}_i(x_1, \dots, x_n))$$

- PAC-learning:

$$\Pr_{rand(\boldsymbol{A})}\left[\Pr_{\boldsymbol{S} \sim U(\{0,1\}^n)}[\boldsymbol{A}(\boldsymbol{S}) = \boldsymbol{f}(\boldsymbol{S})] \geq 1 - \epsilon\right] \geq \frac{1}{2}$$

- Learn every **i-slice** $\boldsymbol{f}_i$ on $(1 - \epsilon / R)$ fraction of arguments => union bound

Learning Fourier coefficients

- Learn f_i (k-DNF) on $1 - \epsilon' = (1 - \epsilon\ /\ R)$ fraction of arguments

- **Fourier sparsity** $S_C(\epsilon)$ = # of largest Fourier coefficients sufficient to PAC-learn every $f \in C$

- $S_{k\text{–DNF}}(\epsilon)$ = $k^{O(k \log(\frac{1}{\epsilon}))}$ [Mansour]: doesn't depend on **n**!
  - Kushilevitz-Mansour (Goldreich-Levin): $poly(n, S_F)$ queries/time.
  - ``Attribute efficient learning'': $\boldsymbol{polylog}(n) \cdot poly(S_F)$ queries
  - Lower bound: $\Omega(2^k)$ queries to learn a random k-junta ($\in k$-DNF) up to constant precision.

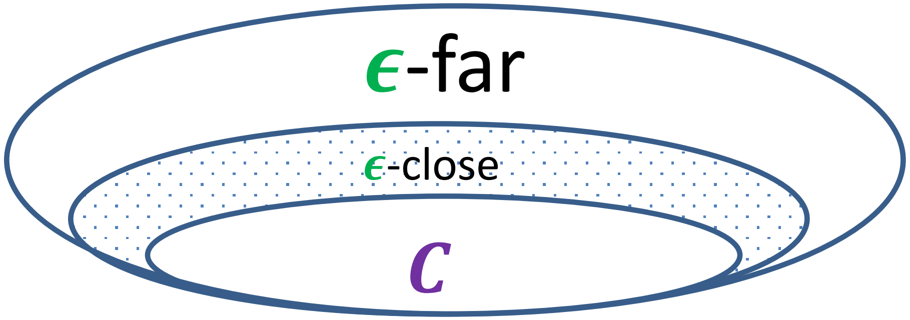
- $S_{DNF^{k,R}}(\epsilon)$ = $k^{O(k \log(\frac{R}{\epsilon}))}$
  - Optimizations: Do all **R** iterations of KM/GL in parallel by reusing queries

Property testing

- Let $\boldsymbol{C}$ be the class of submodular $\boldsymbol{f}: \{0,1\}^n \to \{0, \dots, \boldsymbol{R}\}$
- How to (approximately) test, whether a given $\boldsymbol{f}$ is in $\boldsymbol{C}$?
- Property tester: (randomized) algorithm for distinguishing:

  1. $f \in \boldsymbol{C}$

  2. (ϵ-far): $\min_{g \in \boldsymbol{C}} |\boldsymbol{f} - \boldsymbol{g}|_{\boldsymbol{H}} \geq \epsilon\, 2^n$



- Key idea: $\boldsymbol{k}$-DNFs have small representations:
  - [Gopalan, Meka,Reingold CCC'12] (using quasi-sunflowers [Rossman'10])

  $\forall \epsilon > 0$, $\forall$ $\boldsymbol{k}$-DNF formula F there exists:

  $\boldsymbol{k}$-DNF formula F' of size $\leq \left(\boldsymbol{k} \log \frac{1}{\epsilon}\right)^{O(\boldsymbol{k})}$ such that $|F - F'|_H \leq \epsilon 2^n$

Testing by implicit learning

- **Good approximation by juntas => efficient property testing** [Diakonikolas, Lee, Matulef, Onak ,Rubinfeld, Servedio, Wan]
  - ϵ-approximation by $\boldsymbol{J}(\boldsymbol{\epsilon})$-junta
  - Good dependence on $\boldsymbol{\epsilon}$: $\boldsymbol{J}_{\boldsymbol{k}-\mathrm{DNF}}(\boldsymbol{\epsilon}) = \left(\boldsymbol{k} \log \frac{1}{\boldsymbol{\epsilon}}\right)^{O(\boldsymbol{k})}$
- For submodular functions $\boldsymbol{f}: \{0,1\}^n \to \{0, \dots, \boldsymbol{R}\}$
  - Query complexity $\left(\boldsymbol{R} \log \frac{\boldsymbol{R}}{\boldsymbol{\epsilon}}\right)^{\tilde{O}(\boldsymbol{R})}$, independent of **n**!
  - Running time exponential in $\boldsymbol{J}(\boldsymbol{\epsilon})$
  - $\Omega(\boldsymbol{k})$ lower bound for testing $\boldsymbol{k}$-DNF (reduction from Gap Set Intersection)

- [Blais, Onak, Servedio, Y.] **exact** characterization of submodular functions

$$J(\boldsymbol{\epsilon}) = \left[O\left(\boldsymbol{R} \log \boldsymbol{R} + \log \frac{1}{\boldsymbol{\epsilon}}\right)\right]^{(\boldsymbol{R}+\mathbf{1})}$$

Previous work on testing submodularity

$\boldsymbol{f}: \{0,1\}^n \rightarrow [0, R]$ [Parnas, Ron, Rubinfeld '03, Seshadhri, Vondrak, ICS'11]:

- Upper bound $(\mathbf{1}/\boldsymbol{\epsilon})^{\boldsymbol{O}(\sqrt{\boldsymbol{n}})}$.
- Lower bound: $\boldsymbol{\Omega}(\boldsymbol{n})$

} Gap in query complexity

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

Directions

- Close gaps between upper and lower bounds, extend to more general learning/testing settings
- Connections to optimization?
- What if we use L_1 −distance between functions instead of Hamming distance in property testing? [Berman, Raskhodnikova, Y.]