

# Submodular Functions: Definitions, Examples, Optimization Problems and Applications

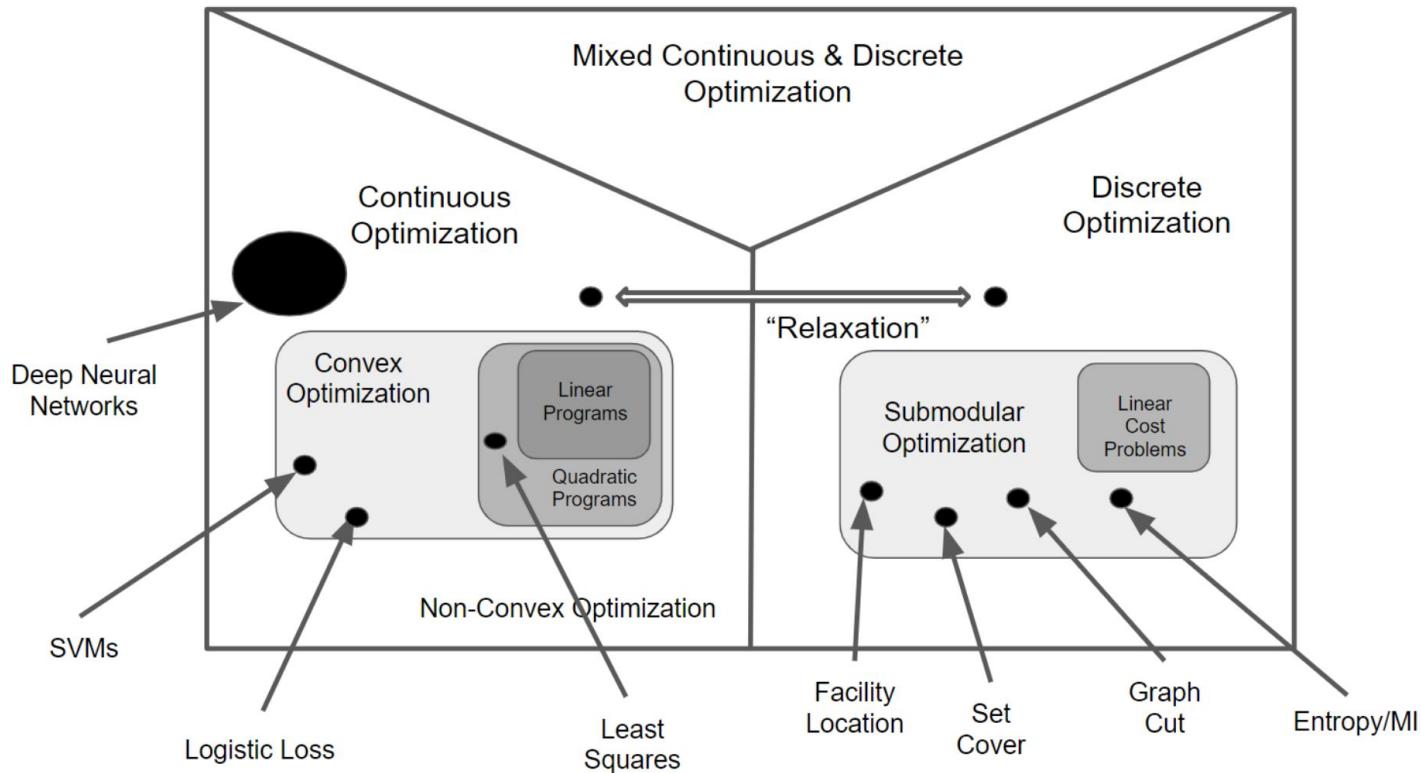
---

**Optimization in Machine Learning**

**Lecture 21 & 22**

# Big Picture: Continuous and Discrete Optimization

---



# Outline

---

- ❑ Discrete Optimization in Machine Learning
- ❑ Lecturer 19 & 20
  - ❑ Definition and Intuition of Submodularity
  - ❑ Modeling Power of Submodular/Set Functions
  - ❑ Examples of Submodular Functions
  - ❑ Examples of Submodular Optimization
- ❑ Lectures 21 & 22
  - ❑ Optimization Algorithms for Different Function Classes and Constraints
  - ❑ Optimization Algorithms for Different Settings (Streaming/Distributed)
  - ❑ Practical Implementation tricks

# Acknowledgements

---

Slides borrowed from several sources:

1. Submodular Optimization course at UW from Jeff Bilmes
2. Tutorial on Submodular Optimization by Stefanie Jegelka, Andreas Krause and Jeff Bilmes at ICML and NIPS
3. Some of my own tutorials at WACV, IJCAI, ECAI, ICPM etc.

# Useful Material

---

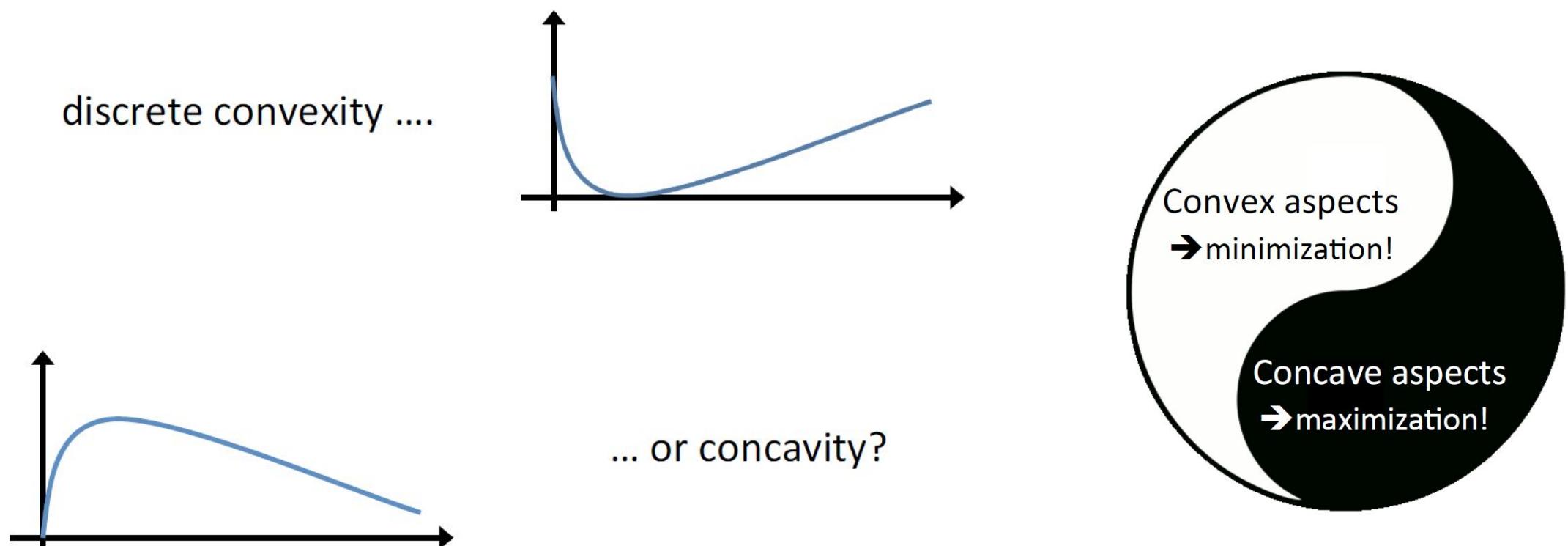
- Fujishige, “Submodular Functions and Optimization”, 2005
- Narayanan, “Submodular Functions and Electrical Networks”, 1997
- Welsh, “Matroid Theory”, 1975
- Oxley, “Matroid Theory”, 1992 (and 2011).
- Lawler, “Combinatorial Optimization: Networks and Matroids”, 1976.
- Schrijver, “Combinatorial Optimization”, 2003
- Gruenbaum, “Convex Polytopes, 2nd Ed”, 2003.
- Additional readings that will be announced here.

# Useful material

---

- Jeff's Class: [http://j.ee.washington.edu/~bilmes/classes/ee596a fall 2014/](http://j.ee.washington.edu/~bilmes/classes/ee596a_fall_2014/)
- Stefanie Jegelka & Andreas Krause's 2013 ICML tutorial:  
<http://techtalks.tv/talks/submodularity-in-machine-learning-new-directions-part-i/58125/>
- Jeff's NIPS, 2013 tutorial on submodularity:  
<http://melodi.ee.washington.edu/~bilmes/pgs/b2hd-bilmes2013-nips-tutorial.html> and  
<http://youtu.be/c4rBof38nKQ>
- Andreas Krause's web page: <http://submodularity.org>
- Francis Bach's updated 2013 text:  
[http://hal.archives-ouvertes.fr/docs/00/87/06/09/PDF/submodular\\_fot\\_revisd\\_hal.pdf](http://hal.archives-ouvertes.fr/docs/00/87/06/09/PDF/submodular_fot_revisd_hal.pdf)
- My WACV 2019 Tutorial: <https://sites.google.com/view/wacv2019summarization/home>
- Tom McCormick's overview paper on submodular minimization:  
<http://people.commerce.ubc.ca/faculty/mccormick/sfmchap8a.pdf>

# Submodularity, Convexity & Concavity

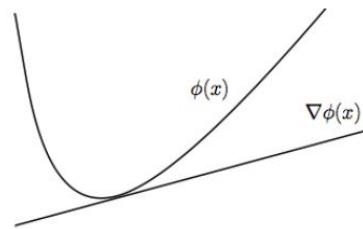


Submodularity is related to both convexity and concavity!

# Submodularity, Convexity & Concavity

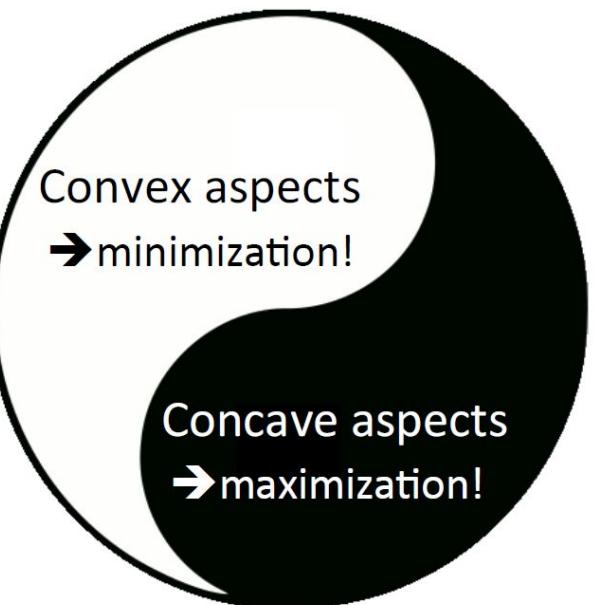
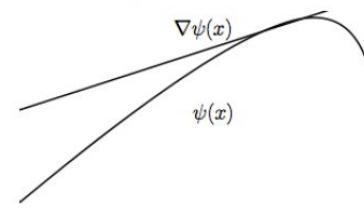
Convex aspects (Fujishige (1984, 2005), Frank (1982))

- Minimization: Poly-time.
- Convex continuous extension - Lovász extension.
- Subgradients and Subdifferential.
- Convex duality, discrete separation etc.



Concave aspects (Vondrak (2007), I-Bilmes (2015))

- Max: constant-factor approx!
- Multilinear extension - concave in a direction.
- Supergradients and Superdifferential.
- Under restricted settings, duality, separation etc.



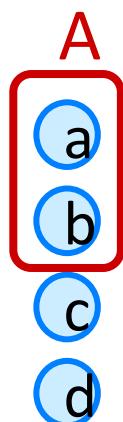
Submodularity is related to both convexity and concavity!

# Set Functions and Boolean Functions

---

any set function  
with  $|V| = n$ .

$$F : 2^V \rightarrow \mathbb{R}$$



$\wedge$

... is a function on  
binary vectors!

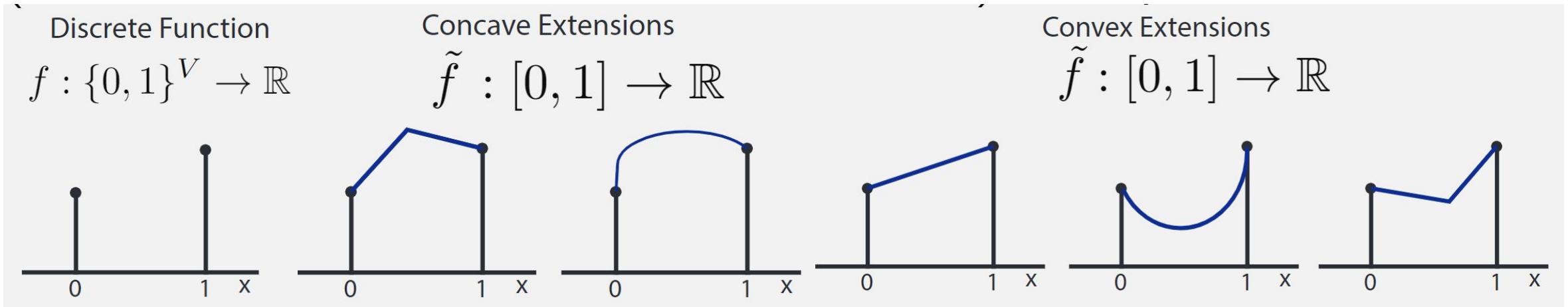
$$F : \{0, 1\}^n \rightarrow \mathbb{R}$$

$$x = e_A$$

1	a
1	b
0	c
0	d

pseudo-boolean  
function

# What are Continuous Extensions?

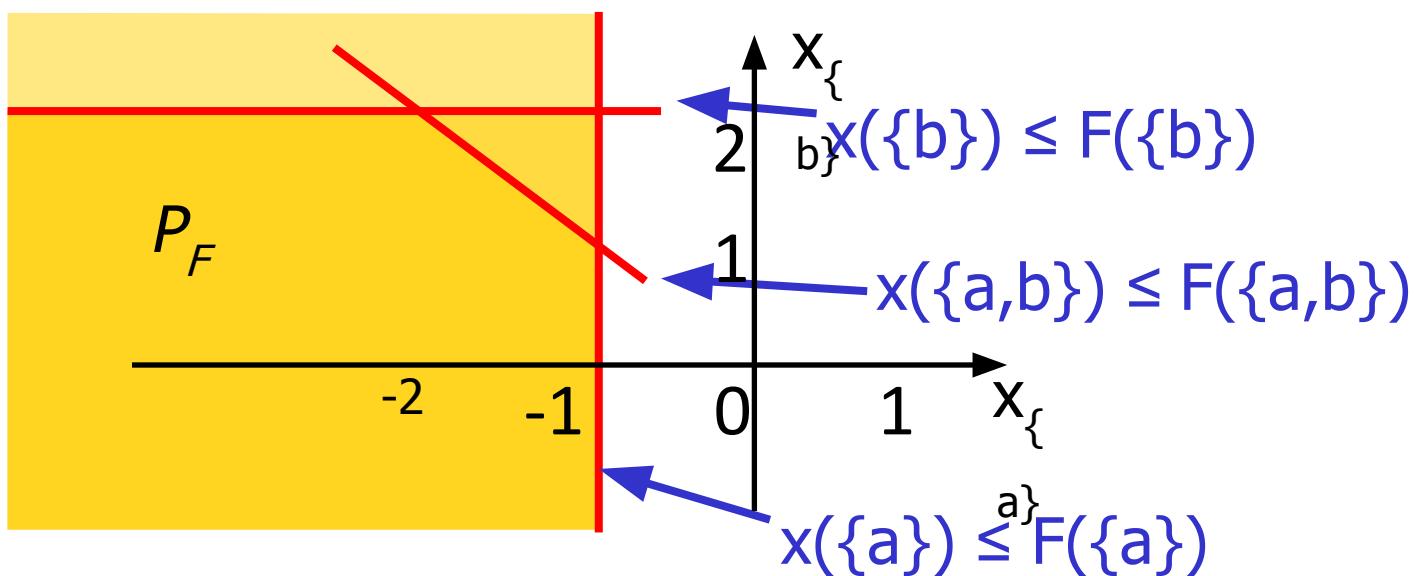


Different kinds of Continuous Extensions of Discrete Functions

# The Submodular Polyhedron

$$P_F = \{x \in \mathbb{R}^n : x(A) \leq F(A) \text{ for all } A \subseteq V\}$$

$$x(A) = \sum_{i \in A} x_i$$

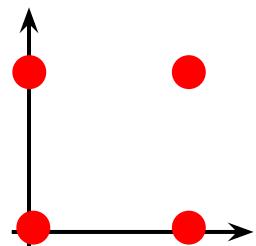


Example:  $V = \{a, b\}$

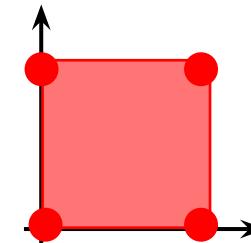
A	F(A)
$\{\}$	0
$\{a\}$	-1
$\{b\}$	2
$\{a, b\}$	0

# Convex Extensions of Submodular Functions

$$F : \{0, 1\}^n \rightarrow \mathbb{R} \quad \longrightarrow \quad f : [0, 1]^n \rightarrow \mathbb{R}$$



extension



Lovász extension

$$f(x) = \max_{y \in P_F} x \cdot y$$

convex

Lovász, 1982

- minimum of  $f$  is a minimum of  $F$
- **submodular minimization** as convex minimization:

**polynomial time!**

1981

Grötschel, Lovász, Schrijver

# Evaluating the Lovasz Extension

$$P_F = \{x \in \mathbb{R}^n : x(A) \leq F(A) \text{ for all } A \subseteq V\}$$

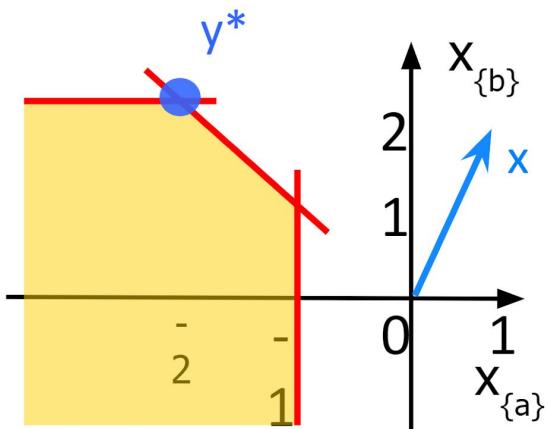
Linear maximization over  $P_F$

$$f(x) = \max_{y \in P_F} x \cdot y$$

Exponentially many constraints!!! 😞

Computable in  $O(n \log n)$  time ☺

[Edmonds '70]



greedy algorithm:

- sort  $x$
- order defines sets  $S_i = \{1, \dots, i\}$
- $y_i = F(S_i) - F(S_{i-1})$

The Lovasz Extension is a non-differentiable  
Convex Function!

# Properties of the Lovasz Extension

## Theorem 15.6.1

Let  $f, g : 2^E \rightarrow \mathbb{R}$  be normalized ( $f(\emptyset) = g(\emptyset) = 0$ ). Then

- ① *Superposition of LE operator:* Given  $f$  and  $g$  with Lovász extensions  $\tilde{f}$  and  $\tilde{g}$  then  $\tilde{f} + \tilde{g}$  is the Lovász extension of  $f + g$  and  $\lambda\tilde{f}$  is the Lovász extension of  $\lambda f$  for  $\lambda \in \mathbb{R}$ .
- ② If  $w \in \mathbb{R}_+^E$  then  $\tilde{f}(w) = \int_0^{+\infty} f(\{w \geq \alpha\})d\alpha$ .
- ③ For  $w \in \mathbb{R}^E$ , and  $\alpha \in \mathbb{R}$ , we have  $\tilde{f}(w + \alpha \mathbf{1}_E) = \tilde{f}(w) + \alpha f(E)$ .
- ④ *Positive homogeneity:* I.e.,  $\tilde{f}(\alpha w) = \alpha \tilde{f}(w)$  for  $\alpha \geq 0$ .
- ⑤ For all  $A \subseteq E$ ,  $\tilde{f}(\mathbf{1}_A) = f(A)$ .
- ⑥  $f$  symmetric as in  $f(A) = f(E \setminus A)$ ,  $\forall A$ , then  $\tilde{f}(w) = \tilde{f}(-w)$  ( $\tilde{f}$  is even).
- ⑦ Given partition  $E^1 \cup E^2 \cup \dots \cup E^k$  of  $E$  and  $w = \sum_{i=1}^k \gamma_i \mathbf{1}_{E_k}$  with  $\gamma_1 \geq \gamma_2 \geq \dots \geq \gamma_k$ , and with  $E^{1:i} = E^1 \cup E^2 \cup \dots \cup E^i$ , then  
$$\tilde{f}(w) = \sum_{i=1}^k \gamma_i f(E^i | E^{1:i-1}) = \sum_{i=1}^{k-1} f(E^{1:i})(\gamma_i - \gamma_{i+1}) + f(E)\gamma_k.$$

# Examples of Lovasz Extension

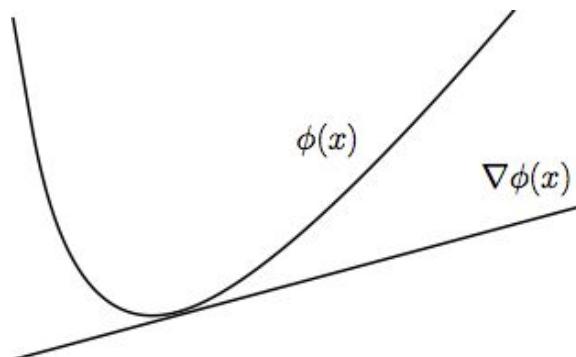
---

Some additional submodular functions and their Lovász extensions, where  $w(e_1) \geq w(e_2) \geq \dots \geq w(e_m) \geq 0$ . Let  $W_k \triangleq \sum_{i=1}^k w(e_i)$ .

$f(A)$	$\tilde{f}(w)$
$ A $	$\ w\ _1$
$\min( A , 1)$	$\ w\ _\infty$
$\min( A , 1) - \max( A  - m + 1, 0)$	$\ w\ _\infty - \min_i w_i$
$\min( A , k)$	$W_k$
$\min( A , k) - \max( A  - (n - k) + 1, 1)$	$2W_k - W_m$
$\min( A ,  E \setminus A )$	$2W_{\lfloor m/2 \rfloor} - W_m$

(thanks to K. Narayanan).

# Submodular Semi-gradients

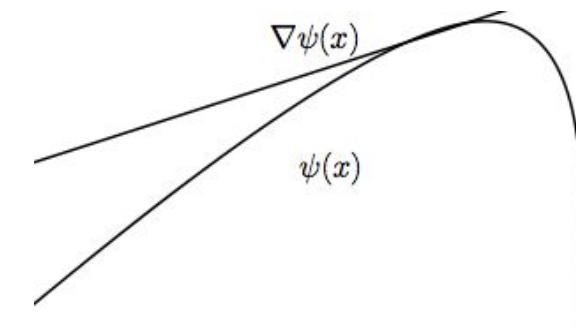


Convex Function

$$\phi(y) \geq \phi(x) + \langle \nabla\phi(x), y - x \rangle$$



Linear Lower Bound



Concave Function

$$\psi(y) \leq \psi(x) + \langle \nabla\psi(x), y - x \rangle$$



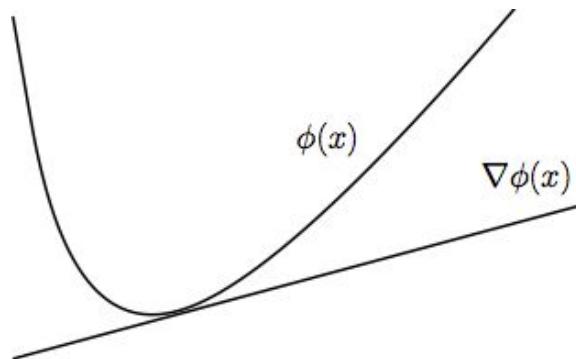
Linear Upper Bound

Can we do something similar for submodular functions?

$$f(Y) \geq f(X) + h_X(Y) - h_X(X)$$

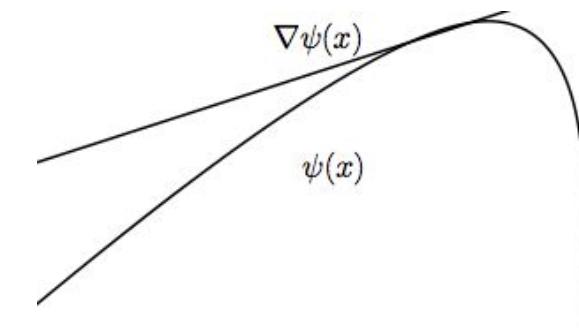
$$f(Y) \leq f(X) + g_X(Y) - g_X(X)$$

# Submodular Semi-gradients



Convex Function

$$\phi(y) \geq \phi(x) + \langle \nabla\phi(x), y - x \rangle$$



Concave Function

$$\psi(y) \leq \psi(x) + \langle \nabla\psi(x), y - x \rangle$$

We can define both Sub-gradients and Super-gradients for  
Submodular Functions!

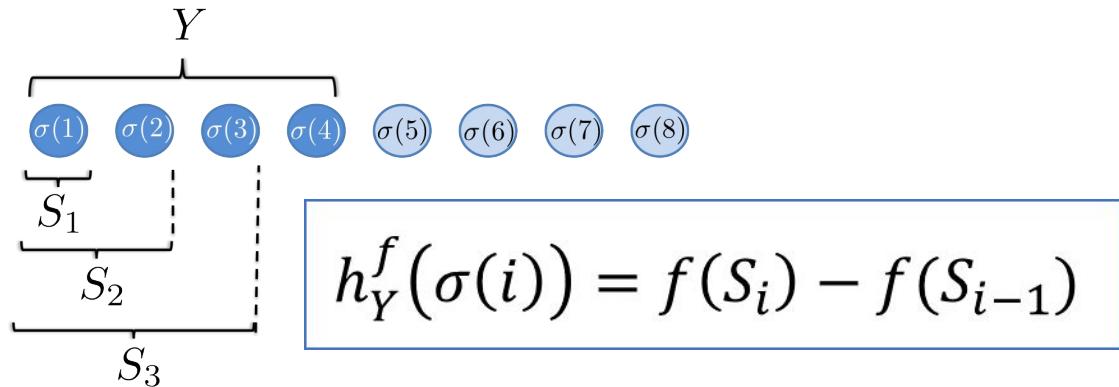
Can we do something similar for submodular functions?

$$f(Y) \geq f(X) + h_X(Y) - h_X(X)$$

$$f(Y) \leq f(X) + g_X(Y) - g_X(X)$$

# Submodular Semigradients

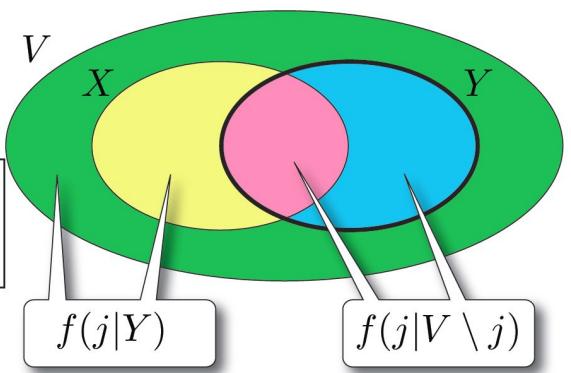
- Define a Sub-gradient  $h_Y^f$  at set  $Y$



- Define a Sup-gradient  $g_Y^f$  at set  $Y$

Grow:

$$\hat{g}_Y(j) = \begin{cases} f(j|Y) & \text{for } j \notin Y \\ f(j|V \setminus \{j\}) & \text{for } j \in Y \end{cases}$$



- Modular Lower bound:

$$m_Y(X) = f(Y) + h_Y^f(X) - h_Y^f(Y) \leq f(X)$$

(Fujishige 2005)

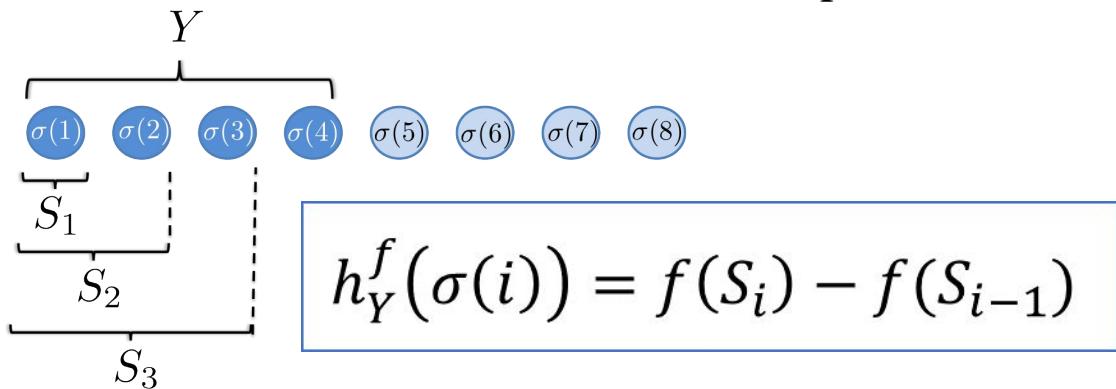
- Modular Upper bound:

$$m^Y(X) = f(Y) + g_Y^f(X) - g_Y^f(Y) \geq f(X)$$

(I-Jegelka-Bilmes ICML2013)

# Submodular Semigradients

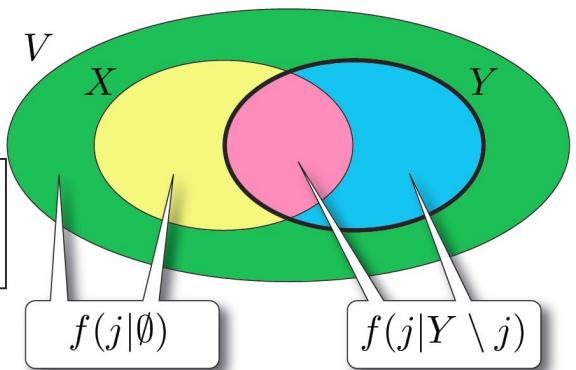
- Define a Sub-gradient  $h_Y^f$  at set  $Y$



- Define a Sup-gradient  $g_Y^f$  at set  $Y$

Shrink:

$$\check{g}_Y(j) = \begin{cases} f(j|\emptyset) & \text{for } j \notin Y \\ f(j|Y \setminus \{j\}) & \text{for } j \in Y \end{cases}$$



- Modular Lower bound:

$$m_Y(X) = f(Y) + h_Y^f(X) - h_Y^f(Y) \leq f(X)$$

(Fujishige 2005)

- Modular Upper bound:

$$m^Y(X) = f(Y) + g_Y^f(X) - g_Y^f(Y) \geq f(X)$$

(I-Jegelka-Bilmes ICML2013)

# Discrete vs Continuous Optimization

---

## Continuous Functions

Convex Functions: Improved  
Bounds

- Strong Convexity
- Lipschitz Continuous  
Gradients
- ...

## Discrete Functions

Submodular Functions:  
Improved Bounds

?

# Discrete vs Continuous Optimization

---

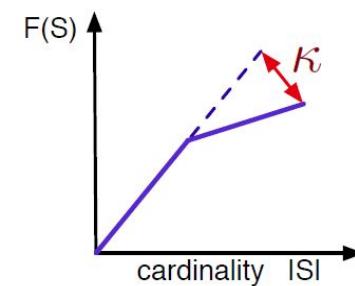
## Continuous Functions

Convex Functions: Improved  
Bounds

- Strong Convexity
- Lipschitz Continuous  
Gradients
- ...

## Discrete Functions

Submodular Functions:  
Improved Bounds



$$\kappa_f = 1 - \min_j \frac{f(j|V \setminus j)}{f(j)}$$

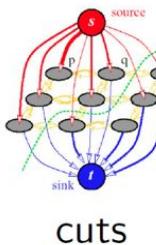
# Minimization vs Maximization

Cooperation/Complexity



Minimize  $f$

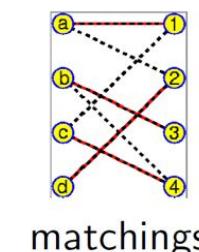
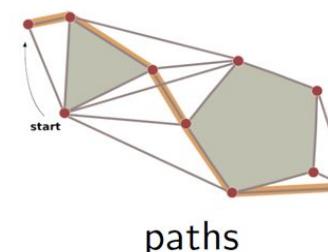
Often Combinatorial  
Constraints...



Coverage/ Diversity



Maximize  $g$



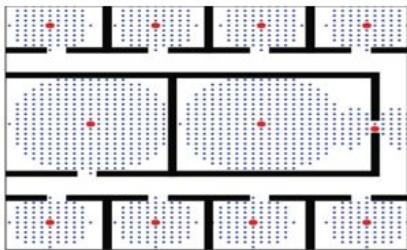
Simultaneously minimizing  $f$   
while maximizing  $g$ ?

Minimizing Cooperative Costs/Complexity while  
Simultaneously maximizing coverage/diversity!

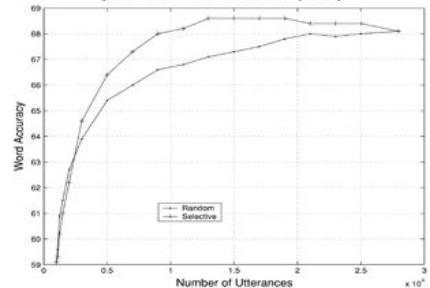
## Submodular Maximization



Data Summarization  
(Lin-Bilmes 11, Tschiatchek-Iyer et al 14, ...)



Sensor Placement  
(Krause et al 09, ...)



Data Subset Selection  
(Wei-Iyer-Bilmes 2015, ...)

## Submodular Minimization



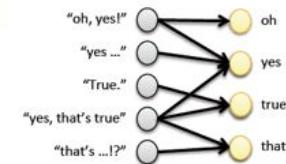
Cooperative Matching  
(Iyer-Bilmes 14)



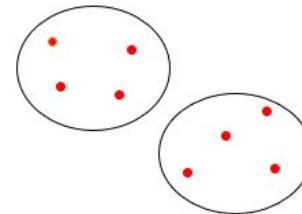
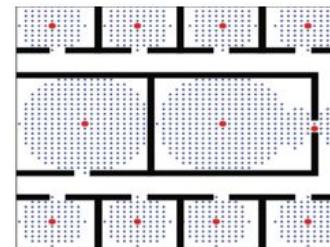
Cooperative Cuts  
(Jegelka-Bilmes 11, ...)

## Simultaneous Min & Max

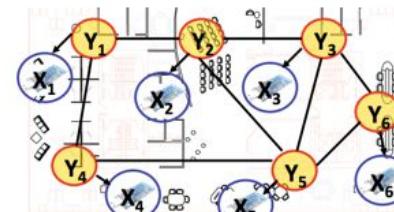
1 all\_right how\_are\_you doing  
2 how\_are\_you with yours  
3 hi nadine my name is lorraine how\_are\_you  
4 good how\_are\_you  
5 hello hi how\_are\_you  
6 good thanks how\_are\_you  
7 uh how\_are\_you  
8 i'm good how\_are\_you  
9 fine how\_are\_you



Limited Vocab. Speech Selection  
(Liu-Iyer-et al 15, ...)



Sensor Placement+ Cooperative costs  
(Iyer-Bilmes 14)



Feature Selection  
(Iyer-Bilmes 12, ...)

# Submodular Optimization Problems

$f \rightarrow$  cooperative cost or algorithmic complexity.

$g \rightarrow$  coverage/ diversity or information.

- Submodular Minimization:

$$\min_{X \in \mathcal{C}} f(X) \quad \longrightarrow$$

Minimizing  
Cooperative Costs

- Submodular Maximization:

$$\max_{X \in \mathcal{C}} g(X) \quad \longrightarrow$$

Maximizing  
Coverage/ Diversity

- Difference of Submodular (DS) optimization:

$$\min_X f(X) - \lambda g(X)$$

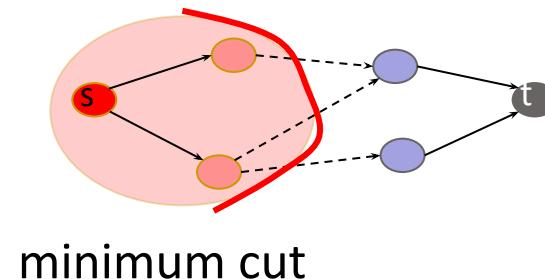
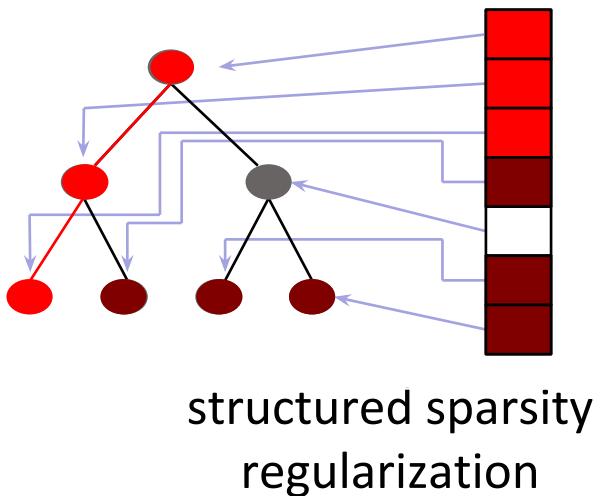
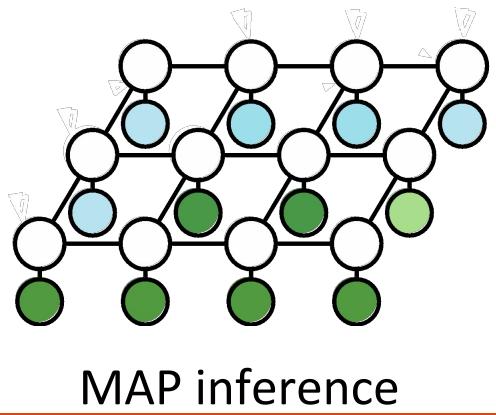
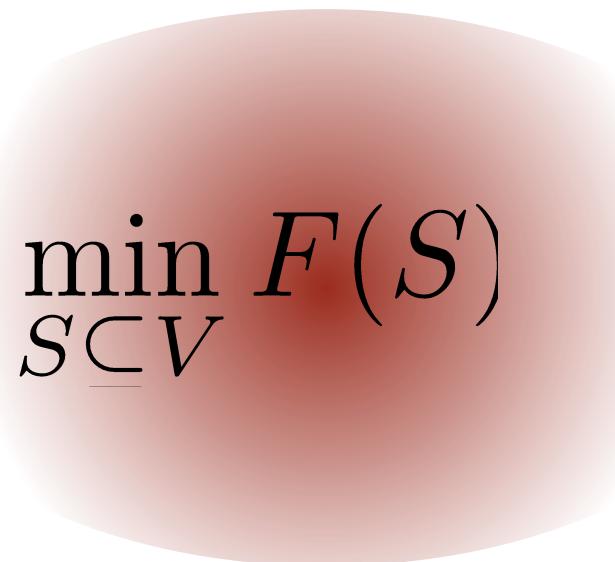
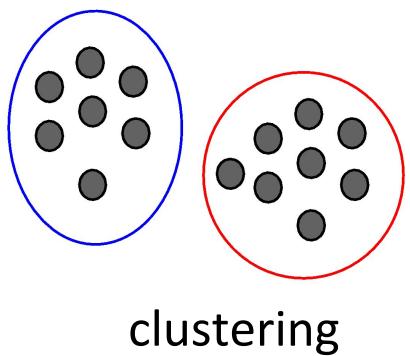
Minimizing  
cooperative  
costs and  
Maximizing  
Coverage/  
Diversity

- Submodular optimization under submodular constraints,

$$\min\{f(X) : g(X) \geq c\}, \quad \max\{g(X) : f(X) \leq b\}$$

Focus of this  
Lecture

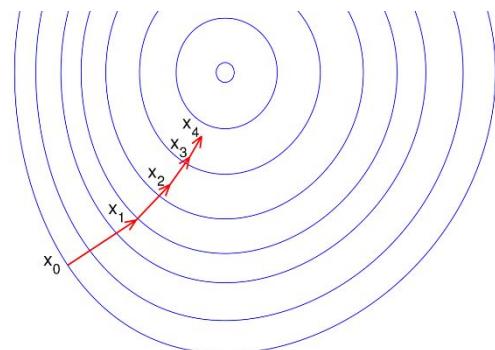
# Submodular Minimization Applications



# Discrete vs Continuous Optimization

---

Continuous Functions

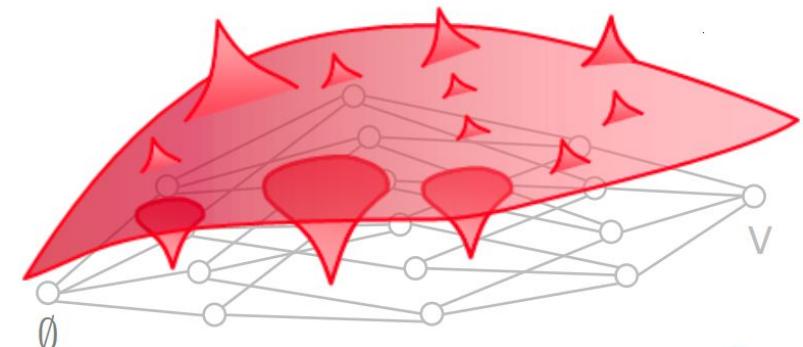


Convex Functions



Gradient Descent Framework

Discrete Functions



Submodular Functions



?

# Unconstrained Submodular Minimization Algorithms

---

minimize convex extension

ellipsoid algorithm

[Grötschel et al. '81]

subgradient method,  
smoothing [Stobbe & Krause '10]

duality: minimum norm point  
algorithm

[Fujishige & Isotani '11]

combinatorial algorithms

Fulkerson prize

Iwata, Fujishige, Fleischer '01 & Schrijver '00

state of the art:

$O(n^4T + n^5\log M)$  [Iwata '03]

$O(n^6 + n^5T)$  [Orlin '09]

$$\min_{A \subset V} F(A)$$

Unconstrained Submodular Minimization is Polynomial time Solvable!

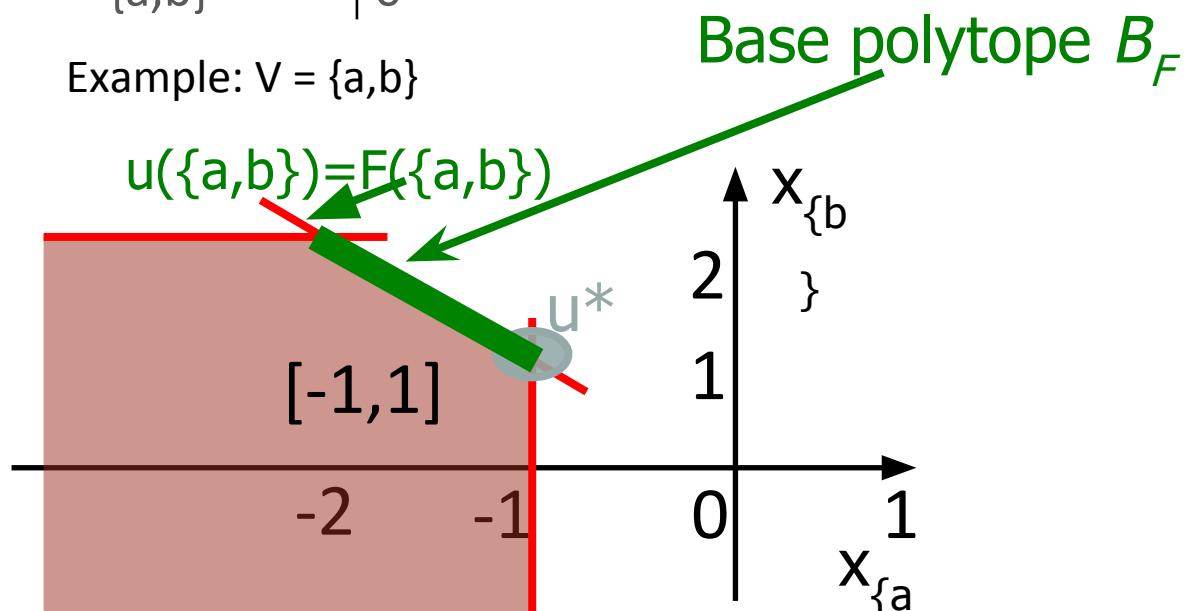
T = time for evaluating  $F$

# The minimum-norm-point algorithm

regularized problem

$$\min_{\substack{\{a\} \\ x \in [0,1]^n \\ \{b\}}} f_1(x) + \frac{1}{2} \|x\|^2$$

Example:  $V = \{a, b\}$



dual: minimum norm problem

$$u^* = \operatorname{argmin}_{u \in B_F} \frac{1}{2} \|u\|^2$$

$$A^* = \{i \mid u^*(i) \leq 0\}$$

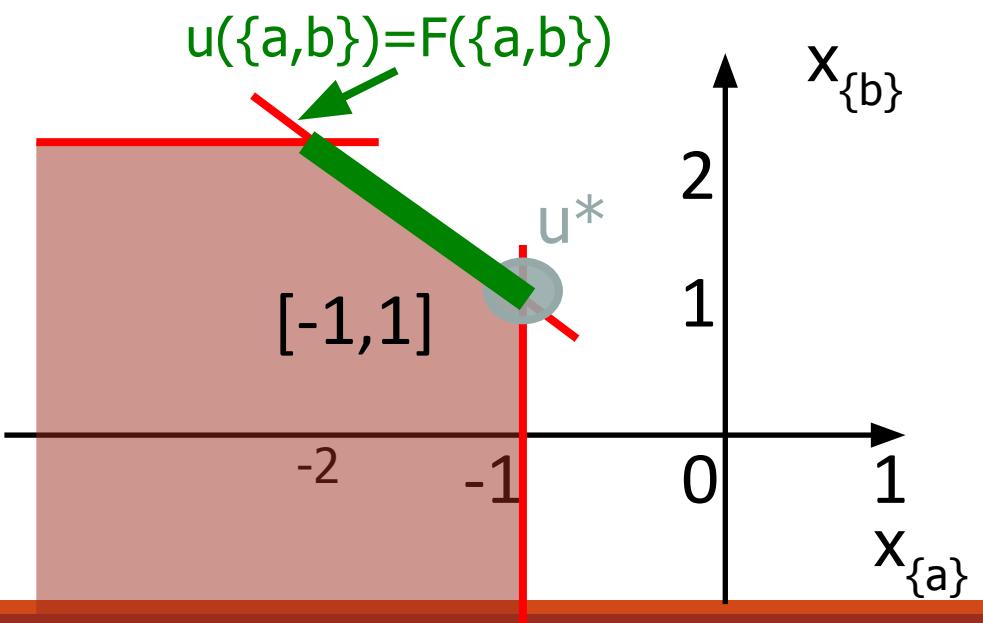
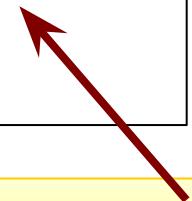
minimizes  $F$ :

$$A^* = \arg \min_{A \subset V} F(A)$$

Fujishige '91, Fujishige & Isotani '11

# The minimum-norm-point algorithm

1. find  $u^* = \arg \min_{u \in B_F} \frac{1}{2} \|u\|^2$
2.  $A^* = \{i \mid u^*(i) \leq 0\}$



yes! 😊

recall: can solve

linear optimization over  $P_F$

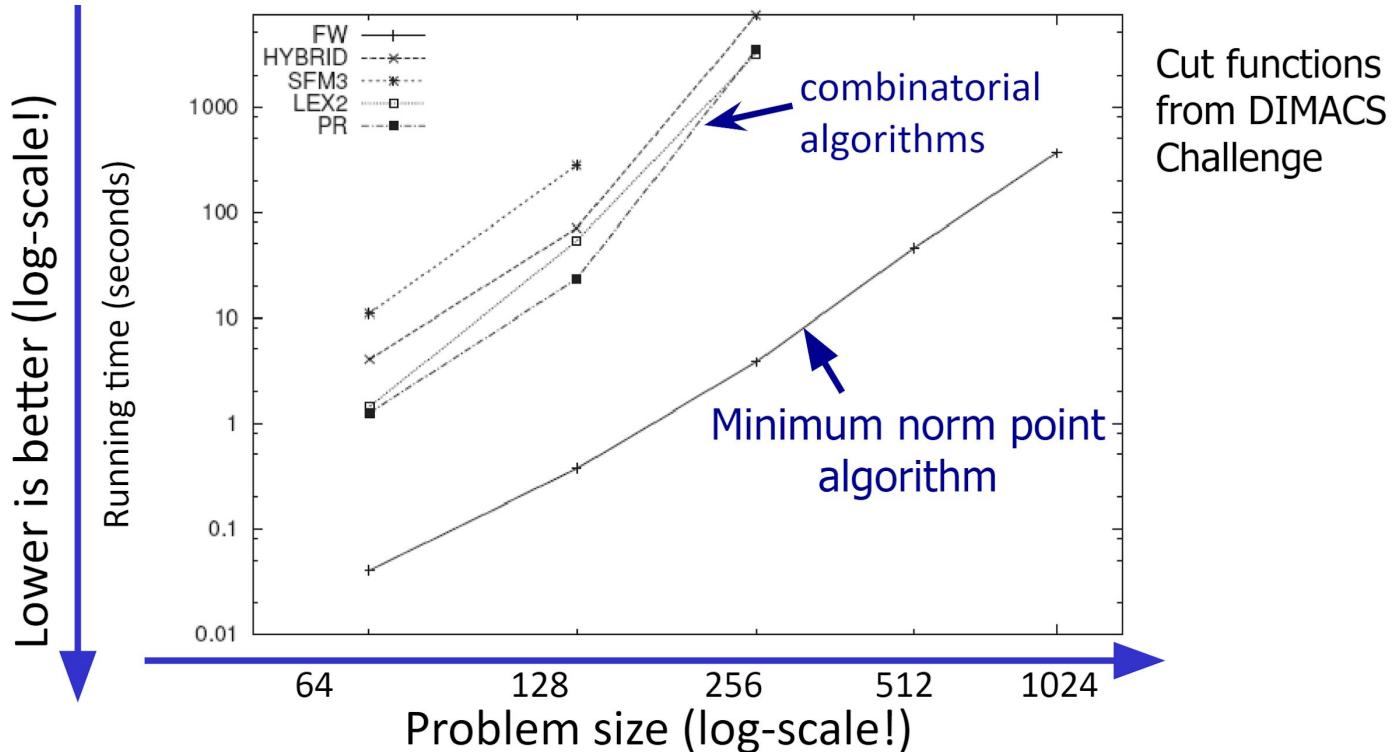
similar: optimization over  $B_F$

□ can find  $u^*$

(Frank-Wolfe algorithm)

can we solve this??

# Empirical Performance



Minimum norm point algorithm: usually orders of magnitude faster

[Fujishige & Isotani '11]

# Submodular Minimization with Constraints

unconstrained:

- nontrivial algorithms,  
**polynomial time**

$$\min F(A) \text{ s.t. } A \subseteq V$$

constraints: e.g.

- limited cases doable:  
odd/even cardinality, inclusion/exclusion of a set

$$\min F(A) \text{ s.t. } |A| \geq k$$

special case:  
balanced  
cut

...

General case: **NP hard**

- hard to approximate within polynomial factors!
- **But: special cases often still work well**

[Lower bounds: Goel et al.'09, Iwata & Nagano '09, Jegelka & Bilmes '11]

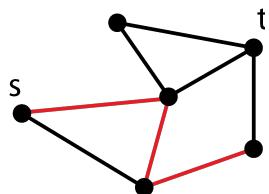
# Submodular Minimization with Constraints

---

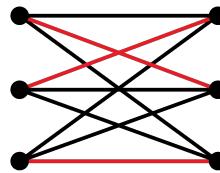
minimum

...

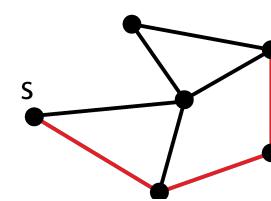
cut



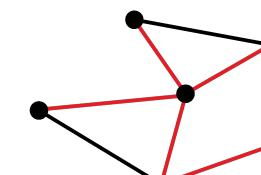
matching



path



spanning tree



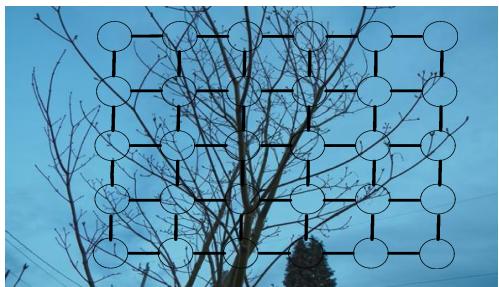
ground set: edges in a graph

$$\min_{S \in \mathcal{C}} \sum_{e \in S} w(e)$$

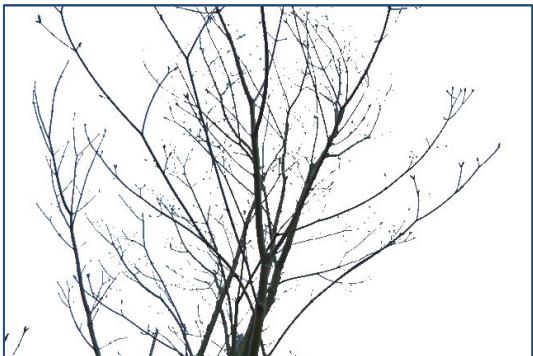


$$\min_{S \in \mathcal{C}} F(S)$$

# Motivation: Image Segmentation



aim  
reality

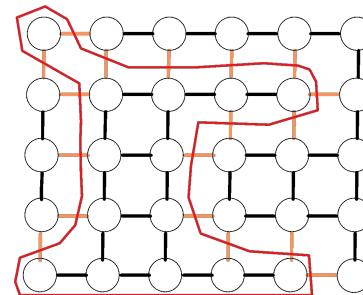


binary labeling:  $x = e_A$

pairwise random field:

$$E(x) = \text{Cut}(A)$$

What's the problem?



minimum cut: prefer  
short cut = short object  
boundary

# Co-operative Cuts

Minimum cut



implicit criterion:

short cut = short boundary

minimize  
sum of edge weights

$$F(C) = \sum_{e \in C} w(e)$$

not a sum of  
edge weights!

Minimum cooperative cut

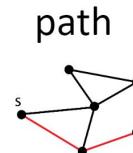
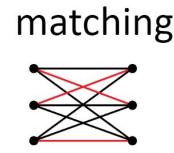
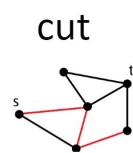


new criterion:

boundary may be long if the boundary is  
homogeneous

minimize  
**submodular** function of edges  $F(C)$

# Constrained Submodular Min: Algorithms



$$\min_{S \in \mathcal{C}} F(S)$$

approximate optimization

convex relaxation

minimize surrogate function

approximation bounds dependent on  $F$ :

polynomial – constant – FPTAS

$O(n)$   $(1 + \epsilon)$

[Goel et al.'09, Iwata & Nagano '09, Goemans et al. '09, Jegelka & Bilmes '11, Iyer et al. ICML '13, Kohli et al '13...]

# Approximation Algorithms

---

- ❑ Given a min(max)-imization problem, obtaining the exact min(max)-imizer implies an approximation factor of 1.
- ❑ Since many problems are NP hard, it is not possible to achieve the exact optima in poly-time
- ❑ Define  $\alpha$  as the approximation factor.
- ❑ A set  $X$  is the  $\alpha$ -approximate minimizer if  $f(X) \leq \alpha f(X^*)$  where  $X^*$  is the global minima ( $\alpha > 1$ )
- ❑ A set  $X$  is the  $\alpha$ -approximate maximizer if  $f(X) \geq \alpha f(X^*)$  where  $X^*$  is the global maxima ( $\alpha < 1$ )

# Constrained Submodular Min Framework

minimize a series of surrogate functions

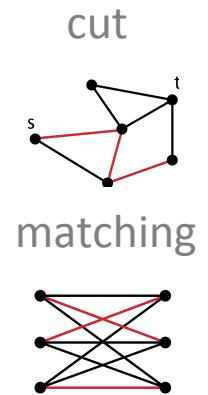
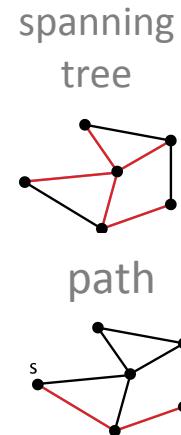
1. compute linear upper bound  $\hat{F}^i(S^i) = F(S^i)$

$$\hat{F}^i(S) = \sum_{e \in S} w^i(e)$$

2. Solve **easy sum-of-weights problem:**

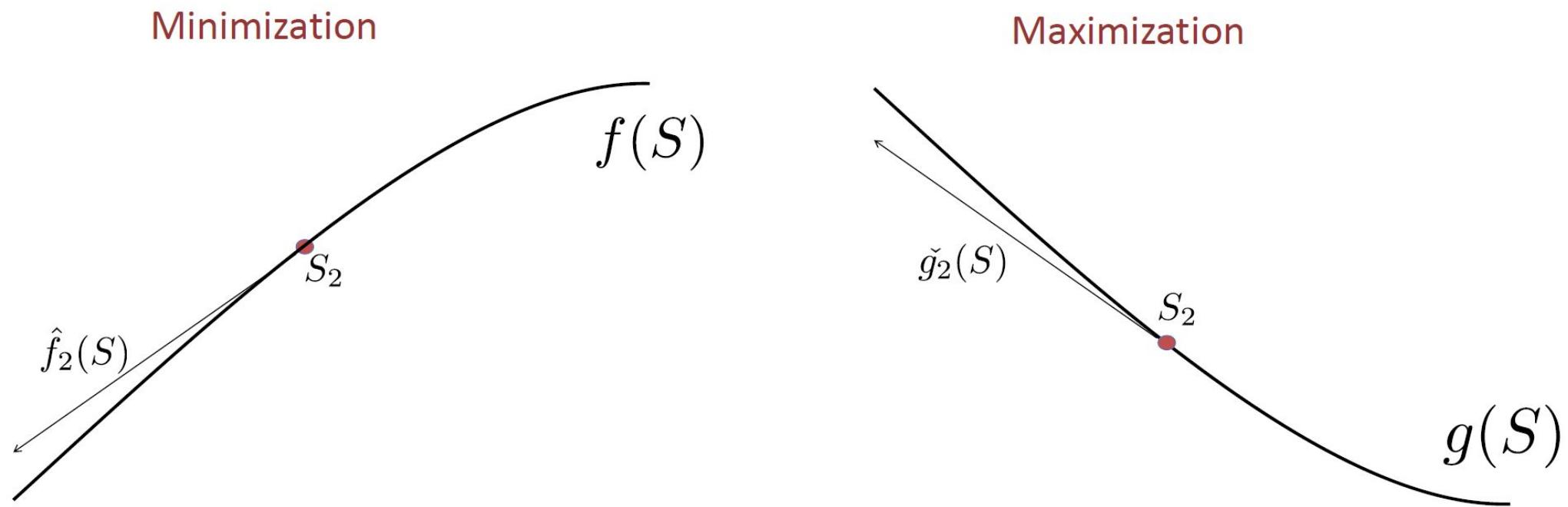
$$S^i = \arg \min_{S \in \mathcal{C}} \hat{F}^i(S) \quad \text{and repeat.}$$

- efficient
- only need to solve sum-of-weights problems



# Majorization-Minimization Framework

For Min (Max) problems use the upper (lower) bounds as proxy functions!



Always Improves the objective value!

# Constrained Submodular Minimization

---

minimize  $F(S) : S \in \mathcal{C}$  = cut/path/matching/cardinality constraint...

```
Initialize  $S_0 = \emptyset$ ;  
for  $i = 0, 1, \dots$  do
```

compute modular upper bound  $\hat{f}_i(S) = m_{S_i}^f(S) \geq f(S)$  based on  $S_i$ ;

Set  $S_{i+1} = \operatorname{argmin}_{S \in \mathcal{C}} \hat{f}_i(S)$  - find best cut/path/matching...;  
only need to solve linear-cost problem! ,

```
end
```

Iyer-Jegelka-Bilmes ICML 2013 (**Award Paper**)

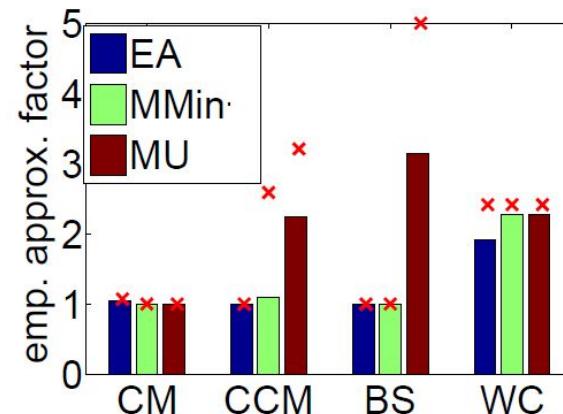
# How does MMin perform?

minimize  $F(S) : S \in \mathcal{C}$  = cut/path/matching/cardinality constraint...

## In Theory...

- Mmin (IJB13):  $\frac{n}{1+(n-1)(1-\kappa)} \approx \frac{1}{1-\kappa}$
- EA\* (Goemans et al):  $O\left(\frac{\sqrt{n}}{1+(\sqrt{n}-1)(1-\kappa)}\right) \approx \frac{1}{1-\kappa}$
- Mmin for subclasses (SCM):  $1 + \epsilon$

## In Practice...



MMin 100x Faster compared to EA!

\* Theoretically Tight

# How does MMin perform?

---

minimize  $F(S) : S \in \mathcal{C}$  = cut/path/matching/cardinality constraint...

For graph based problems,  $m$  = number of edges,  $n$  = number of vertices.

How good are these algorithms?  $f(S) \leq \alpha f(S^*)$

Constraint:	MMin	EA	Lower bound
trees/matchings	$\frac{n}{1+(n-1)(1-\kappa_f)}$	$O\left(\frac{\sqrt{m}}{1+\sqrt{m}-1}(1-\kappa_f)\right)$	$\Omega\left(\frac{n}{1+(n-1)(1-\kappa_f)}\right)$
cuts	$\frac{m}{1+(m-1)(1-\kappa_f)}$	$O\left(\frac{\sqrt{m}}{1+\sqrt{m}-1}(1-\kappa_f)\right)$	$\Omega\left(\frac{\sqrt{m}}{1+\sqrt{m}-1}(1-\kappa_f)\right)$
paths	$\frac{n}{1+(n-1)(1-\kappa_f)}$	$O\left(\frac{\sqrt{m}}{1+\sqrt{m}-1}(1-\kappa_f)\right)$	$\Omega\left(\frac{n^{2/3}}{1+(n^{2/3}-1)(1-\kappa_f)}\right)$
cardinality( $k$ )	$\frac{k}{1+(k-1)(1-\kappa_f)}$	$O\left(\frac{\sqrt{n}}{1+\sqrt{n}-1}(1-\kappa_f)\right)$	$\Omega\left(\frac{\sqrt{n}}{1+\sqrt{n}-1}(1-\kappa_f)\right)$

Worst case upper/lower bounds bounded by  $O\left(\frac{1}{(1-\kappa_f)}\right)$

# Submodular Maximization

Set Function  $\max_{\mathcal{A} \subseteq \mathcal{V}} F(\mathcal{A})$  Selected set

Selection cost subject to  $C(\mathcal{A}) \leq B$  Budget

$F = \text{Monotone Submodular,}$   
 $\text{Non Monotone Submodular,}$   
 $\text{Dispersion Functions,}$   
....

- F Models:
- Diversity
  - Representation
  - Coverage
  - Information
  - Importance
  - ...

We shall study this and variants of this Master Optimization Problem!

# Maximizing submodular functions

---

$$A^* = \arg \max_A F(A) \text{ s.t. } A \subseteq V$$

Suppose we want for **submodular**  $F$

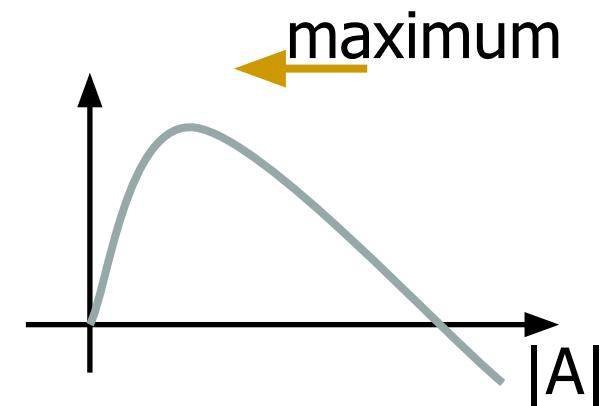
Example:

- $F(A) = U(A) - C(A)$  where  $U(A)$  is submodular utility, and  $C(A)$  is supermodular cost function

In general: NP hard. Moreover:

If  $F(A)$  can take negative values:

As hard to approximate as maximum independent set  
(i.e., NP hard to get  $O(n^{1-\varepsilon})$  approximation)



# Exact maximization of SFs

---

Mixed integer programming

- Series of mixed integer programs [Nemhauser et al ‘81]
- Constraint generation [Kawahara et al ‘09]

Branch-and-bound

- „Data-Correcting Algorithm“ [Goldengorin et al ‘99]

Useful for small/moderate problems

All algorithms worst-case exponential!

# Monotone Submodular Maximization

$$\max_S F(S) \text{ s.t. } |S| \leq k$$

What is the Constraint?  
 $C(S) = |S|$

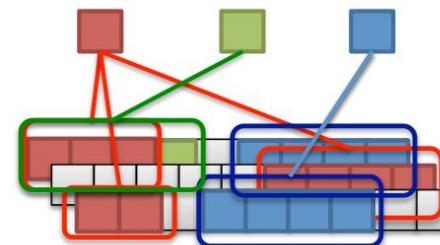
- greedy algorithm:

$$S_0 = \emptyset$$

for  $i = 0, \dots, k-1$

$$e^* = \arg \max_{e \in \mathcal{V} \setminus S_i} F(S_i \cup \{e\})$$

$$S_{i+1} = S_i \cup \{e^*\}$$

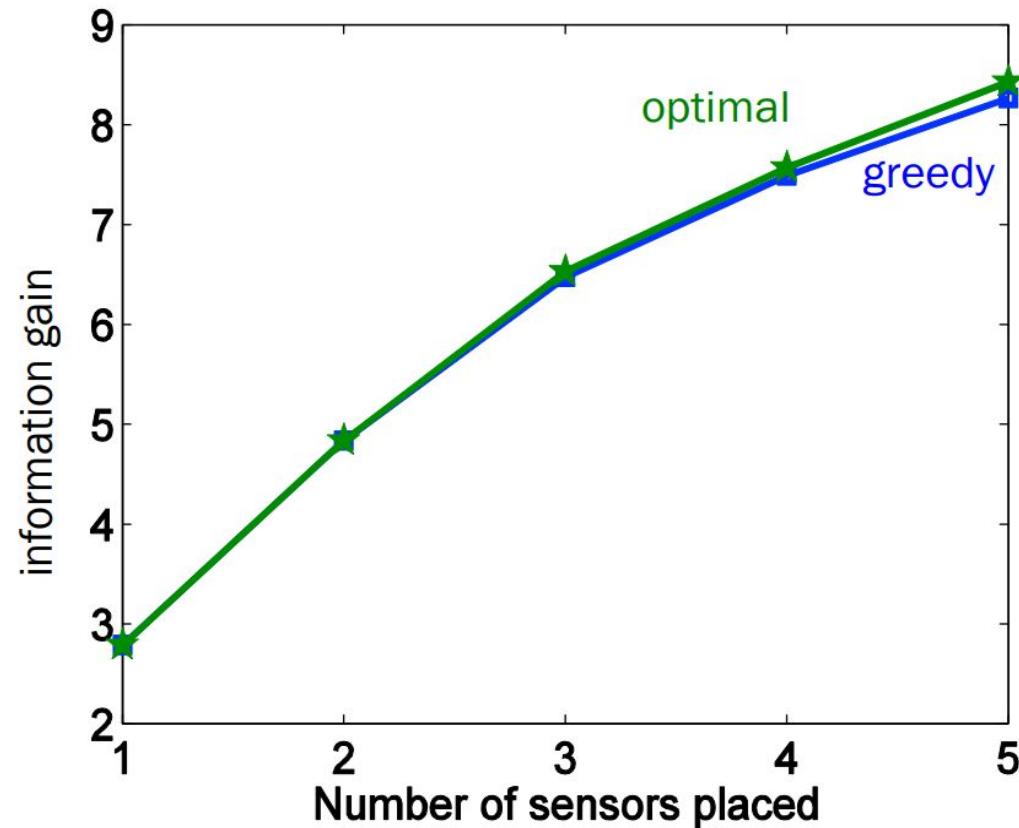


How “good” is  $S_k$  ?

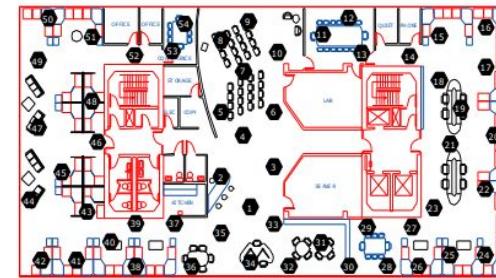
Approximation  
Guarantee!

# How good is Greedy in Practice?

empirically:



sensor placement



# How good is Greedy in Theory?

$$\max_S F(S) \text{ s.t. } |S| \leq k$$

**Theorem (Nemhauser, Fisher, Wolsey '78)**

$F$  monotone submodular,  $S_k$  solution of greedy. Then

$$F(S_k) \geq \left(1 - \frac{1}{e}\right) F(S^*)$$

optimal solution

No Poly-time algorithm can do better than this in the worst case!

# Monotone Submodular – Budget Constraints

$$\max F(S) \text{ s.t. } \sum_{e \in S} c(e) \leq B$$

1. run greedy:  $S_{\text{gr}}$
2. run a modified greedy:  $S_{\text{mod}}$

$$e^* = \arg \max \frac{F(S_i \cup \{e\}) - F(S_i)}{c(e)}$$

3. pick better of  $S_{\text{gr}}, S_{\text{mod}}$

→ approximation factor:

$$\frac{1}{2} \left(1 - \frac{1}{e}\right)$$

even better but less fast:  
partial enumeration  
(Sviridenko, 2004) or  
filtering (Badanidiyuru &  
Vondrák 2014)

**Sviridenko 2004:**

- Run the cost-sensitive greedy algorithm starting with all possible initial sets {I,j,k}
- $O(n^3)$  initial complexity
- $1 - 1/e$  approximation!

Sviridenko 2004, Leskovec et al 2007

# Summary: Greedy Algorithm Framework

Monotone Submodular Function

$$\max_{S \subseteq V, c(S) \leq \mathcal{B}} f(S)$$

Cost of Summary Subset S (e.g. size)

**Problem Formulation**

Initialization  $S \leftarrow \emptyset$ .

**repeat**

Pick an element  $v^* \in \operatorname{argmax}_{v \in V \setminus S} \frac{f(v \cup S) - f(S)}{c(v)}$

Update  $S \leftarrow S \cup v^*$

**until** Reaching the budget, i.e.,  $c(S) > \mathcal{B}$

**Greedy Algorithm**

# Dual Problem: Monotone Submodular Cover

---

$$A^* = \operatorname{argmin} |A| \text{ s.t. } F(A) \geq Q$$

# Dual Problem: Monotone Submodular Cover

---

$$A^* = \operatorname{argmin} |A| \text{ s.t. } F(A) \geq Q$$

Greedy algorithm:

Start with  $A := \emptyset$ ;

While  $F(A) < Q$  and  $|A| < n$

$s^* := \operatorname{argmax}_s F(A \cup \{s\})$

$A := A \cup \{s^*\}$

# Dual Problem: Monotone Submodular Cover

---

$$A^* = \operatorname{argmin} |A| \text{ s.t. } F(A) \geq Q$$

Greedy algorithm:

Start with  $A := \emptyset$ ;

While  $F(A) < Q$  and  $|A| < n$

$s^* := \operatorname{argmax}_s F(A \cup \{s\})$

$A := A \cup \{s^*\}$

For bound, assume  
 $F$  is integral.  
If not, just round it.

**Theorem [Wolsey et al]:** Greedy will return  $A_{\text{greedy}}$   
 $|A_{\text{greedy}}| \leq (1 + \log \max_s F(\{s\})) |A_{\text{opt}}|$

# More Complex Constraints: Matroid Constraints

- Ground set:  $V = \{1_a, 1_b, \dots, 5_a, 5_b\}$
- Sensing quality model:  $F : 2^V \rightarrow \mathbb{R}$

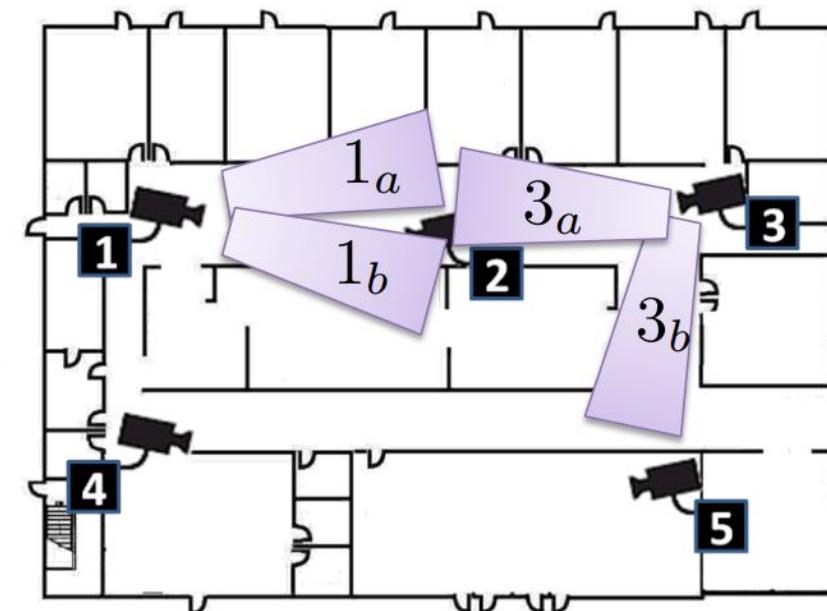
- Configuration (subset) is feasible if no camera is pointed in two directions at once

- Constraints:

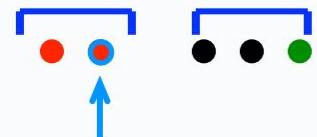
$$P_1 = \{1_a, 1_b\}, \dots, P_5 = \{5_a, 5_b\}$$

require:

$$|S \cap P_i| \leq 1$$

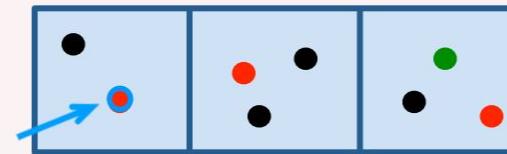


# Matroid Constraints?



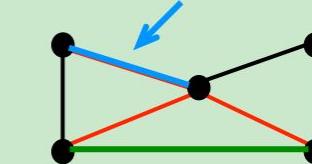
...  $|S| \leq k$

Uniform matroid



...  $S$  contains at most  
one element from  
each group

Partition matroid

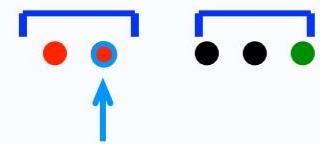


...  $S$  contains no  
cycles

Graphic matroid

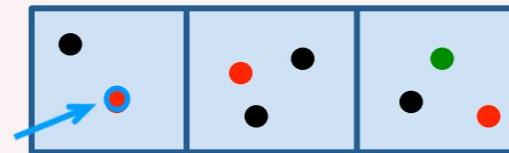
- $S$  independent  $\rightarrow T \subseteq S$  also independent

# Matroid Constraints?



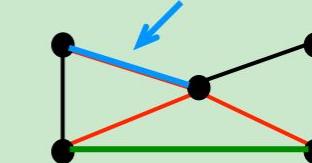
...  $|S| \leq k$

Uniform matroid



...  $S$  contains at most  
one element from  
each group

Partition matroid



...  $S$  contains no  
cycles

Graphic matroid

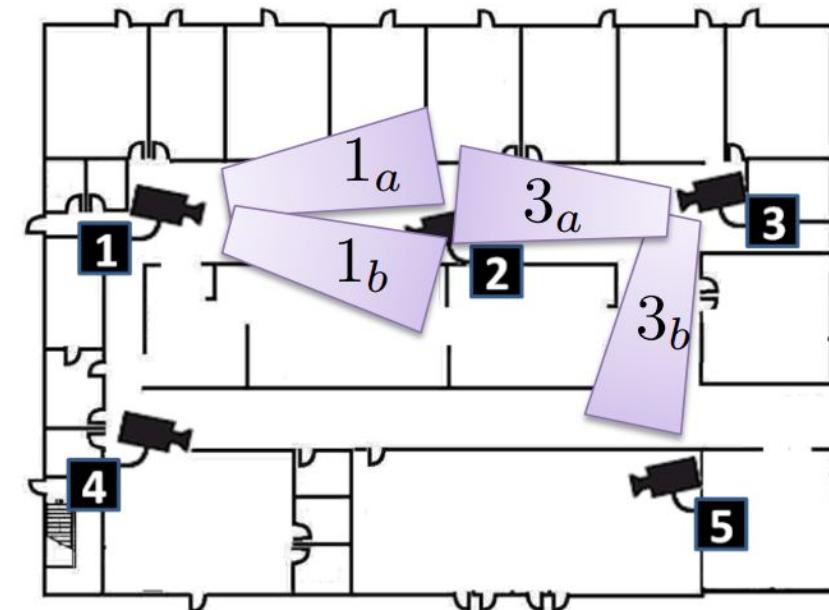
- $S$  independent  $\rightarrow T \subseteq S$  also independent
- Exchange property:  $S, U$  independent,  $|S| > |U|$   
 $\rightarrow$  some  $e \in S$  can be added to  $U$ :  $U \cup e$  independent
- All maximal independent sets have the same size

# Monotone Submodular Maximization subject to Matroid Constraints

$$S = \emptyset$$

**While**  $\exists e : S \cup e$  feasible

$e^* \leftarrow \operatorname{argmax}\{F(S \cup e) \mid S \cup e \text{ feasible}\}$

$$S \leftarrow S \cup e^*$$


# Monotone Submodular Maximization subject to Matroid Constraints

$$S = \emptyset$$

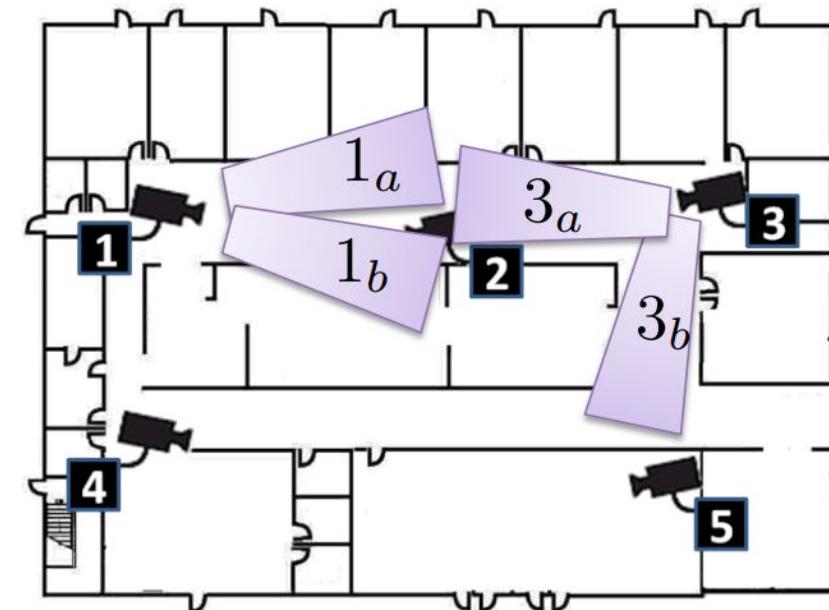
**While**  $\exists e : S \cup e$  feasible

$$\begin{aligned} e^* &\leftarrow \operatorname{argmax}\{F(S \cup e) \mid S \cup e \text{ feasible}\} \\ S &\leftarrow S \cup e^* \end{aligned}$$

**Theorem** (Nemhauser, Wolsey, Fisher 78)

For monotone submodular functions:

$$F(S_{\text{greedy}}) \geq \frac{1}{2}F(S^*)$$



# Non-Monotone Submodular Functions

Monotone Submodular Function

$$\max_{S \subseteq V, c(S) \leq \mathcal{B}} f(S)$$

Cost of Summary Subset S (e.g. size)

Start with  $Y_0 = \emptyset$

**for**  $i = 1$  **to**  $k$  **do**

    Let  $M_i = \operatorname{argmax}_{X \subseteq V \setminus Y_{i-1}, |X|=k} \sum_{v \in X} f(v|Y_{i-1})$ ;

    Choose  $y$  as a uniformly random element in  $M_i$ ;

$Y_i = Y_{i-1} \cup y$ ;

**return**  $Y_k$ .

Theorem (Buchbinder et al 2014): The Randomized Greedy Algorithm achieves a  $1/e$  approximation guarantee for Non-Monotone Submodular Maximization subject to cardinality constraints!

# Budget into the Objective: Unconstrained Submodular Max

Submodular Function



$$\max_{X \subseteq V} f(X) - \lambda c(X)$$



Cost of Summary Subset S (e.g. size)

Unconstrained Non-Monotone  
Submodular Maximization (USM)

Start:  $A = \emptyset, B = \mathcal{V}$

for  $i=1, \dots, n$  *//add or remove?*

add with probability

$$\mathbb{P}(\text{add}) = \frac{\Delta_+}{\Delta_+ + \Delta_-}$$

**add to A** or **remove from B**

Greedy can be arbitrarily bad!

Theorem (Buchbinder et al 2012): Bidirectional Greedy obtains a  $\frac{1}{2}$  approximation for USM!

# Submodular + Dispersion Functions

Submodular + Dispersion

$$\max_{S \subseteq V, c(S) \leq \mathcal{B}} f(S)$$

Cost of Summary Subset S (e.g. size)

**Problem Formulation**

Initialization  $S \leftarrow \emptyset$ .

**repeat**

Pick an element  $v^* \in \operatorname{argmax}_{v \in V \setminus S} \frac{f(v \cup S) - f(S)}{c(v)}$

Update  $S \leftarrow S \cup v^*$

**until** Reaching the budget, i.e.,  $c(S) > \mathcal{B}$

**Greedy Algorithm**

Theorem (Dasgupta et al 2013): The greedy algorithm achieves a  $\frac{1}{2}$  and  $\frac{1}{4}$  approximation

For the Submodular + Dispersion Sum and Submodular + Dispersion Min Optimization Problems

Subject to cardinality constraints

# Minorization-Maximization for Submodular Maximization

---

Submodular Maximization:  $\max_{X \in \mathcal{C}} g(X)$

```
Initialize  $S_0 = \emptyset$ ;  
for  $i = 0, 1, \dots$  do
```

compute modular lower bound  $\check{g}_i = h_{S_i}^g \leq g$  based on  $S_i$ ;

Set  $S_{i+1} = \operatorname{argmax}_{S \in \mathcal{C}} \check{g}_i(S)$ ;  
only need to solve linear-cost problem! ,

```
end
```

# Theoretical Results?

Submodular Maximization:  $\max_{X \in \mathcal{C}} g(X)$

Different Orderings => Known approximation  
Algorithms => Approximation bounds

## Unconstrained Maximization

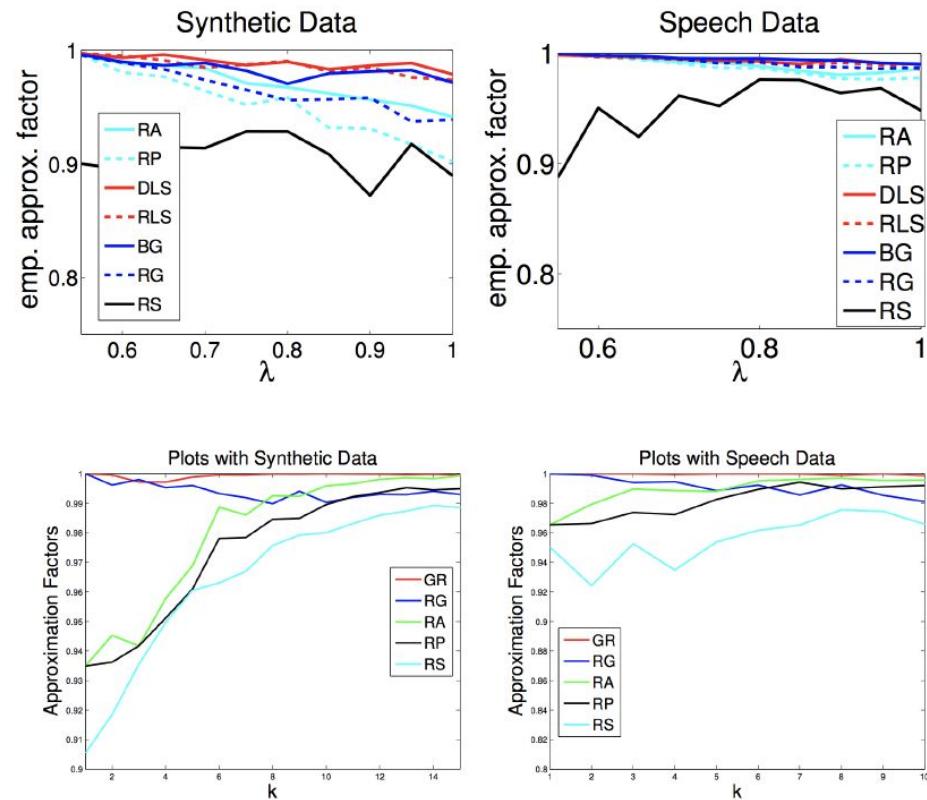
- Random Ordering (1/4)
- Local Search (1/3)
- Bidirectional Greedy (1/3)
- Randomized Greedy (1/2)
- ...

## Constrained Maximization

- Random Ordering ( $k/n$ )
- Greedy ( $1 - 1/e$ )
- Randomized Greedy ( $1/e$ )
- Lazier than Lazy Greedy ( $1 - 1/e$ )
- ...

# Example 2: Empirically Performance

---



## Observations:

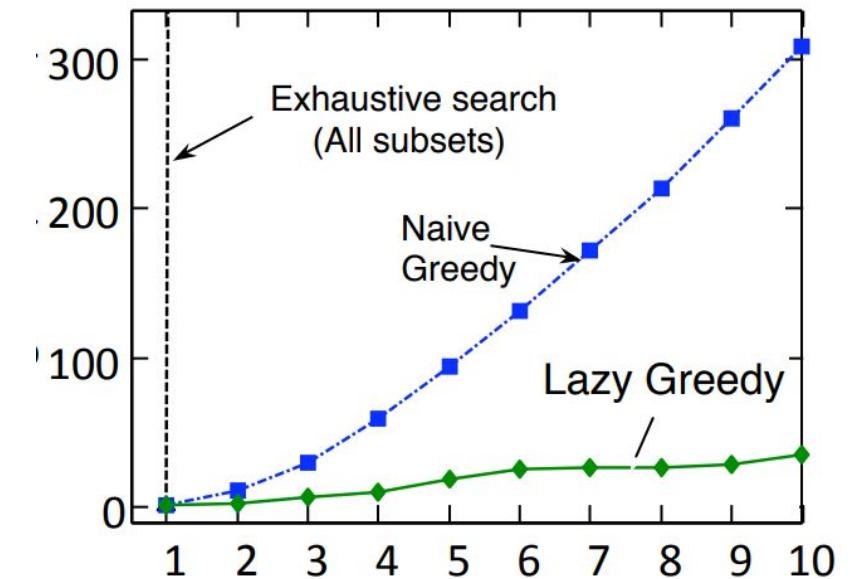
- Greedy Algorithms often perform best in practice!
- Theoretically tightest algorithms do not often work the best!

# Implementational Tricks: Lazy greedy

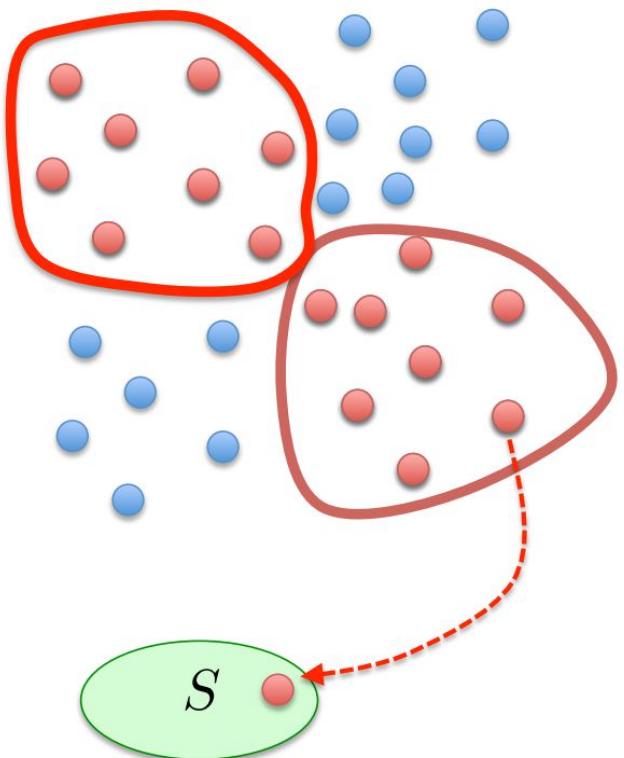
- ❑ (Minoux 1982) that the greedy algorithm can be efficiently implemented via a lazy greedy algorithm.
- ❑ Idea is to accelerate the algorithm via priority queues and maintains upper bounds on the marginal gains
- ❑ In practice this brings the complexity to linear time

Complexity still  $O(nkF)$  in worst case, where  
 $F$  = Function Eval Complexity.

Can we do better?



# Making Lazy Greedy Faster: Stochastic Greedy



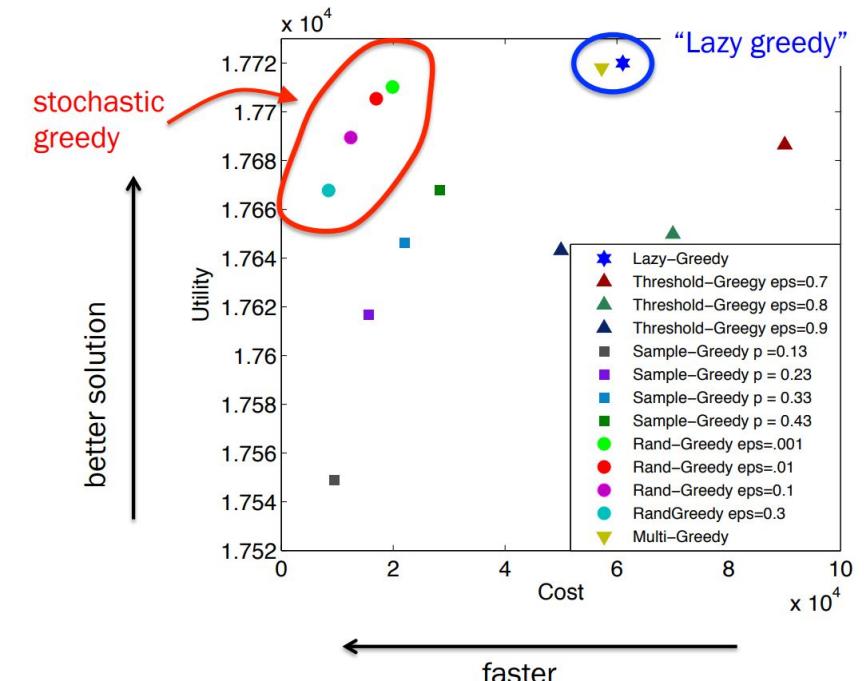
$$\max_S F(S) \text{ s.t. } |S| \leq k$$

for  $i=1\dots k$ :

- randomly pick set  $T$  of size  $\frac{n}{k} \log \frac{1}{\epsilon}$
- find best  $a$  element in  $T$  and add

$$a_i = \arg \max_{a \in T} F(a | S_{i-1})$$

$$S_i \leftarrow S_{i-1} \cup \{a_i\}$$



Mirzasoleiman et al 2014,

# Streaming Submodular Maximization

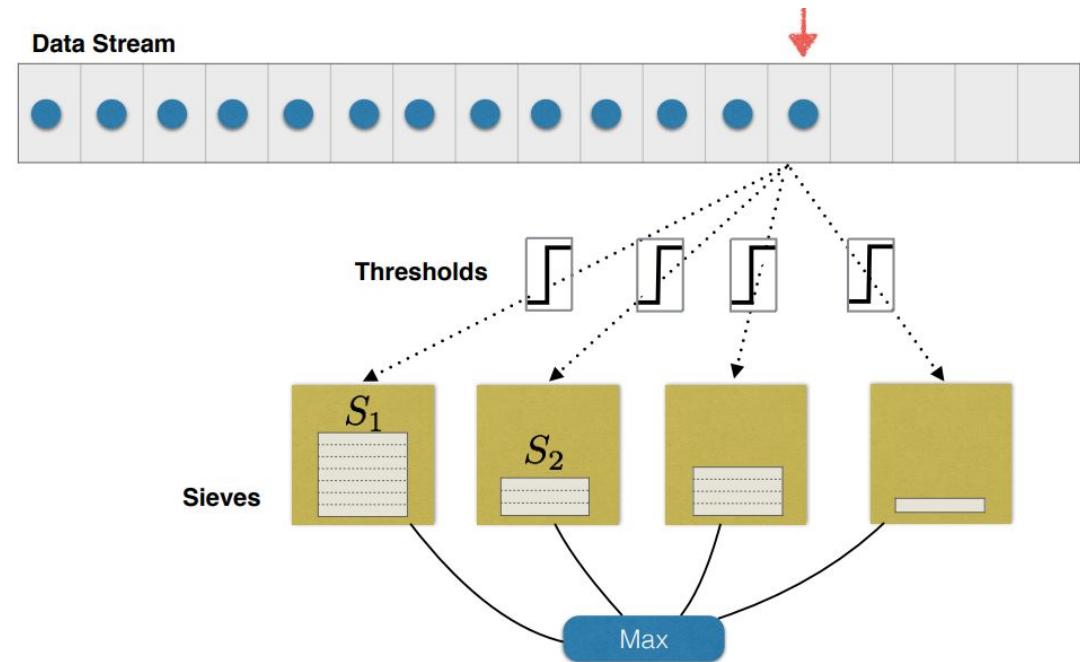
---

## Algorithm SIEVE-STREAMING

---

```
1:  $O = \{(1 + \epsilon)^i | i \in \mathbb{Z}\}$ 
2: For each  $v \in O$ ,  $S_v := \emptyset$  (maintain the sets only for the necessary  $v$ 's lazily)
3:  $m := 0$ 
4: for  $i = 1$  to  $n$  do
5:    $m := \max(m, f(\{e_i\}))$ 
6:    $O_i = \{(1 + \epsilon)^i | m \leq (1 + \epsilon)^i \leq 2 \cdot k \cdot m\}$ 
7:   Delete all  $S_v$  such that  $v \notin O_i$ .
8:   for  $v \in O_i$  do
9:     if  $\Delta_f(e_i | S_v) \geq \frac{v/2 - f(S_v)}{k - |S_v|}$  and  $|S_v| < k$  then
10:       $S_v := S_v \cup \{e_i\}$ 
11: return  $\operatorname{argmax}_{v \in O_n} f(S_v)$ 
```

---

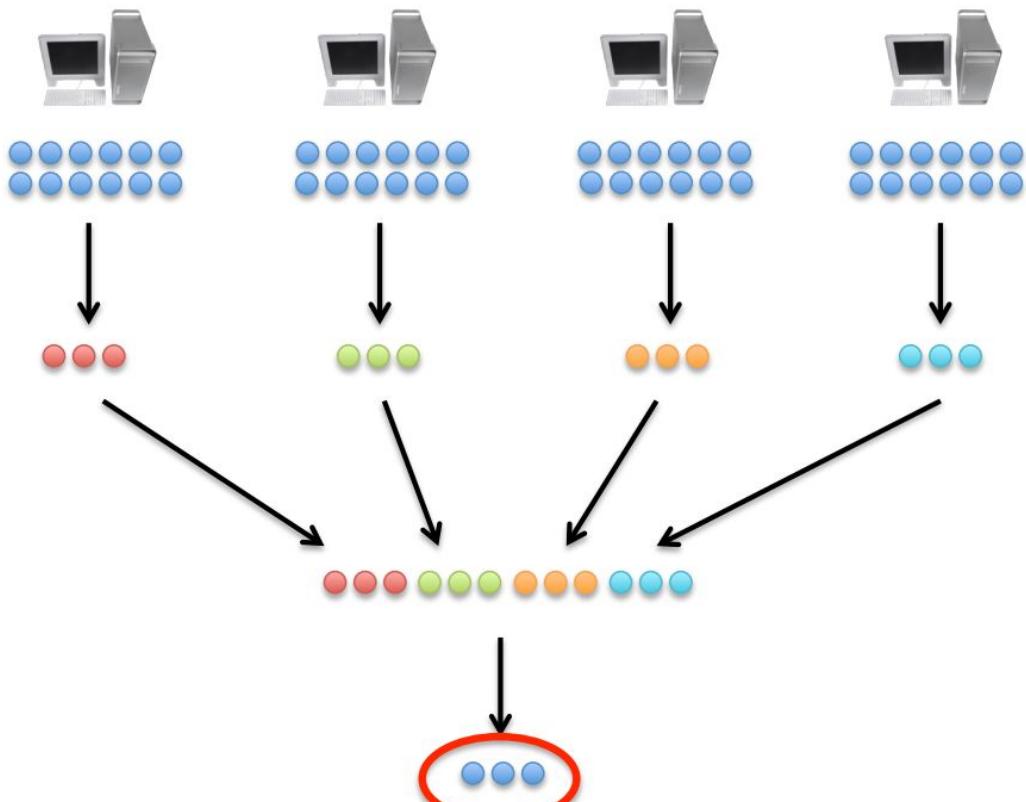


[Badanidiyuru et al 2014] **Theorem:** Sieve Streaming Is a single pass  $\frac{1}{2}$  approximation algorithm!



even more data ...  
distributed greedy algorithm?

# Distributed Greedy Algorithm I

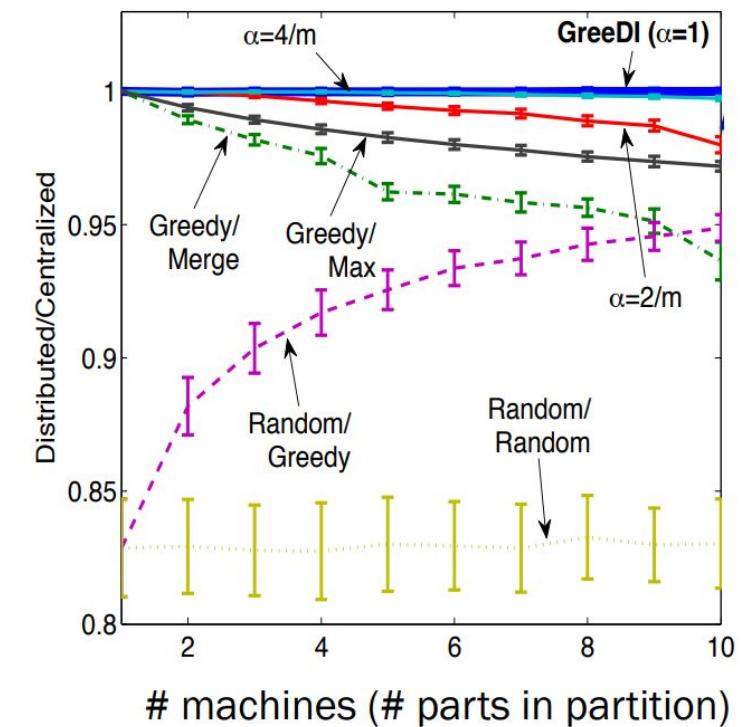


greedy is sequential.  
pick in parallel??

pick  $k$  elements  
on each machine.

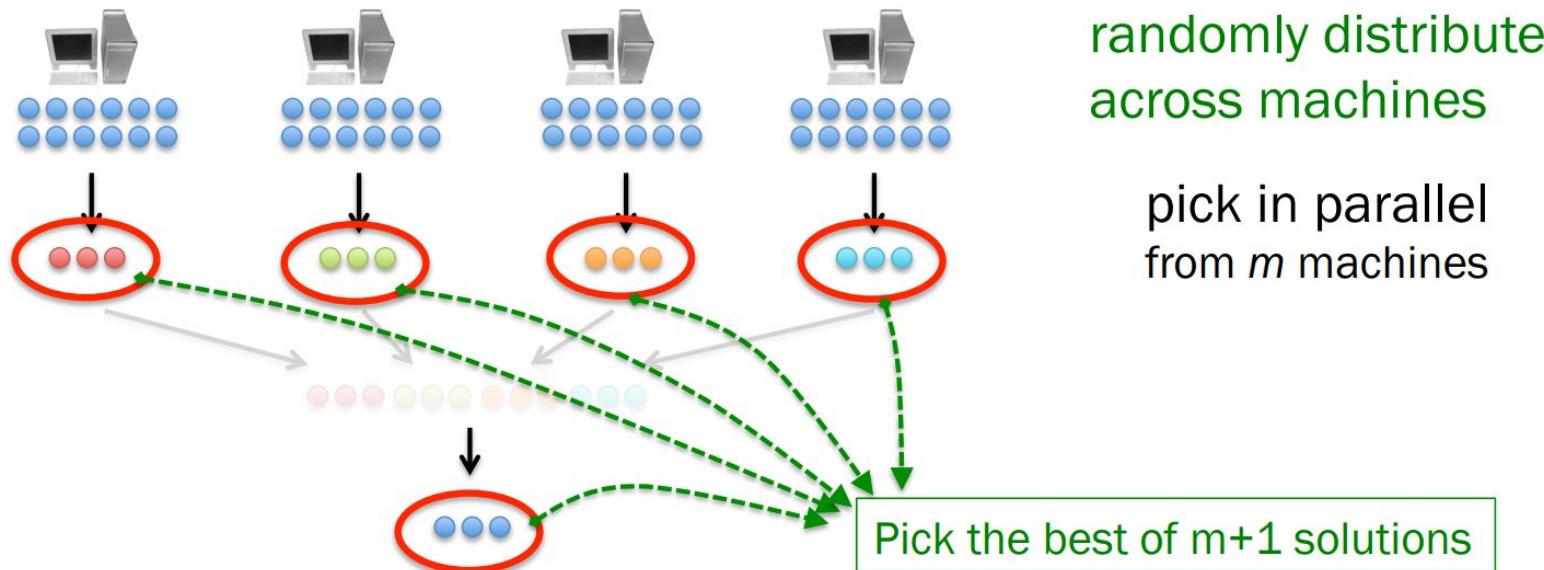
combine and run  
greedy again.

Approximation factor:  
 $O\left(\frac{1}{\min\{\sqrt{k}, m\}}\right)$



Mirzasoleiman et al 2013,

# Distributed Greedy Algorithm II



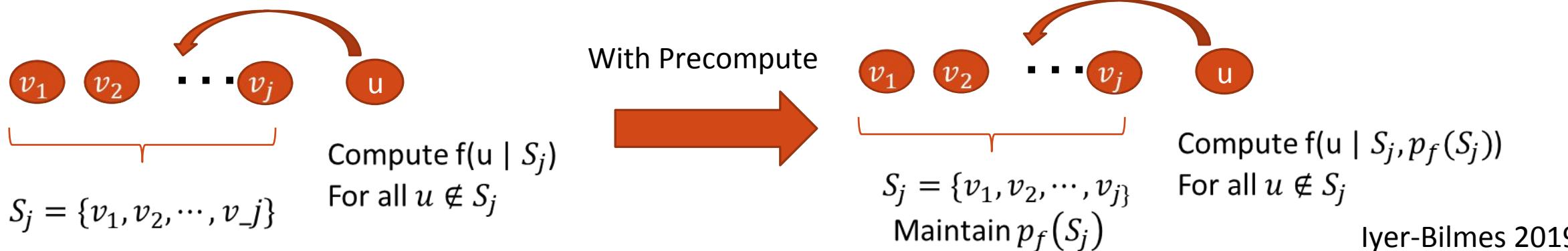
With greedy algorithm on both levels:  
 $\alpha = \beta = 1 - \frac{1}{e}$ , overall factor:  
 $\frac{1}{2}(1 - \frac{1}{e})$

- each machine:  $\alpha$ -approximation algorithm
  - level 2:  $\beta$ -approximation algorithm
- overall approximation factor:  $\mathbb{E}[F(\hat{S})] \geq \frac{\alpha\beta}{\alpha + \beta} F(S^*)$

de Ponte Barbosa et al 2015,

# Memoization Framework to Implement Submodular Optimization Algorithms

- ❑ Idea of Memoization: Cache intermediate variables to be able to faster compute the gains of adding elements to sets
- ❑ Maintain precomputed statistics for each set  $X$
- ❑ Given the precomputed statistics, the evaluation of the gains is much more efficient
- ❑ Significant speedups compared to simply using the value-oracle model



# Precomputed Statistics for Submodular Functions

Name	$f(X)$	$p_f(X)$	$T_f^o$	$T_f^p$
Facility Location	$\sum_{i \in V} \max_{k \in X} s_{ik}$	$[\max_{k \in X} s_{ik}, i \in V]$	$O(n^2)$	$O(n)$
Saturated Coverage	$\sum_{i \in V} \min\{\sum_{j \in X} s_{ij}, \alpha_i\}$	$[\sum_{j \in X} s_{ij}, i \in V]$	$O(n^2)$	$O(n)$
Graph Cut	$\lambda \sum_{i \in V} \sum_{j \in X} s_{ij} - \sum_{i,j \in X} s_{ij}$	$[\sum_{j \in X} s_{ij}, i \in V]$	$O(n^2)$	$O(n)$
Feature Based	$\sum_{i \in \mathcal{F}} \psi(w_i(X))$	$[w_i(X), i \in \mathcal{F}]$	$O(n \mathcal{F} )$	$O( \mathcal{F} )$
Set Cover	$w(\cup_{i \in X} U_i)$	$\cup_{i \in X} U_i$	$O(n U )$	$ U $
Prob. Set Cover	$\sum_{i \in \mathcal{U}} w_i [1 - \prod_{k \in X} (1 - p_{ik})]$	$[\prod_{k \in X} (1 - p_{ik}), i \in \mathcal{U}]$	$O(n \mathcal{U} )$	$O( \mathcal{U} )$
DPP	$\log \det(S_X)$	SVD( $S_X$ )	$O( X ^3)$	$O( X ^2)$
Dispersion Min	$\min_{k,l \in X, k \neq l} d_{kl}$	$\min_{k,l \in X, k \neq l} d_{kl}$	$O( X ^2)$	$O( X )$
Dispersion Sum	$\sum_{k,l \in X} d_{kl}$	$[\sum_{k \in X} d_{kl}, l \in X]$	$O( X ^2)$	$O( X )$
Dispersion Min-Sum	$\sum_{k \in X} \min_{l \in X} d_{kl}$	$[\min_{k \in X} d_{kl}, l \in X]$	$O( X ^2)$	$O( X )$

Table 1. List of Submodular Functions used, with the precompute statistics  $p_f(X)$ , gain evaluated using the precomputed statistics  $p_f(X)$  and finally  $T_f^o$  as the cost of evaluation the function without memoization and  $T_f^p$  as the cost with memoization. It is easy to see that memoization saves an order of magnitude in computation.

# Benefit of Memoization in Practice

---

Algorithm/Function	n = 4620	n = 192000	n = 10.2M
Lazy Greedy (Fac-Loc)	0.31	4.1	212.1
Lazy Greedy (Feat Based)	0.16	2.7	98.1
Min Norm Point (Complexity)	7.1	88.1	4321
SCSC (Feat-Based, Complexity)	1.2	12.3	567.1

Scaling across Dataset sizes

Algorithm/Function	Memoization	No Memoization
Lazy Greedy (Fac-Loc)	0.31	48
Lazy Greedy (Feat Based)	0.16	21
Min Norm Point (Complexity)	7.1	2023.1
SCSC (Feat-Based, Complexity)	1.2	101.2

Memoization vs No Memoization

Function	Ours	Gygli et al 2016	SFO (Krause et al 2008)
Facility Location	0.34	26.8	52
Graph Cut	0.39	35.7	43.2

SMTK vs Other ToolKits

# Big Picture of Submodular Optimization

## Submodular Functions

### Minimization

- Convex
- Complexity/Cooperation/Attraction
- Ex: Bipartite Neighbor, Graph Cut/Ising, Concave over Mod

### Maximization

- Concave
- Diversity/Representation/Coverage/Information
- Ex: Facility Location, DPPs, Set Cover, Dispersion etc.

## Learning Submodular Functions

### Minimization

- Log-Supermodular
- Ising Models
- Sampling, Mode, Approximate Partition Function

### Maximization

- Log-Submodular
- DPPs
- Sampling, Partition Fn, Maximum Likelihood
- Max-Margin learning

## Optimization Algorithms

### Minimization

- SFM Algorithms
- Majorization-Minimization
- Continuous Relaxations (Lovasz Extension)

### Maximization

- Greedy Algorithms
- Distributed Greedy/Streaming Greedy
- Randomized Greedy
- Bidirectional Greedy

## Applications

### Minimization

- Image Segmentation
- Sparse Reconstruction
- Limited Complexity Data Selection
- ...

### Maximization

- Data Subset Selection
- Image/Video Summarization
- Data Partitioning,
- ...

# Additional Reading

---

- Jeff's Class: [http://j.ee.washington.edu/~bilmes/classes/ee596a fall 2014/](http://j.ee.washington.edu/~bilmes/classes/ee596a_fall_2014/)
- Stefanie Jegelka & Andreas Krause's 2013 ICML tutorial:  
<http://techtalks.tv/talks/submodularity-in-machine-learning-new-directions-part-i/58125/>
- Jeff's NIPS, 2013 tutorial on submodularity:  
<http://melodi.ee.washington.edu/~bilmes/pgs/b2hd-bilmes2013-nips-tutorial.html> and  
<http://youtu.be/c4rBof38nKQ>
- Andreas Krause's web page: <http://submodularity.org>
- Francis Bach's updated 2013 text:  
[http://hal.archives-ouvertes.fr/docs/00/87/06/09/PDF/submodular\\_fot\\_revisd\\_hal.pdf](http://hal.archives-ouvertes.fr/docs/00/87/06/09/PDF/submodular_fot_revisd_hal.pdf)
- My WACV 2019 Tutorial: <https://sites.google.com/view/wacv2019summarization/home>
- Tom McCormick's overview paper on submodular minimization:  
<http://people.commerce.ubc.ca/faculty/mccormick/sfmchap8a.pdf>

# References

---

1. Mirzasoleiman, Baharan, Stefanie Jegelka, and Andreas Krause. **Streaming non-monotone submodular maximization: Personalized video summarization on the fly.** *arXiv preprint arXiv:1706.03583* (2017).
2. Rishabh Iyer and Jeff Bilmes, **A Memoization Framework for Scaling Submodular Optimization to Large Scale Problems**, Artificial Intelligence and Statistics (AISTATS) 2019
3. Rishabh Iyer, Stefanie Jegelka, Jeff Bilmes, **Fast semidifferential-based submodular function optimization**, International Conference on Machine Learning (ICML) 2013
4. Mirzasoleiman, Baharan, et al. **Distributed submodular maximization: Identifying representative elements in massive data.** Advances in Neural Information Processing Systems. 2013.
5. Barbosa, R., Ene, A., Nguyen, H., & Ward, J. (2015, June). **The power of randomization: Distributed submodular maximization on massive datasets.** In International Conference on Machine Learning (pp. 1236-1244).
6. Fisher, Marshall L., George L. Nemhauser, and Laurence A. Wolsey. **An analysis of approximations for maximizing submodular set functions.** " Polyhedral combinatorics. Springer, Berlin, Heidelberg, 1978. 73-87.
7. Mirzasoleiman, Baharan, et al. **Lazier Than Lazy Greedy.** AAAI. 2015.
8. Buchbinder, Niv, et al. **Submodular maximization with cardinality constraints.** Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, 2014.