Massachusetts Institute of Technology

Submodular Functions – Part II

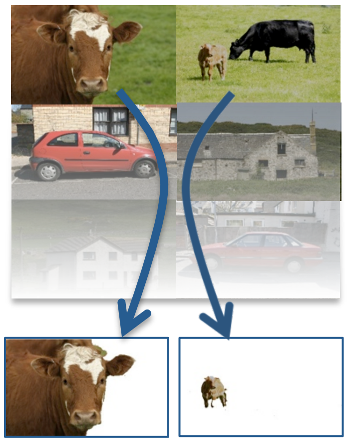
ML Summer School Cádiz

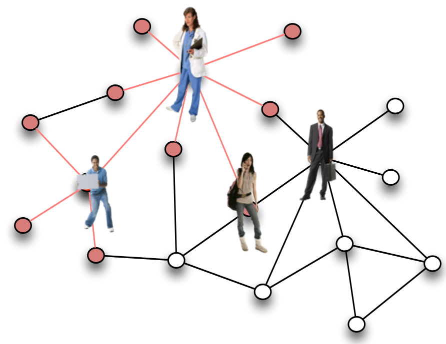
Stefanie Jegelka

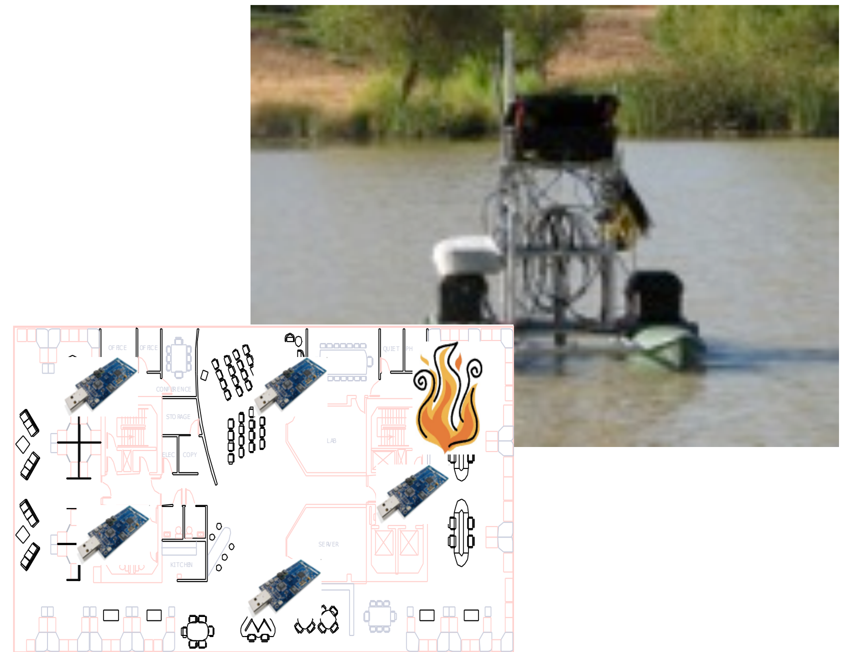
MIT

more reading & papers: http://people.csail.mit.edu/stefje/mlss/literature.pdf

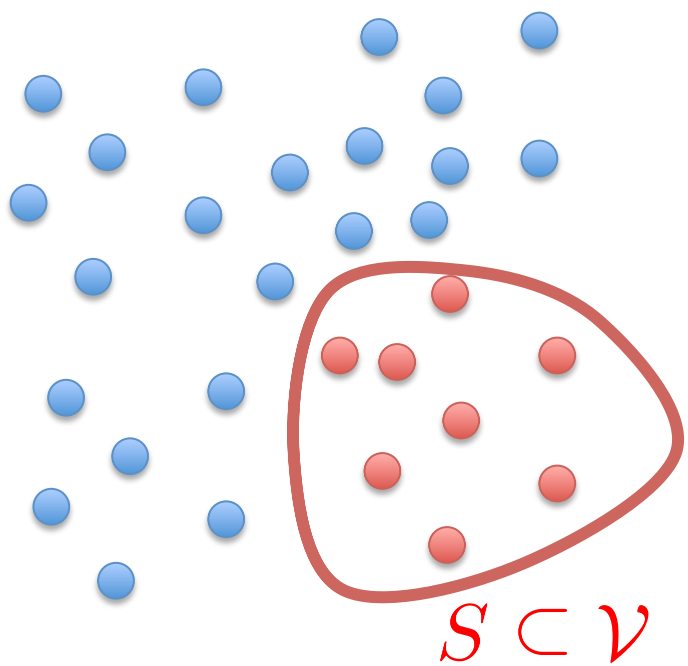
Set functions in machine learning







Setup



- ground set $\mathcal{V}$
- (scoring) function $F : 2^{\mathcal{V}} \to \mathbb{R}_+$

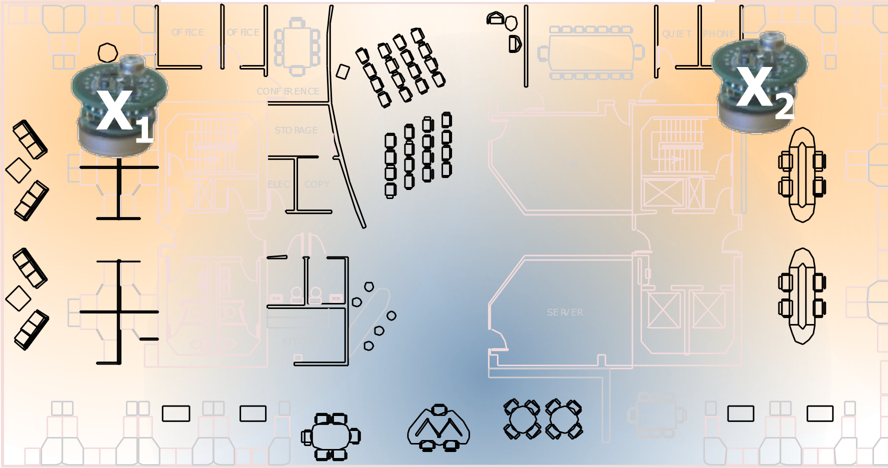
$$\max \quad F(S)$$

$$\min_{S \subseteq \mathcal{V}} F(S)$$

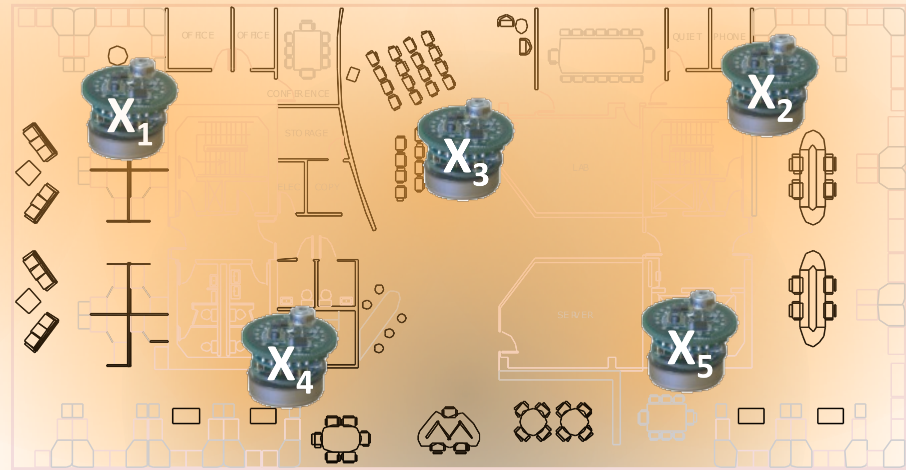
We assume:
- $F(\emptyset) = 0$
- we can evaluate *F*

Diminishing marginal gains

placement A = {1,2}



placement B = {1,…,5}



Big gain

new sensor s

small gain

A + • s

B + • s

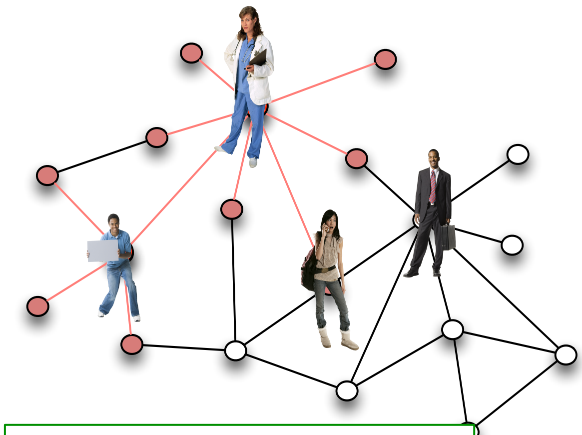
$$A \subseteq B$$

$$F(A \cup s) - F(A) \quad \geq \quad F(B \cup s) - F(B)$$

Maximizing submodular utility

maximize information / coverage

(Krause & Guestrin 2005)



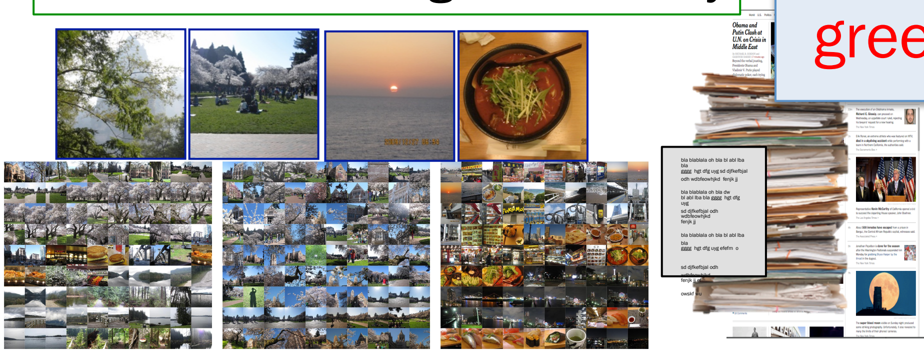
maximize influence

e, Kleinberg, Tardos 2003, Mossel & Roch 2007)

$$\max_{|S| \leq k} F(S)$$

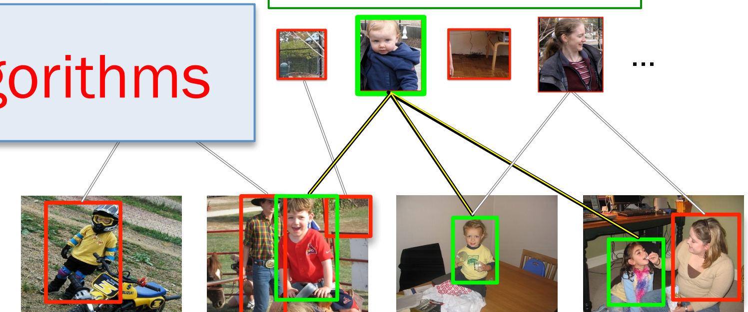
greedy algorithms

maximize coverage & diversity



(Lin & Bilmes 2011, Tschiatschek et al 2014, Kim et al 2014, Gygli et al 2015...)

find exemplars



(Song, Lee, Jegelka, Darrell 2014, Song, Girshick, Jegelka, Mairal, Harchaoui, Darrell 2014, Kim et al 2011)

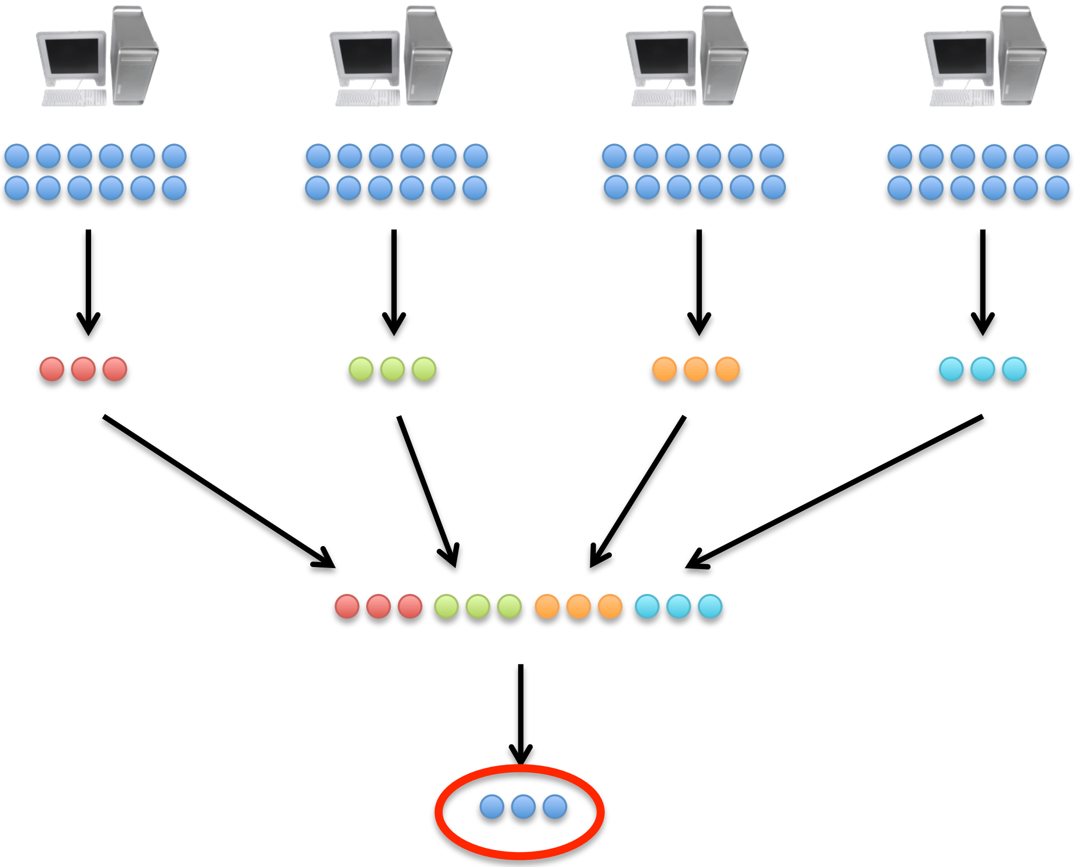
Questions

- What if I have more complex constraints?
  - matroid constraints
  - budget constraints
- Greedy takes *O(nk)* time. What if n, k are large?
  - stochastic
  - distributed
  - structured
- What if my function is not monotone?

even more data ...

distributed greedy algorithm?

Distributed greedy algorithms



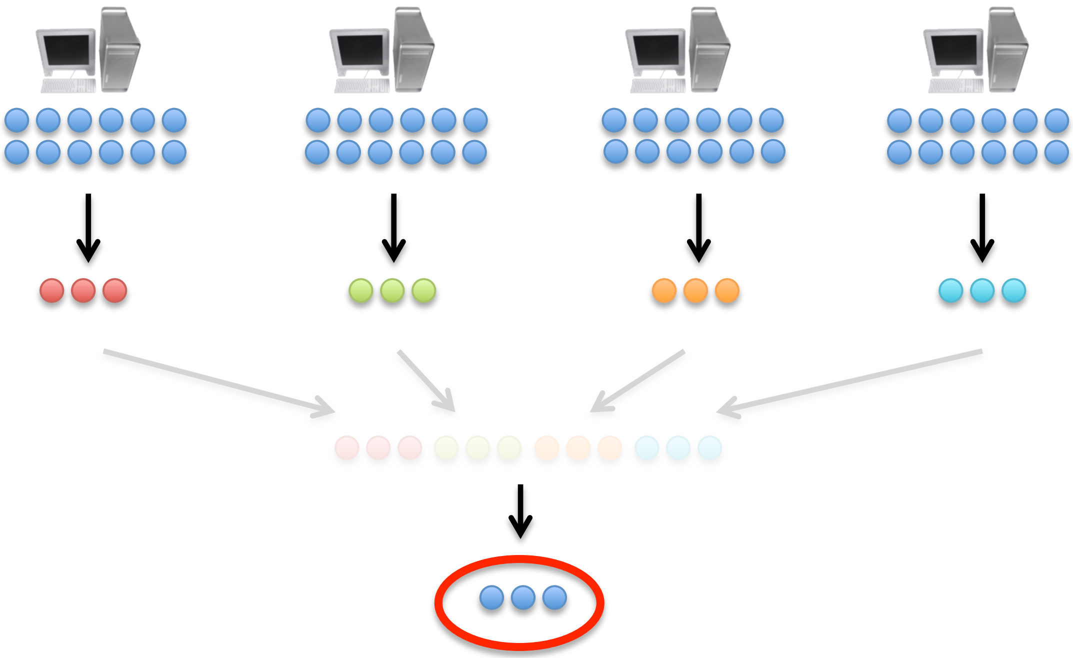
greedy is sequential.
pick in parallel??

pick k elements
on each machine.

combine and run
greedy again.

Is this useful?

Distributed greedy algorithms



pick in parallel
from *m* machines

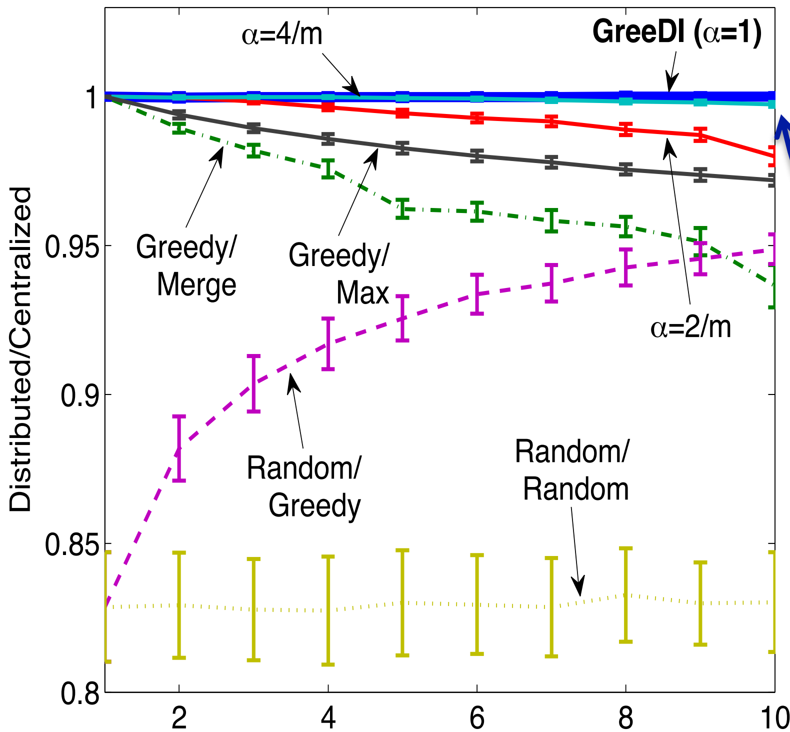
Is this useful?

Approximation factor:

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

(*Mirzasoleiman et al 2013*)

Distributed Greedy



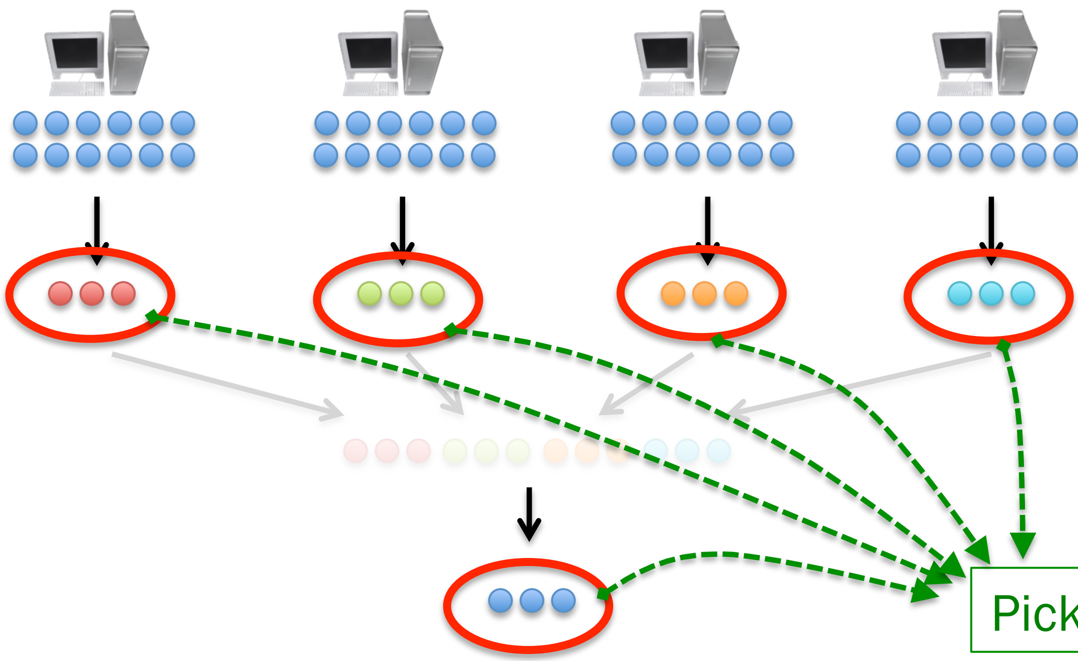
machines (# parts in partition)

In practice, performs often quite well.

1. special structure: Improved guarantees if F is Lipschitz or a sum of many terms
2. randomization

(Mirzasoleiman et al 2013)

Distributed greedy algorithms



randomly distribute across machines

pick in parallel
from *m* machines

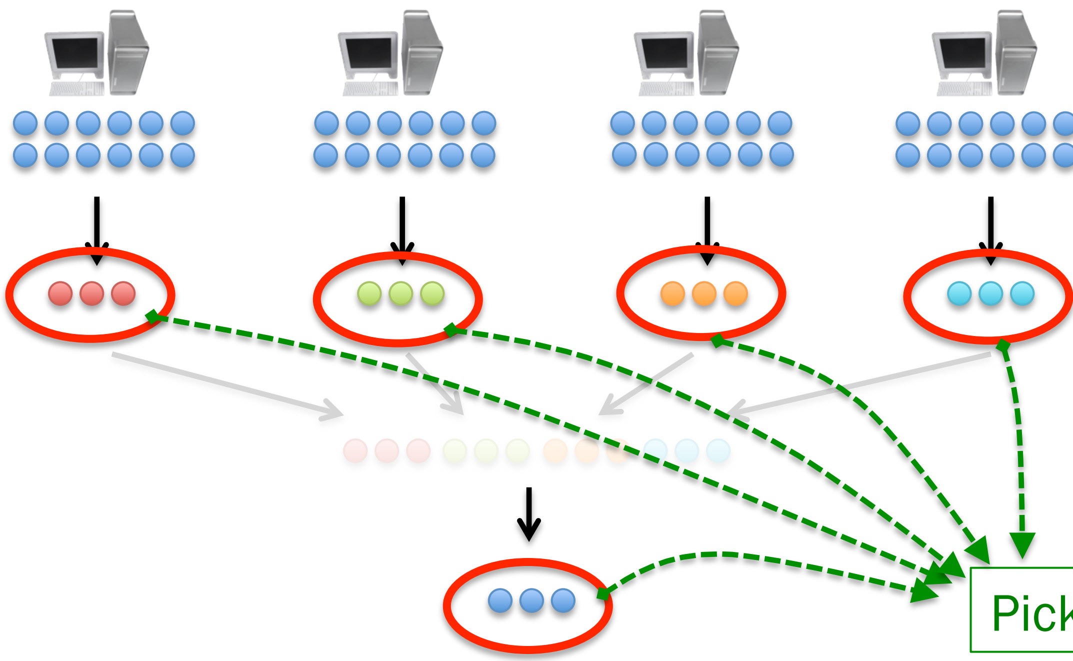
Pick the best of m+1 solutions

- each machine: $\alpha-$approximation algorithm
- level 2: $\beta-$ approximation algorithm

➔ overall approximation factor: $$\mathbb{E}[F(\widehat{S})] \geq \frac{\alpha\beta}{\alpha+\beta} F(S^*)$$

(*Mirzasoleiman et al 2013, de Ponte Barbosa et al 2015, see also Mirrokni, Zadimoghaddam 2015*)

Distributed greedy algorithms



randomly distribute across machines

pick in parallel
from *m* machines

Pick the best of m+1 solutions

$$\mathbb{E}[F(\widehat{S})] \;\geq\; \frac{\alpha\beta}{\alpha+\beta} F(S^*)$$

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

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

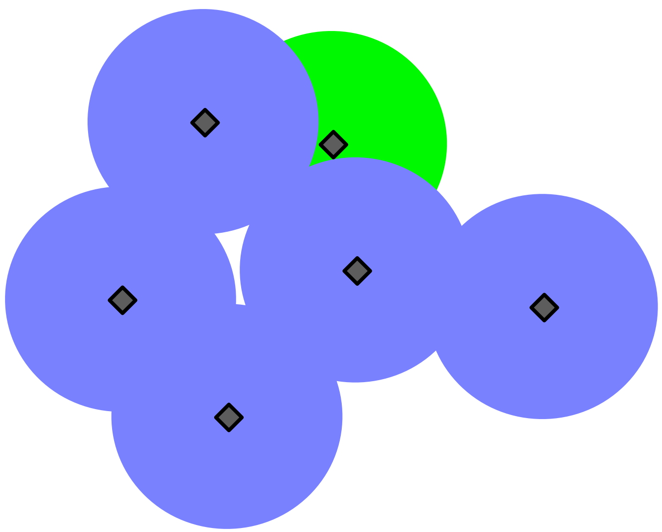
(Mirzasoleiman et al 2013, de Ponte Barbosa et al 2015, see also Mirrokni, Zadimoghaddam 2015)

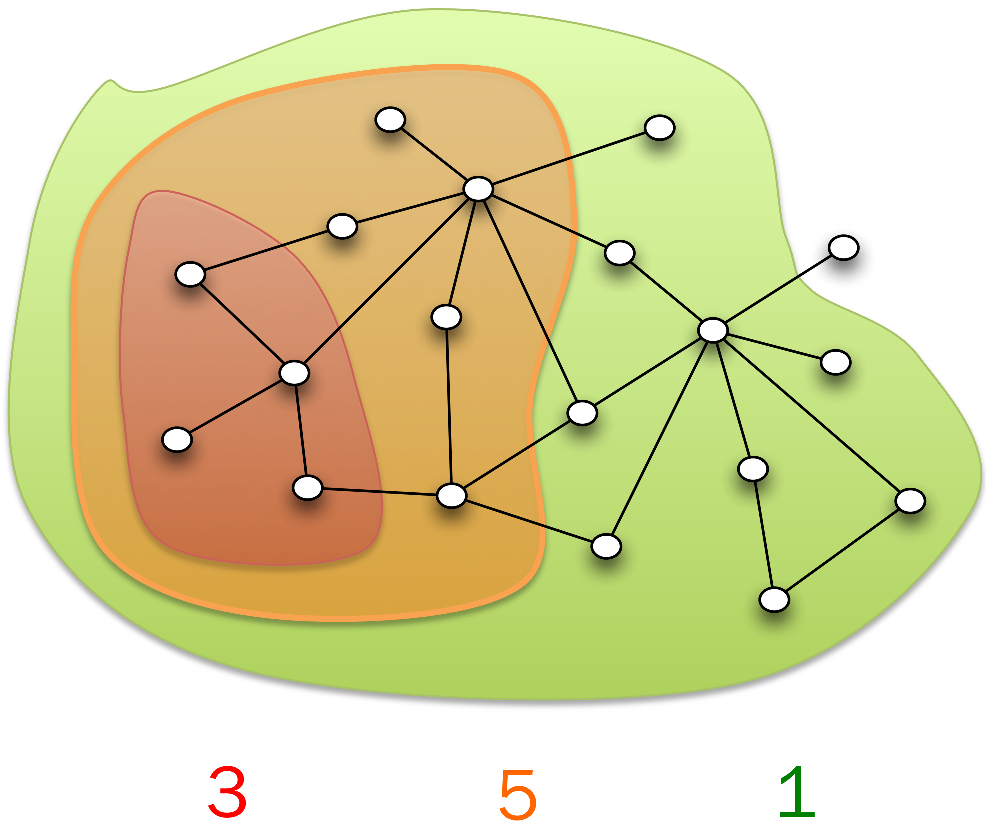
Questions

- What if I have more complex constraints?
  - matroid constraints: later (Sri)
  - budget constraints
- Greedy takes *O(nk)* time. What if n, k are large?
  - stochastic
  - distributed
  - structured
- What if my function is not monotone?

Non-monotone functions

$$\cancel{\text{if } \; S \subseteq T \quad \text{then} \quad F(S) \leq F(T)}$$





still assume:

$$F(S) \geq 0 \quad \text{for all } \; S$$

Greedy can fail ...

$$F(A) = \left| \bigcup_{a \in A} \mathrm{area}(a) \right| - \sum_{a \in A} c(a)$$

greedy

$F(A)$

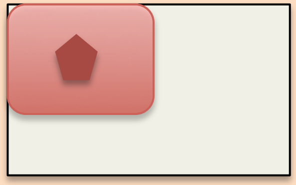
optimal solution

$F(A) = 95$

sensor 1

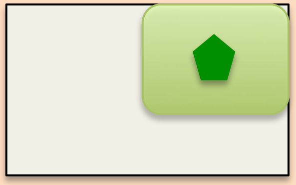
coverage: 100
cost: -60
gain 40

sensor 2



coverage: 30
cost: - 1
gain 29

sensor 3



coverage: 30
cost: - 1
gain 29

sensor 4

coverage: 40
cost: - 3
gain 37

$$S_0 = \emptyset \qquad S_1 = \emptyset \cup \arg\max_{a \in \mathcal{V}} F(a)$$

Greedy can fail ...

$$F(A) = \left| \bigcup_{a \in A} \text{area}(a) \right| - \sum_{a \in A} c(a)$$

greedy solution:

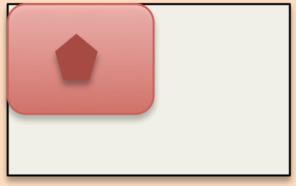
$F(A) = 40$

optimal solution: $F(A) = 95$

sensor 1

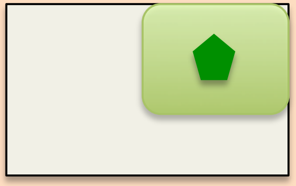
coverage: 100
cost: -60
gain 40

sensor 2



coverage: 30
cost: - 1
gain 29

sensor 3

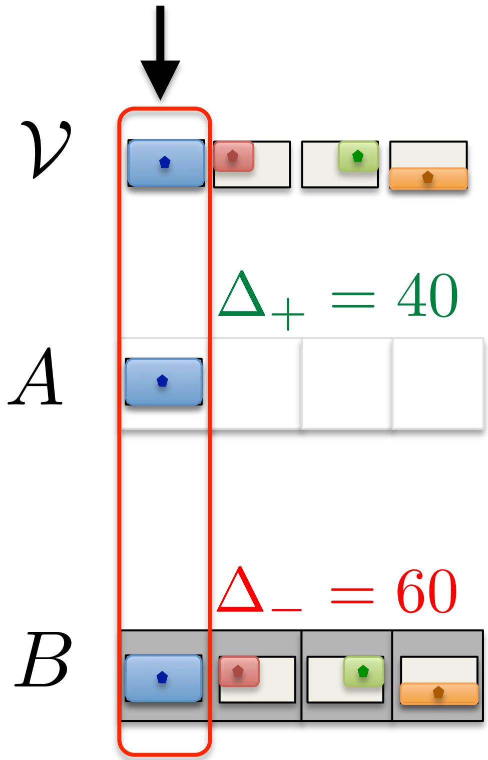


coverage: 30
cost: - 1
gain 29

sensor 4

coverage: 40
cost: - 3
gain 37

Double (bidirectional) greedy



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

for *i=1, ..., n* *//add or remove?*

- gain of adding (to A):

$$\Delta_+ = [\, F(A \cup a_i) - F(A)\,]_+$$

- gain of removing (from B):

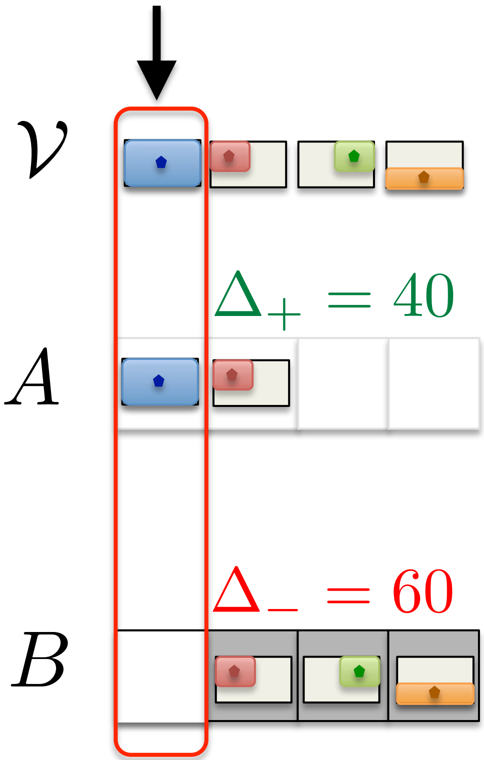
$$\Delta_- = [\, F(B \setminus a) - F(B)\,]_+$$

add with probability

$$\mathbb{P}(\mathrm{add}) = \frac{\Delta_+}{\Delta_+ + \Delta_-} \quad = 40\%$$

coverage: 100
cost: -60

Double (bidirectional) greedy



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

for *i=1, ..., n* *//add or remove?*

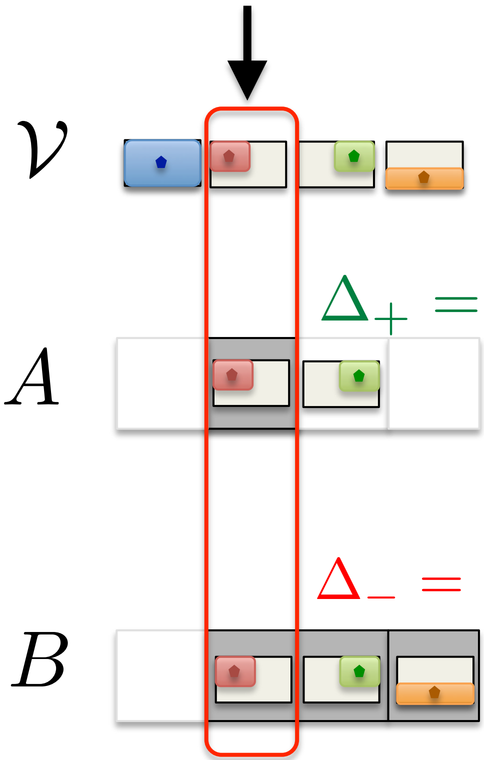
add with probability

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

add to A or $\boxed{\text{remove from B}}$

coverage:	100
cost:	-60

Double (bidirectional) greedy



$\Delta_+ = 29$

$\Delta_- = [-29]_+ = 0$

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

for *i=1, ..., n* *//add or remove?*

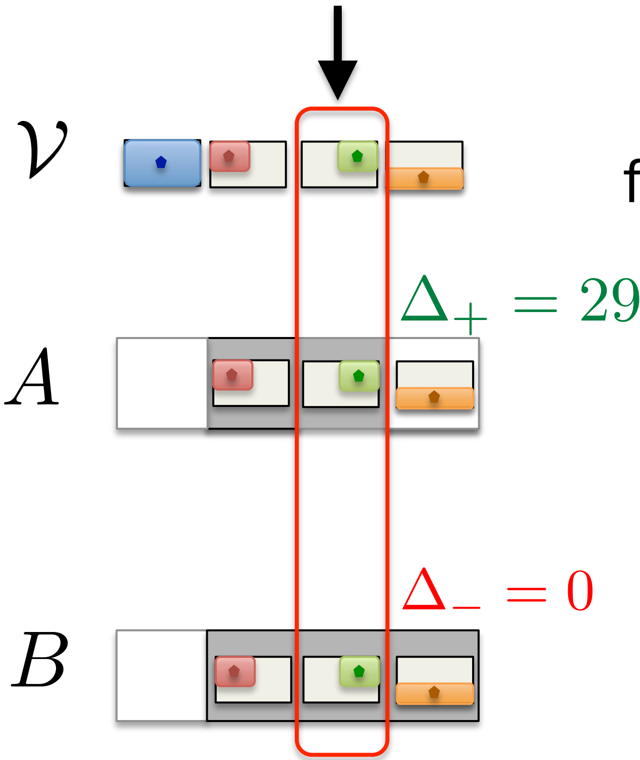
add with probability

$$\mathbb{P}(\text{add}) = \frac{\Delta_+}{\Delta_+ + \Delta_-} = \frac{29}{29}$$

$\boxed{\text{add to A}}$ or remove from B

coverage: 30
cost: - 1

Double (bidirectional) greedy



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

for *i=1, ..., n* *//add or remove?*

add with probability

$$\mathbb{P}(\text{add}) = \frac{\Delta_+}{\Delta_+ + \Delta_-} = \frac{29}{29} \quad \frac{37}{40}$$

add to A or remove from B

coverage: 30
cost: - 1

Double greedy

$$\max_{S \subseteq \mathcal{V}} \; F(S)$$

Theorem *(Buchbinder, Feldman, Naor, Schwartz '12)*

F submodular, S_g solution of double greedy. Then

$$\mathbb{E}[F(S_g)] \geq \frac{1}{2} F(S^*)$$

optimal solution

Non-monotone maximization

- alternatives to double greedy?
  local search *(Feige et al 2007)*

- constraints?
  possible, but different algorithms

- distributed algorithms? yes!
  - divide-and-conquer as before *(de Ponte Barbosa et al 2015)*
  - concurrency control / Hogwild *(Pan et al 2014)*

Submodular maximization: summary

- many applications: diverse, informative subsets

- NP-hard, but greedy or local search
- distinguish monotone / non-monotone

- several constraints possible
  (monotone and non-monotone)

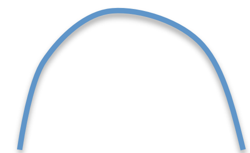
Roadmap

- Submodular set functions
  - what is this? where does it occur? how recognize?

- Maximizing submodular functions:

  diversity, repulsion, concavity

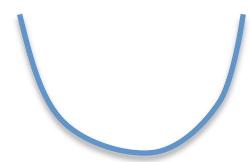
  greed is not too bad



- Minimizing submodular functions:

  coherence, regularization, convexity

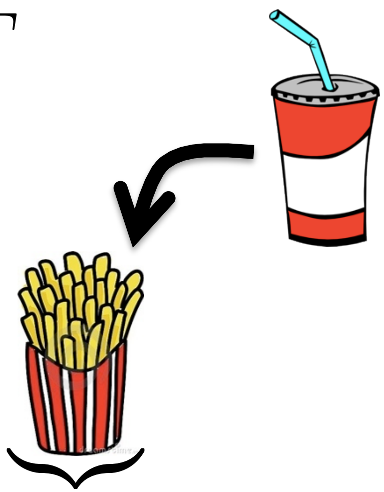
  the magic of "discrete analog of convex"



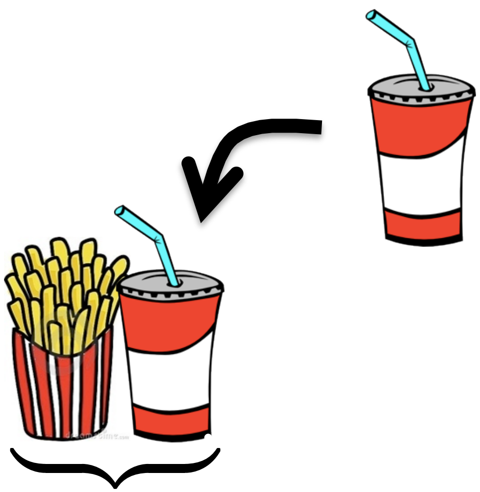
- Other questions around submodularity & ML

Submodularity

$S \subseteq T$



S



T

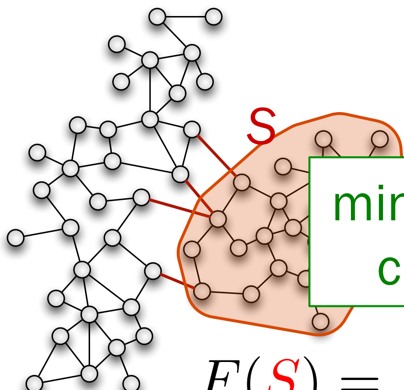
$$F(S \cup s) - F(S) \quad \geq \quad F(T \cup s) - F(T)$$

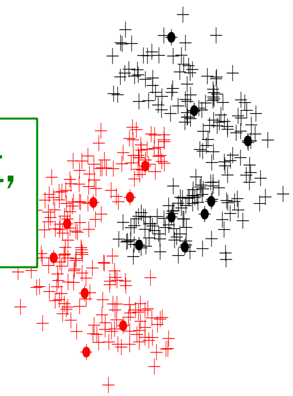
extra cost:
one drink

extra cost:
free refill ☺

diminishing marginal costs – economies of scale

Minimize incoherence

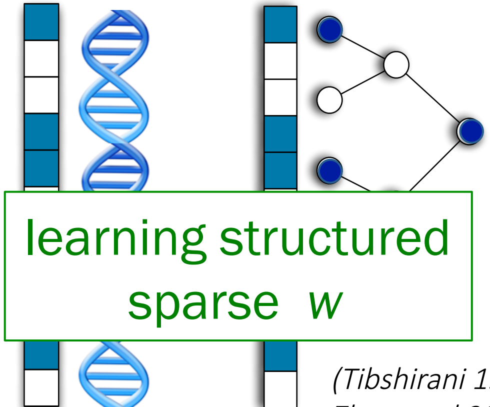




minimum cut, clustering

$$F(S) = \sum_{e \text{ cut}} w(e)$$

$$\text{loss}(w) + F(S)$$



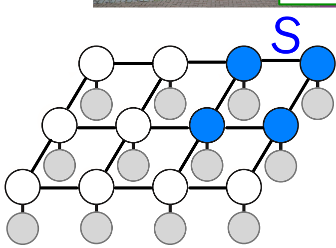
learning structured sparse w

$$S = \text{support}(w)$$

(Tibshirani 1996, Chen et al 1998, Zhao et al 2009, Bach 2010,…)

MAP inference

$$\max_x P(x \mid I)$$



spatial/temporal coherence

(Ising, Potts, Porteous et al, …)

$$P(x = 1_S) \propto \exp(-F(S))$$

$$\boxed{\min_{S \subseteq \mathcal{V}} \quad F(S)}$$

Minimize incoherence/
maximize coherence

Convex functions *(Lovász, 1983)*

- "occur in many models in economy, engineering and other sciences", "often the only nontrivial property that can be stated in general"
- preserved under many operations and transformations: larger effective range of results
- sufficient structure for a "mathematically beautiful and practically useful theory"
- efficient minimization

"It is less apparent, but we claim and hope to prove to a certain extent, that a similar role is played in discrete optimization by *submodular set-functions*" [...]
they share the above four properties.

Submodular Minimization in 3 steps

1. Relaxation: continuous (Lovasz) extension

2. submodular polyhedra show: this is convex!

3. minimization via convex optimization

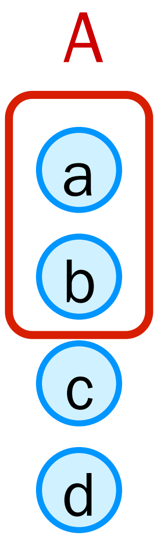
Submodularity and convexity

any set function

with $|V| = n$.

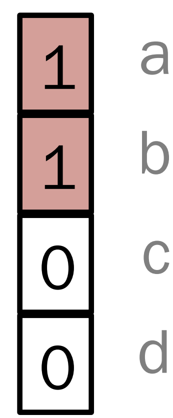
... is a function on binary vectors

$$F : 2^V \to \mathbb{R} \qquad\qquad F : \{0,1\}^n \to \mathbb{R}$$



$$\hat{=}$$

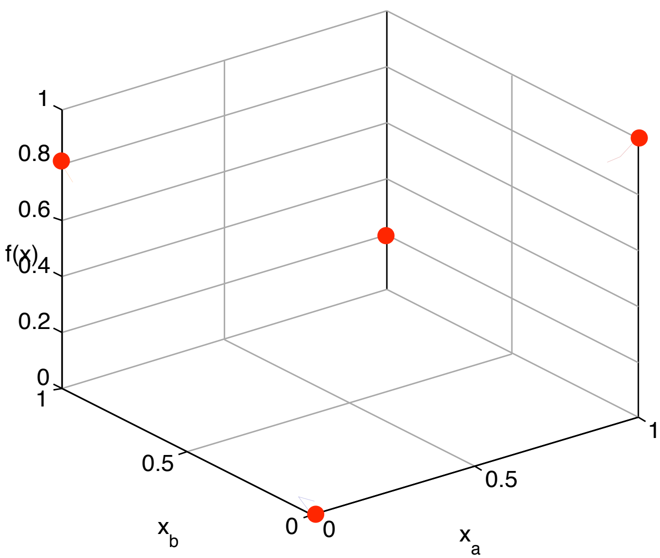
$$x = 1_A$$

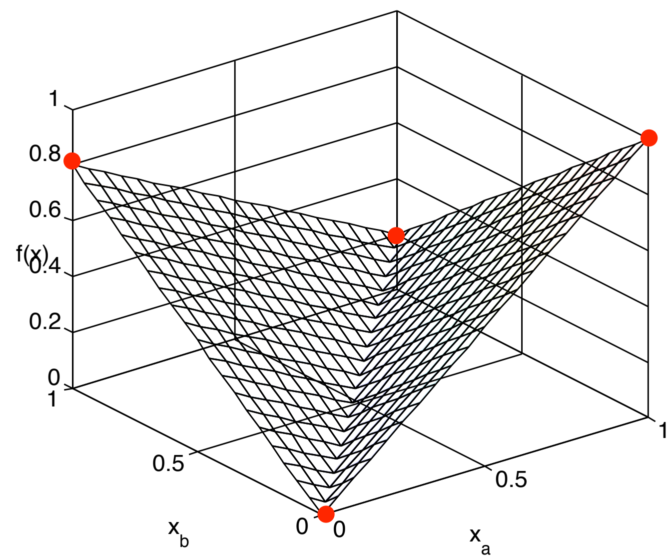


optimizing set function = finding binary labeling!

Relaxation: idea

$$\min_{x \in \{0,1\}^n} F(x) \longrightarrow \min_{x \in [0,1]^n} f(x)$$





this should be
“easy” to minimize

Relaxations

have

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

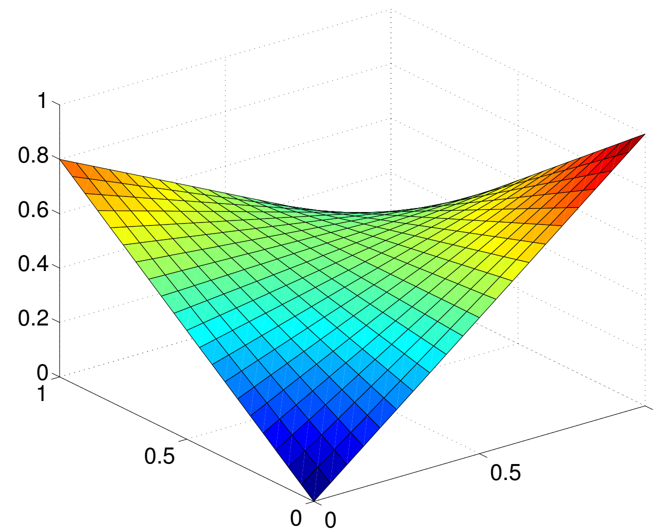
want: extension

$$f : \mathbb{R}^n_+ \to \mathbb{R}$$

$$f(1_S) = F(S)$$

- assume for the moment vectors $x \in [0,1]^n$
- recall multilinear extension: use expectation ☺

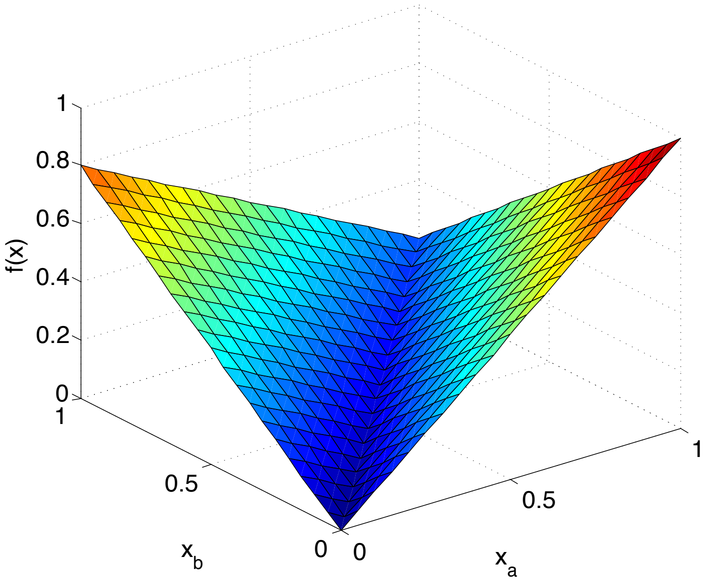
$$f_M(x) = \mathbb{E}_{S \sim p_x}[F(S)]$$



but: not easy to minimize.

We want a convex function!

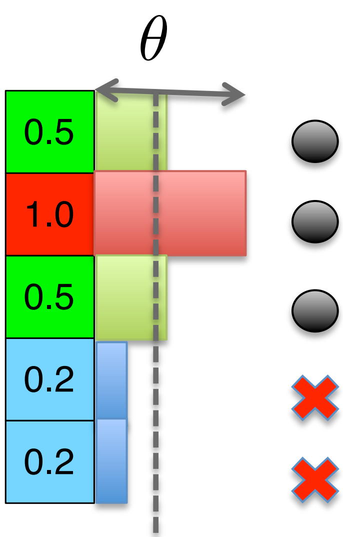
Lovász extension



- sample a threshold θ uniformly between 0 and 1
- Pick

$$S^{\theta} = \{\, i \mid x_i \geq \theta \,\}$$

$$f_L(x) = \mathbb{E}_{S \sim \theta}\left[F(S)\right]$$



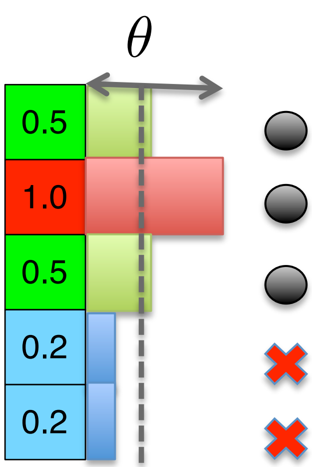
e.g. $\theta = 0.4$

$$\boxed{f(x) = \sum_{i=1}^{k} \alpha_i \, F(S_i)}$$

Lovász extension

$$f_L(x) = \mathbb{E}_{S\sim\theta}\left[F(S)\right]$$

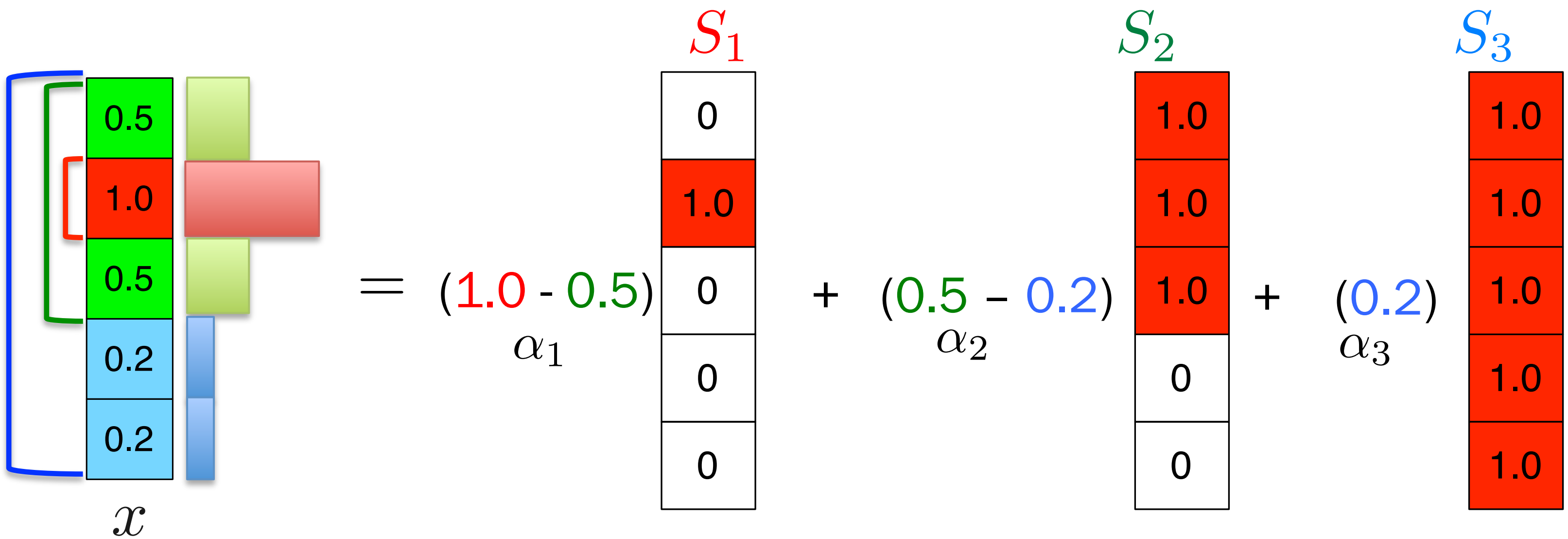
$$\underset{x}{\begin{bmatrix} 0.5 \\ 1.0 \\ 0.5 \\ 0.2 \\ 0.2 \end{bmatrix}} = \underset{\alpha_1}{(1.0 - 0.5)} \overset{S_1}{\begin{bmatrix} 0 \\ 1.0 \\ 0 \\ 0 \\ 0 \end{bmatrix}} + \underset{\alpha_2}{(0.5 - 0.2)} \overset{S_2}{\begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 0 \\ 0 \end{bmatrix}} + \underset{\alpha_3}{(0.2)} \overset{S_3}{\begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}}$$



$$x = \sum_{i=1}^{k} \alpha_i \mathbf{1}_{S_i}$$

$$\boxed{f(x) = \sum_{i=1}^{k} \alpha_i F(S_i)}$$

Lovász extension is easy to compute!



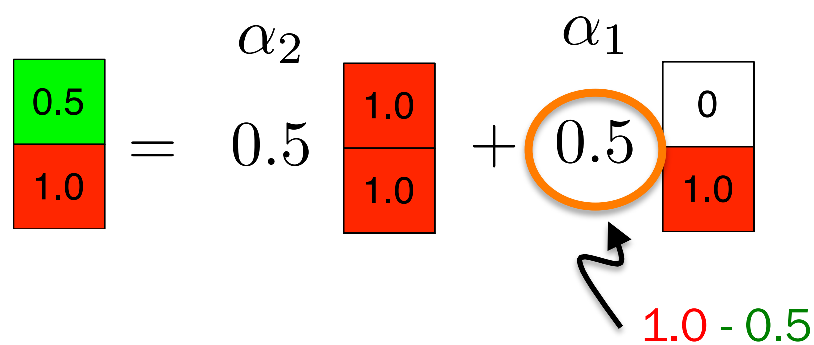
1. sort x: $\quad x_{\pi(1)} \geq x_{\pi(2)} \geq \ldots \geq x_{\pi(n)}$
2. then $\quad \alpha_i = x_{\pi(i)} - x_{\pi(i-1)}, \;\; \alpha_n = x_{\pi(n)}$

$$S_i = \{\pi(1), \ldots \pi(i)\}$$

$$\boxed{f(x) = \sum_{i=1}^{k} \alpha_i \, F(S_i)}$$

Examples

$$f(x) = \sum_{i=1}^{k} \alpha_i \, F(S_i)$$



- truncation

$$F(S) = \min\{|S|, 1\}$$

$$f(x) = 0.5 + 0.5 \quad = \max_i \; x_i$$

- cut function

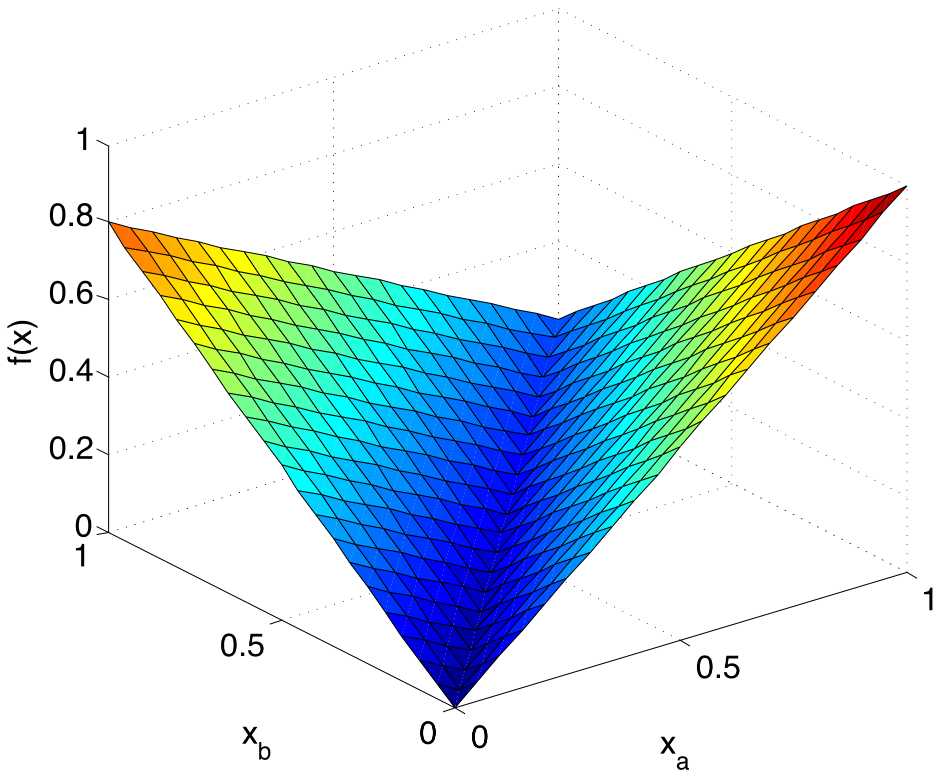
$$F(S) = \begin{cases} 1 & \text{if } S = \{1\}, \{2\} \\ 0 & \text{if } S = \emptyset, \{1, 2\} \end{cases}$$

$$f(x) = 0.5 \cdot 0 + (1 - 0.5) \cdot 1$$

$$= |x_1 - x_2|$$

"total variation"!

Is this useful?



✓ easy to compute (sort)

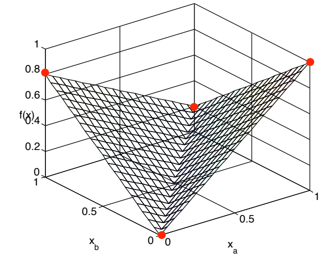
- convex?

Alternative characterization

$$f(x) = \sum_{i=1}^{k} \alpha_i \, F(S_i)$$

if *F* is submodular, this is equivalent to:

$$\boxed{f(x) = \max_{y \in \mathcal{B}_F} \; y^\top x}$$



Theorem *(Lovász, 1983)*
Lovasz extension is convex $\Leftrightarrow$ F is submodular.

Submodular polyhedra

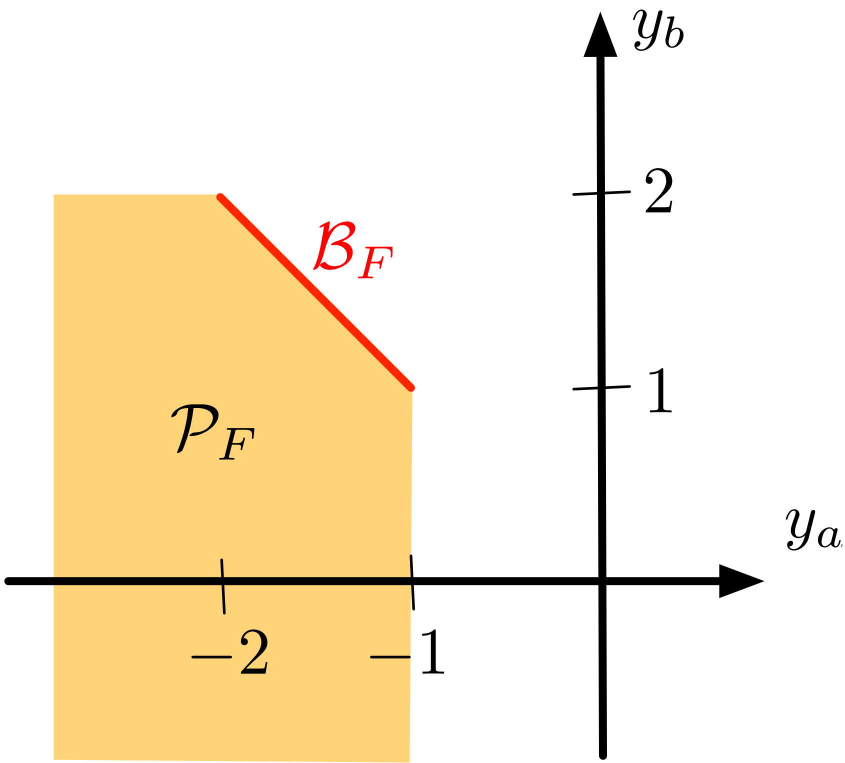
submodular polyhedron:

$$\mathcal{P}_F = \{y \in \mathbb{R}^n \mid y(A) \leq F(A) \text{ for all } A \subseteq \mathcal{V}\}$$

$$y(A) = \sum_{a \in A} y_a$$

Base polytope

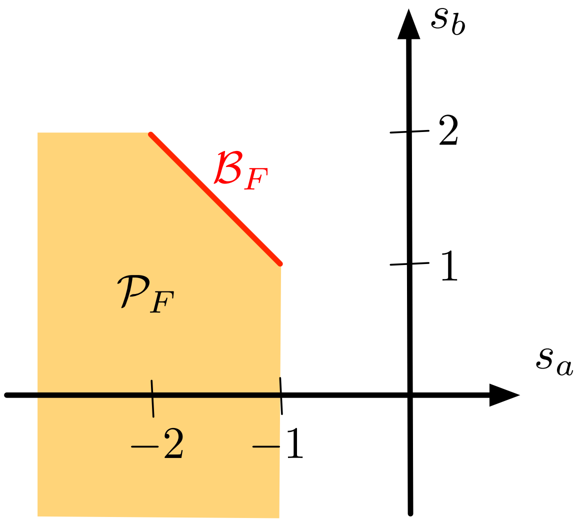
$$\mathcal{B}_F = \{y \in \mathcal{P}_F \mid y(\mathcal{V}) = F(\mathcal{V})\}$$



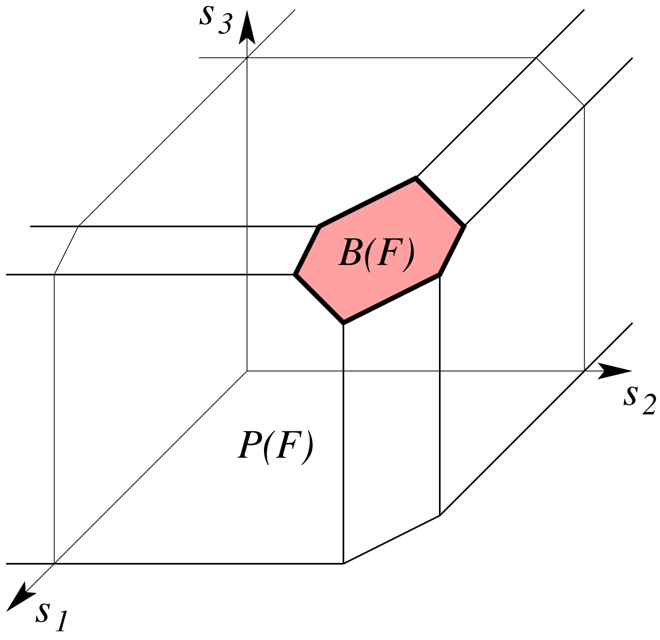
A	$F(A)$
$\emptyset$	0
a	-1
b	2
$\{a,b\}$	0

Base polytopes

2D (2 elements)



3D (3 elements)



Other interesting base polytopes

$$\mathcal{P}_F = \{y \in \mathbb{R}^n \mid y(A) \leq F(A) \text{ for all } A \subseteq \mathcal{V}\}$$

$$\mathcal{B}_F = \{y \in \mathcal{P}_F \mid y(\mathcal{V}) = F(\mathcal{V})\}$$

- Probability Simplex

$$F(S) = \min\{|S|, 1\}$$

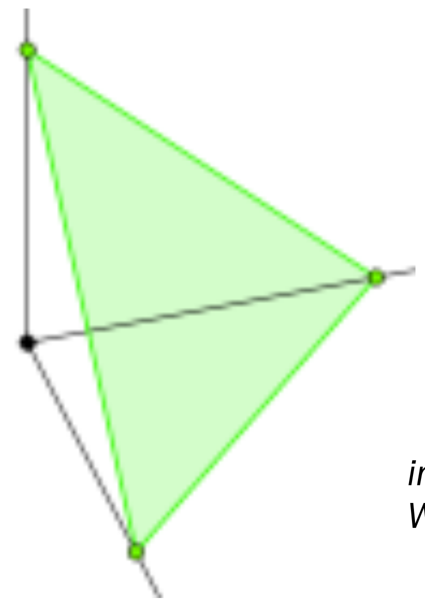


image source: Wikipedia

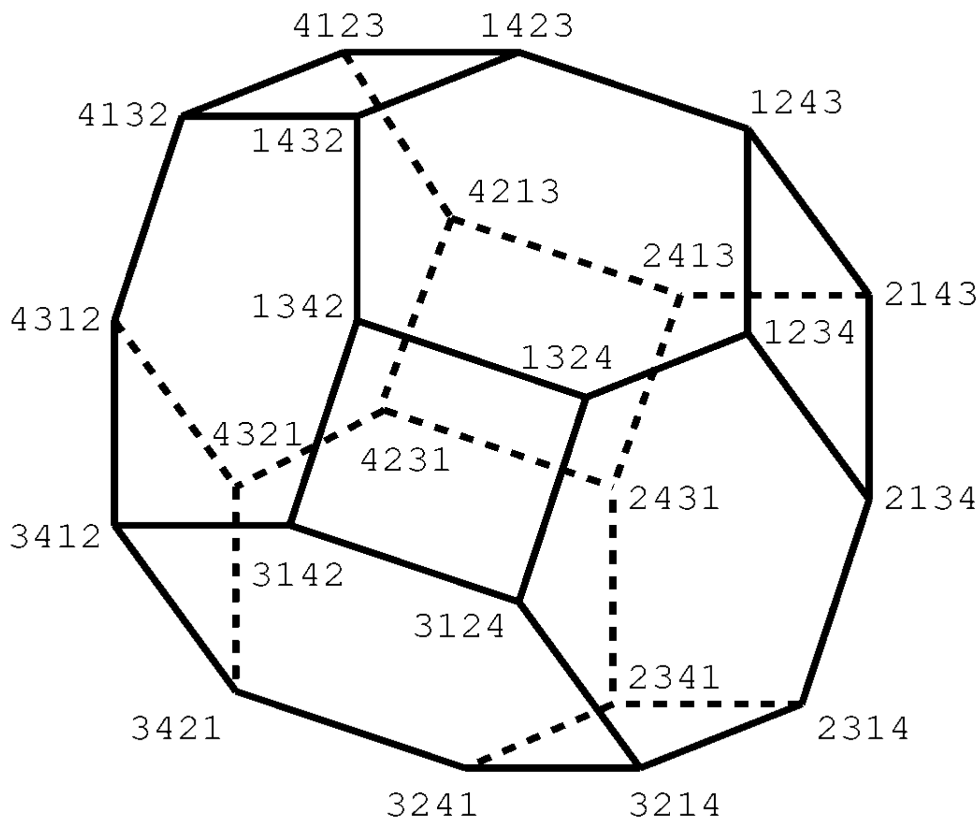
Other interesting base polytopes

$$\mathcal{P}_F = \{y \in \mathbb{R}^n \mid y(A) \le F(A) \text{ for all } A \subseteq \mathcal{V}\}$$

$$\mathcal{B}_F = \{y \in \mathcal{P}_F \mid y(\mathcal{V}) = F(\mathcal{V})\}$$

- Permutahedron

$$F(S) = \sum_{i=1}^{|S|}(n - i + 1)$$



Computing the "Lovasz extension"

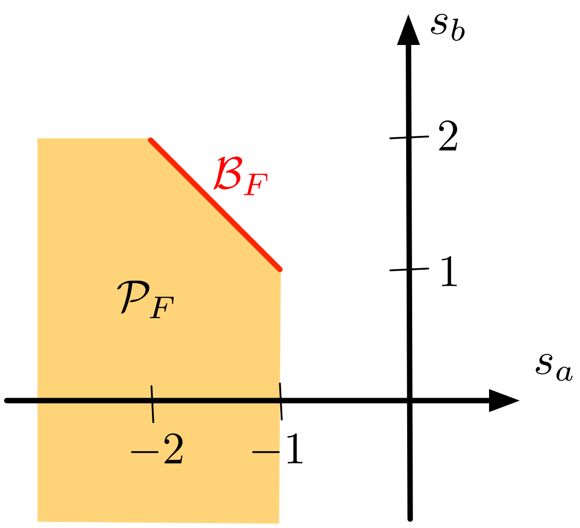
$$\mathcal{P}_F = \{y \in \mathbb{R}^n \mid y(A) \le F(A) \text{ for all } A \subseteq \mathcal{V}\}$$

exponentially many constraints!

Base polytope

$$\mathcal{B}_F = \{y \in \mathcal{P}_F \mid y(\mathcal{V}) = F(\mathcal{V})\}$$

$$\boxed{f(x) = \max_{y \in \mathcal{B}_F} y^\top x}$$



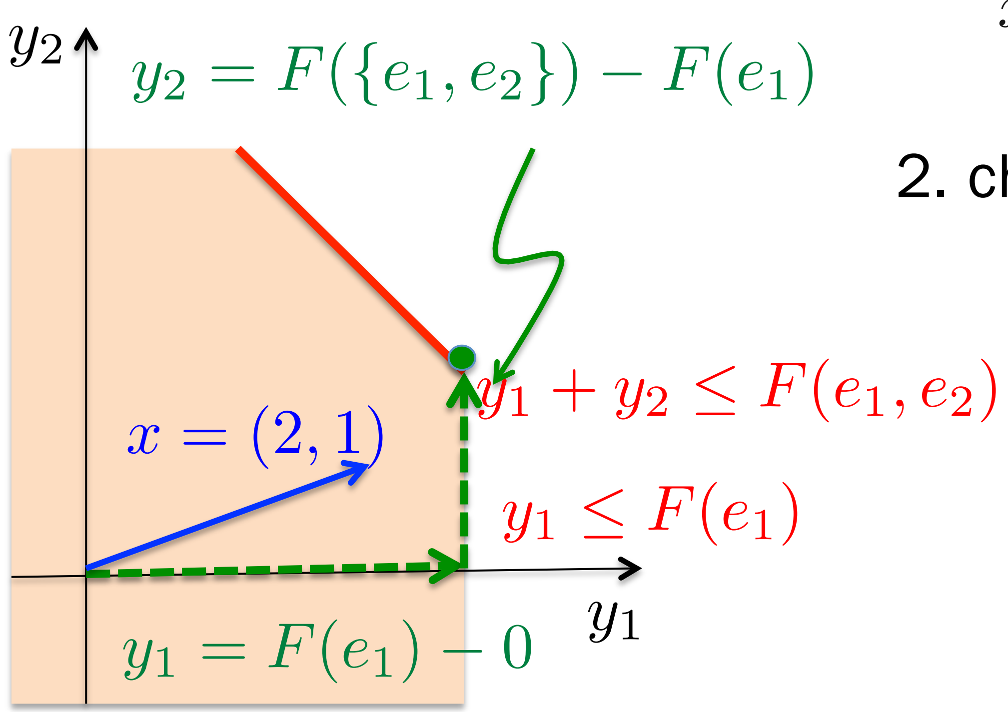
Edmonds 1970: "magic"
compute argmax in *O(n log n)* ☺

basis of (almost all) optimization!
-- separation oracle – subgradient --

Optimization over base polytope

$$\mathcal{B}_F = \Big\{ y \in \mathbb{R}^n \mid \sum_{a \in S} y_a \le F(S)$$
$$y(\mathcal{V}) = F(\mathcal{V}) \Big\}$$

$$\boxed{f(x) = \max_{y \in \mathcal{B}_F} \; y^\top x}$$



Edmonds' greedy algorithm:

1. sort

$$x_{\pi(1)} \ge x_{\pi(2)} \ge \ldots \ge x_{\pi(n)}$$

2. chain of sets

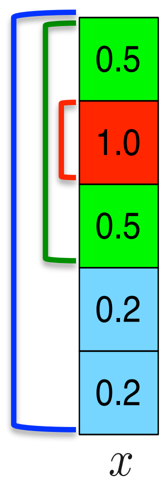
$$S_0 = \emptyset,$$
$$S_1 = \{\, \pi(1) \,\} \ldots$$
$$S_i = \{\, \pi(1), \ldots, \pi(i) \,\}$$

Base polytope

$$\boxed{f(x) = \max_{y \in \mathcal{B}_F} \; y^\top x}$$

Remarks:

- chain of sets same as before!
- y is a subgradient of f at x



1. sort

$$x_{\pi(1)} \geq x_{\pi(2)} \geq \ldots \geq x_{\pi(n)}$$

2. chain of sets

$$S_0 = \emptyset, \; S_i = \{\pi(1), \ldots, \pi(i)\}$$

3. assign values

$$y_{\pi(i)} = F(S_i) - F(S_{i-1})$$

$$\boxed{\sum_i \alpha_i F(S_i) = \sum_i (x_{\pi(i)} - x_{\pi(i-1)}) F(S_i) \quad = \sum_i y_{\pi(i)} x_{\pi(i)}}$$

Re-computing our examples

$$F(S) = \max\{|S|, 1\}$$

$$x = \begin{bmatrix} 0.5 \\ 1.0 \end{bmatrix} = 0.5 \begin{bmatrix} 1.0 \\ 1.0 \end{bmatrix} + 0.5 \begin{bmatrix} 0 \\ 1.0 \end{bmatrix}$$

sort: $x_2 \geq x_1 \quad \Rightarrow \quad S_1 = \{2\},\ S_2 = \{2, 1\}$

$$y_2 = F(2) = 1$$

$$y_1 = F(2,1) - F(2) = 1 - 1 = 0$$

$$f(x) = y^\top x = 1 \cdot x_1 + 0 \cdot x_2 = \max_i x_i$$

in general: $F(S_i) - F(S_{i-1}) > 0$ only for i=1!

$$\Rightarrow \quad f(x) = \max_i x_i$$

Re-computing our examples

$$x = \begin{bmatrix} 0.5 \\ 1.0 \end{bmatrix} = 0.5 \begin{bmatrix} 1.0 \\ 1.0 \end{bmatrix} + 0.5 \begin{bmatrix} 0 \\ 1.0 \end{bmatrix}$$

$$F(S) = \begin{cases} 1 & \text{if } S = \{1\},\ \{2\} \\ 0 & \text{if } S = \emptyset,\ \{1,2\} \end{cases}$$

sort: $x_2 \geq x_1 \quad \Rightarrow \quad S_1 = \{2\},\ S_2 = \{2,1\}$

$$y_2 = F(2) = 1$$

$$y_1 = F(2,1) - F(2) = 0 - 1 = -1$$

$$f(x) = y^\top x = -0.5 + 1 = |x_1 - x_2|$$

Back to our plan

✓ find a relaxation (extension): Lovasz extension

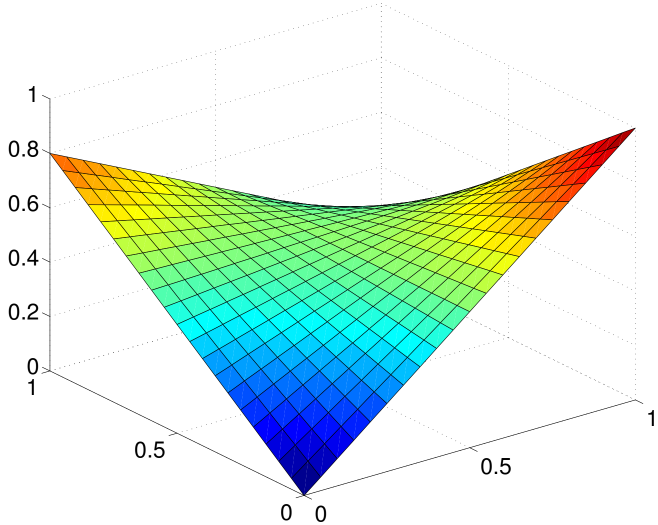
✓ magic of special polyhedra

➔ Lovasz extension is convex

- minimize Lovasz extension: up next
- get a set from solution

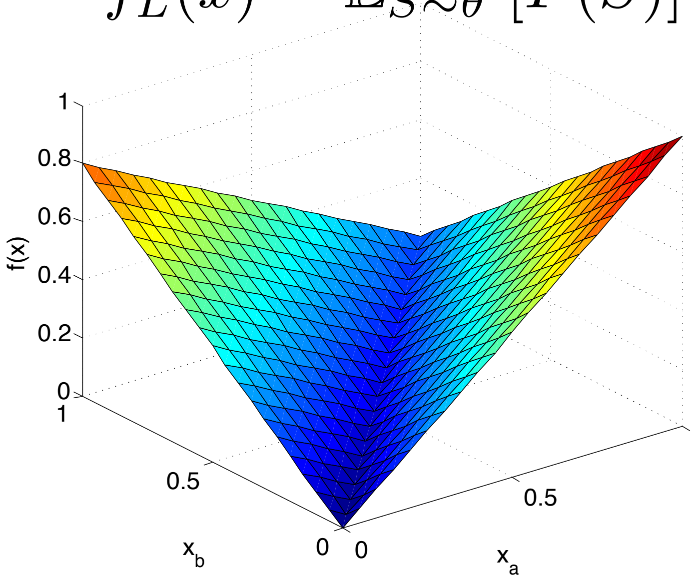
Multilinear relaxation vs. Lovász ext.

$$f_M(x) = \mathbb{E}_{S\sim x}\,[F(S)]$$



- concave in certain directions, convex in others
- approximate by sampling

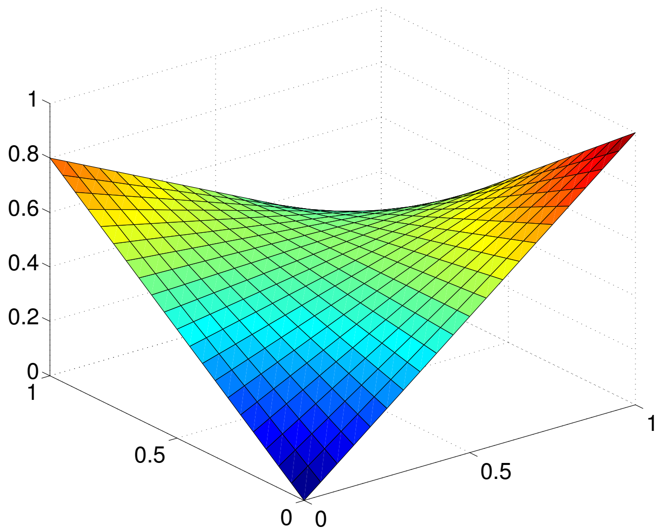
$$f_L(x) = \mathbb{E}_{S\sim\theta}\,[F(S)]$$

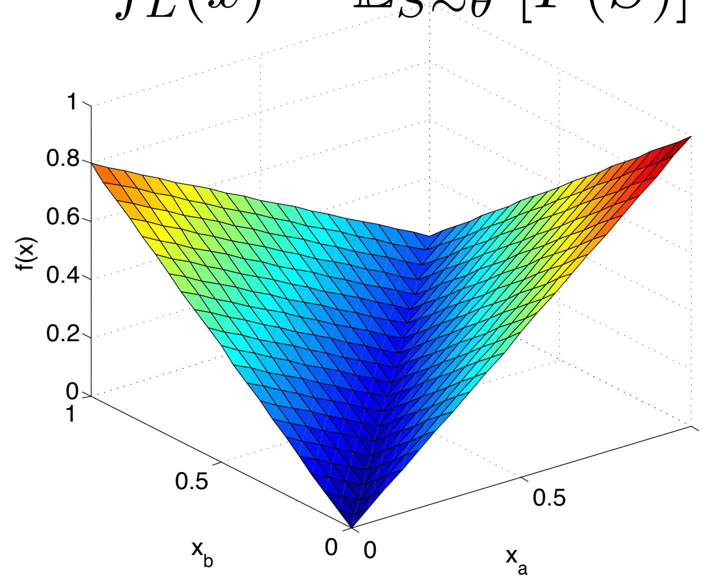


- convex
- computable in O(n log n)

Multilinear relaxation vs. Lovász ext.

$$f_M(x) = \mathbb{E}_{S\sim x}\left[F(S)\right] \qquad\qquad f_L(x) = \mathbb{E}_{S\sim\theta}\left[F(S)\right]$$





example: cut function

$$f_M(x) = x_u + x_v - 2x_u x_v \qquad\qquad f_L(x) = |x_u - x_v|$$

Back to our plan

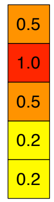
✓ find a relaxation (extension): Lovasz extension

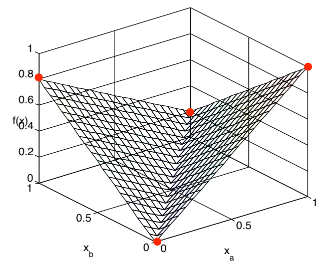
✓ magic of special polyhedra

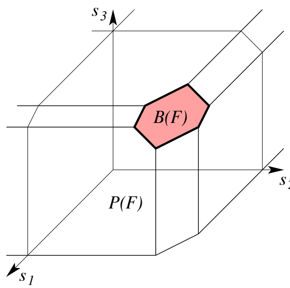
➔ Lovasz extension is convex

- minimize Lovasz extension: up next
- get a set from solution

Convex relaxation







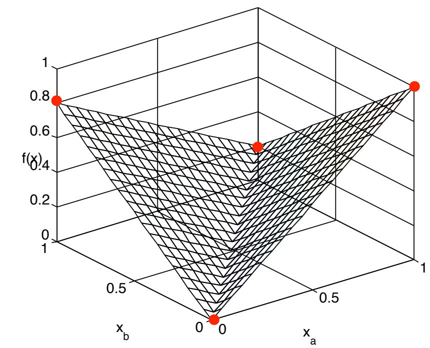
$$\min_{S \subseteq \mathcal{V}} F(S) \quad = \quad \min_{x \in \{0,1\}^n} F(x) \qquad \longrightarrow \qquad \min_{x \in [0,1]^n} f(x)$$

1. relaxation: convex optimization (non-smooth)

2. relaxation is exact!
   ➔ submodular minimization in polynomial time!
   (Grötschel, Lovász, Schrijver 1981)

Minimizing the Lovasz extension

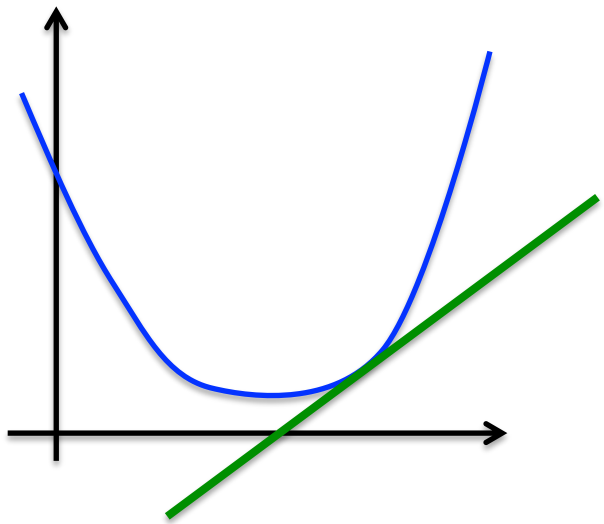
$$\min_{x \in [0,1]^n} \quad f(x)$$



- subgradient method

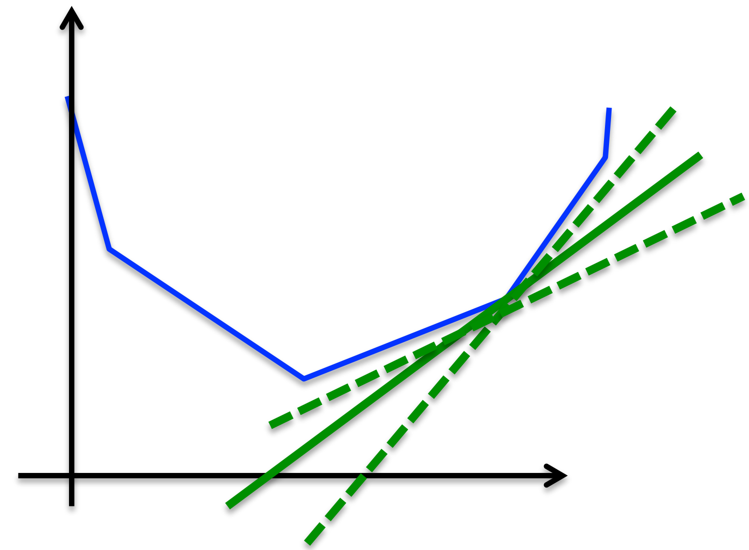
- combinatorial algorithms: dual

Subgradients



recall: gradient descent

$$x^{k+1} = x^k - \alpha \nabla f(x^k)$$



subgradient at x: vector g such that

$$\forall x' : \; f(x') \geq f(x) + \langle x' - x, g \rangle$$

subgradient of Lovasz extension: $g_x \in \arg\max_y \; y^\top x \quad \text{s.t. } y \in \mathcal{B}_F$

Projected subgradient method

$$\min_{x \in [0,1]^n} \quad f(x)$$

$$g^t = \arg\min_{y \in \mathcal{B}_F} \quad y^\top x^t$$

Edmonds' greedy algorithm ☺

$x^0 = 0$

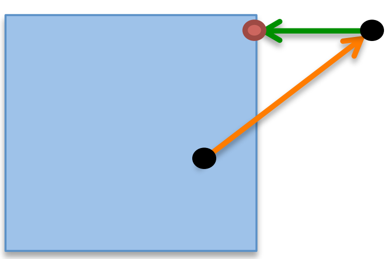
for $t = 0, \ldots$ **do**

  find $g^t \in \partial f(x^t)$

  $x^{t+1} = \Pi_{[0,1]^n}(x^t + \alpha_t g^t)$

end for

$$\Pi_{[0,1]^n}(y) = \arg\min_{z \in [0,1]^n} \|y - z\|^2$$



Convergence

Theorem

Let $D = \sqrt{n}$ and $L = \max_{g \in \mathcal{B}_F} \|g\| \leq 3 \max_S |F(S)|$.
With step size $\alpha_t = \frac{D}{L\sqrt{t}}$, the error decreases as

$$\min_{\tau \leq t} f(x^\tau) - f(x^*) \quad \leq \quad \frac{4DL}{\sqrt{t}}$$

- D: diameter of $[0,1]^n$
  L: Lipschitz constant
- for an error $\leq \epsilon$ need $O(\frac{1}{\epsilon^2})$ iterations

Submodular minimization

convex optimization

- ellipsoid method
  (Grötschel-Lovasz-Schrijver 81)
- subgradient method
- minimum-norm point / Fujishige-Wolfe algorithm
- ...

combinatorial methods

- first polynomial-time:
  (Schrijver 00, Iwata-Fleischer-Fujishige 01)
- $O(n^4 T + n^5 \log M)$ *(Iwata 03)*,
  $O(n^6 + n^5 T)$ *(Orlin 09)*

Latest result: $O(n^2 T \log nM + n^3 \log^c nM)$
$O(n^3 T \log^2 n + n^4 \log^c n)$ *(Lee-Sidford-Wong 15)*

Convex duality

$$\min_{S \subseteq \mathcal{V}} F(S) = \min_{x \in [0,1]^n} f(x)$$

$$= \min_{x \in [0,1]^n} \max_{y \in \mathcal{B}_F} y^\top x$$

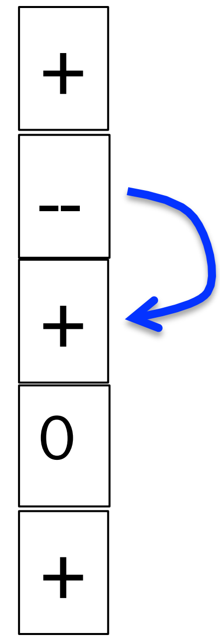
$$= \max_{y \in \mathcal{B}_F} \min_{x \in [0,1]^n} x^\top y \qquad = \max_{y \in \mathcal{B}_F} \Big(\sum_{i=1}^{n} \min\{y_i, 0\} \Big)$$

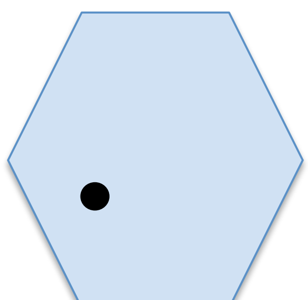
Optimality conditions: (S^*, y^*) optimal primal-dual pair if

1. $y^* \in \mathcal{B}_F$.
2. $\{y^* < 0\} \subseteq S^* \subseteq \{y^* \leq 0\}$.
3. $y^*(S^*) = F(S^*)$

Combinatorial algorithms

- solve $$\max_{y \in \mathcal{B}_F} \Big(\sum_{i=1}^{n} \min\{y_i, 0\} \Big)$$
- remove "negative mass"
- challenges:
  - need to stay in polytope
  - cannot test feasibility
  - ➔ network flow algorithms





Submodular minimization

convex optimization

- ellipsoid method
  (Grötschel-Lovasz-Schrijver 81)
- subgradient method
- minimum-norm point / Fujishige-Wolfe algorithm
- ...

combinatorial methods

- first polynomial-time:
  (Schrijver 00, Iwata-Fleischer-Fujishige 01)
- $O(n^4 T + n^5 \log M)$ *(Iwata 03)*,
  $O(n^6 + n^5 T)$ *(Orlin 09)*

Latest result: $O(n^2 T \log nM + n^3 \log^c nM)$
$O(n^3 T \log^2 n + n^4 \log^c n)$ *(Lee-Sidford-Wong 15)*

Proximal problem

proximal problem

Lovász extension

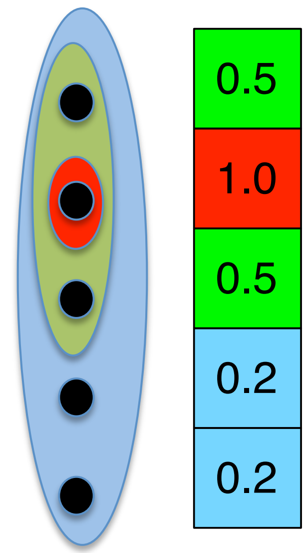
$$\min_{x \in [0,1]^n} f(x) + \frac{1}{2}\|x\|^2$$

why? solves

$$\min_{S \subseteq \mathcal{V}} F(S) + \alpha|S| \quad \text{for all } \alpha$$

- Let S^α be the largest minimizer of $F(S) + \alpha|S|$
- can show: if $\alpha < \beta$, then $S^\alpha \supseteq S^\beta$
  ➔ chain $\emptyset \subset S^{\alpha_1} \subset S^{\alpha_2} \subset \ldots \mathcal{V}$
- "encode" in vector u :

$$\{e \mid u_e \geq \alpha\} = S^\alpha$$



$$\boxed{u = \arg\min_x f(x) + \tfrac{1}{2}\|x\|^2}$$

3 equivalent problems

$$y^* = -x^*$$

projection

$$\min_{y \in \mathcal{B}_F} \ \frac{1}{2}\|y\|^2$$

convex dual problem

proximal

$$\min_x \ f(x) + \frac{1}{2}\|x\|^2$$

parametric

$$\min_S \ F(S) + \alpha|S|$$

$$S^\alpha = \{e \mid x_e^* \geq \alpha\}$$

thresholding

$$x_e^* = \sup\{\alpha \mid e \in S^\alpha\}$$

divide-and-conquer

(Topkis 1978, 1998; Granot-Veinott 1985; Hochbaum 01; Nagano 2007; Fujishige & Isotani 11; …)

Minimum-norm-point algorithm

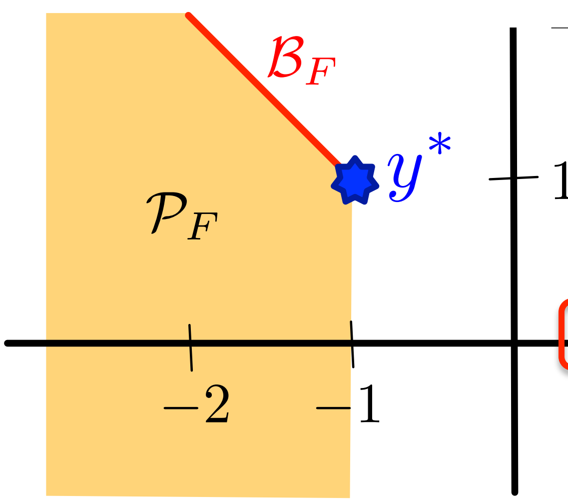
proximal problem

$$\min_{x} \; f(x) + \frac{1}{2}\|x\|^2$$

dual: minimum norm problem

$$y^* = \arg\min_{y \in \mathcal{B}_F} \; \frac{1}{2}\|y\|^2$$

$$y^* = \begin{array}{cc} \begin{bmatrix} -1 \\ 1 \end{bmatrix} & \begin{array}{c} a \\ b \end{array} \end{array}$$



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

$$S^* = \{i \mid y_i^* \leq 0\}$$

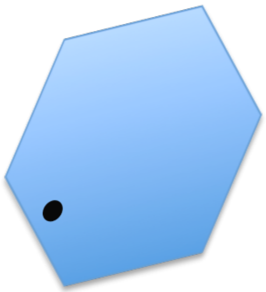
minimizes *F* !

$$S^* = \arg\min_{S \subseteq \mathcal{V}} \; F(S)$$

(Fujishige '91, Fujishige & Isotani '11)

Solving the min-norm problem

$$\min_{y \in \mathcal{B}_F} \quad \frac{1}{2}\|y\|^2$$



- costly: testing membership in $\mathcal{B}_F$
- costly: projection onto $\mathcal{B}_F$
- easy: linear optimization over $\mathcal{B}_F$: greedy algorithm! ☺

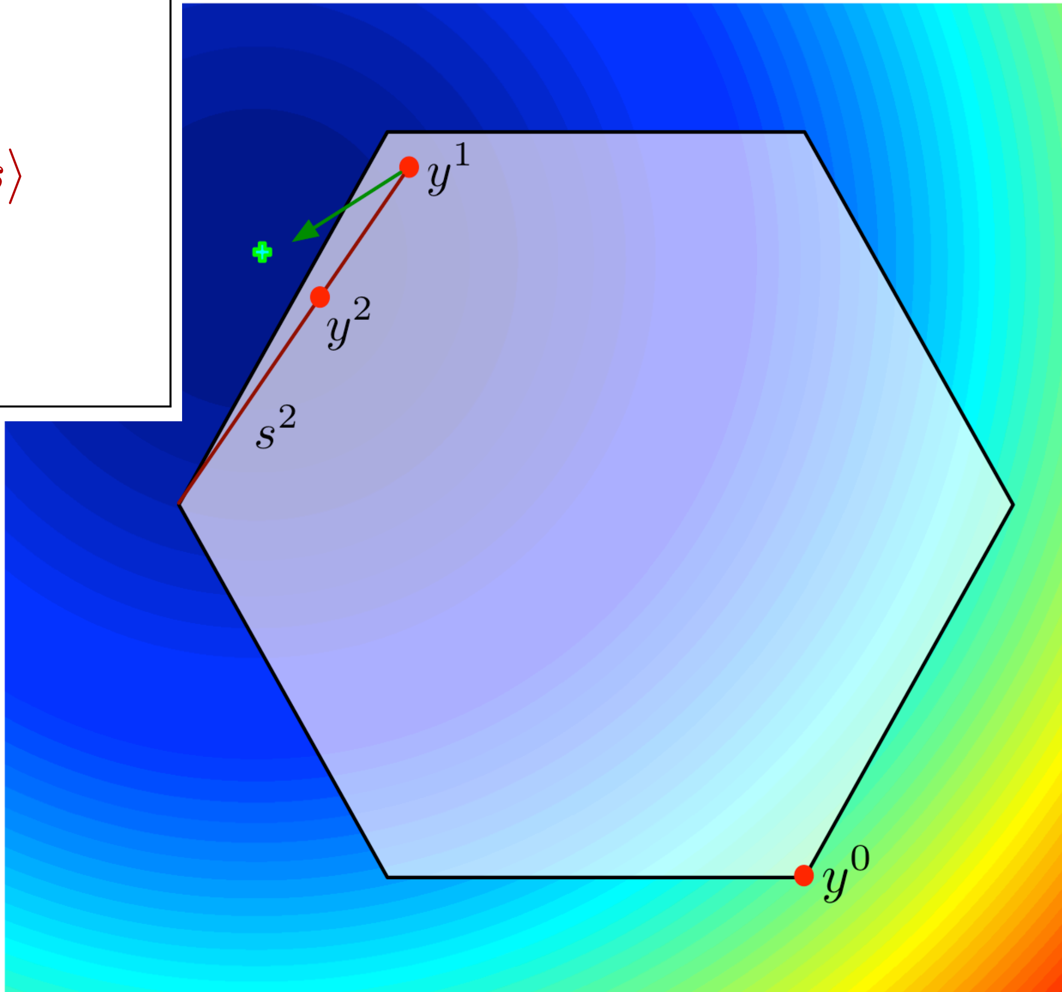
- conditional gradient (Frank-Wolfe) algorithm *(Frank & Wolfe 1956)*
- active set methods: Fujishige-Wolfe *(Fujishige & Isotani 2011, Chakrabarty-Jain-Kothari 2014,…)*

Frank-Wolfe algorithm

$$\min\ h(y) \quad \text{s.t. } y \in \mathcal{P}$$

h convex, differentiable
P polyhedral

$y^0 \in \mathcal{P}$
for $t = 0, 1, \dots$ to T **do**
$\quad s^t = \text{argmin}_{s \in \mathcal{P}} \langle \nabla h(y^t),\ s \rangle$
$\quad y^{t+1} = (1-\gamma)y^t + \gamma s^t$
end for



step size?
how many iterations?

Frank-Wolfe algorithm: step sizes

$$\min \ h(y) \quad \text{s.t. } y \in \mathcal{P}$$

h convex, differentiable
P polyhedral

for $t = 0, 1, \ldots$ to T **do**
$s^t = \text{argmin}_{s \in \mathcal{P}} \langle \nabla h(y^t), s \rangle$
$y^{t+1} = (1-\gamma) y^t + \gamma s^t$
end for

1. fixed step size: $\gamma^t = \frac{2}{t+2}$

2. line search: $\gamma^t = \arg \min_{\gamma \in [0,1]} h(y^t + \gamma(s^t - y^t))$

3. re-optimization: $y^{t+1} = \text{argmin}_{y \in \text{conv}(s^0, \ldots, s^t)} h(y)$

How many iterations?

We are solving

dual problem

$$\max_{y \in \mathcal{B}_F} -\tfrac{1}{2}\|y\|^2$$

We want

discrete primal

$$\min_{S \subseteq \mathcal{V}} F(S)$$

Theorem *(Jaggi 2013, Bach 2013)*

1. **relaxation:** After *T* iterations of Frank-Wolfe, have an iterate y^T with
$$\text{gap}(y^T) \leq \frac{16C}{T+2}$$

2. **discrete:** after *T* iterations of Frank-Wolfe, can get a set S with
$$\text{gap}(S) \leq \sqrt{2n\,\text{gap}(y^T)} \quad = O(\sqrt{n/T})$$

cf subgradient method

How many iterations? (Details)

$$y^* = -x^*$$

dual problem	primal problem
$\max_{y \in \mathcal{B}_F} -\frac{1}{2}\|y\|^2$	$\min_{x} \quad f(x) + \frac{1}{2}\|x\|^2$

always: primal value ≥ dual value

at optimum: primal value = dual value

➔ bound the duality gap: primal – dual value

Theorem *(Jaggi 2013)*

After *T* iterations of the algorithm, there is an iterate y^T with

$$\mathrm{gap}(y^T) \leq \frac{16C}{T+2}$$

For $h(y) = \|y\|^2, \; C = \mathrm{diam}(\mathcal{P})^2$

Exercise: how many iterations for a gap $\leq \epsilon$?

Are we done yet? (Details)

want: $\min_{S \subseteq \mathcal{V}} \quad F(S)$ - - - - - - - - - → $\max_{y \in \mathcal{B}_F} -\frac{1}{2}\|y\|^2$

$S^t_\alpha = \{e \mid -y^t_e \geq \alpha\}$ ← - - - - - - - - - solve via conditional gradient & friends

have convergence bound

How many iterations until $F(S^t) - F(S^*) \leq \varepsilon$?

Theorem *(Bach 2013)*

If $\quad \mathrm{gap}(y^\tau) \leq \epsilon'$ then there exists an α such that the discrete duality gap is bounded as

$$F(S^\tau_\alpha) - F(S^*) \leq \quad F(S^\tau_\alpha) - y^\tau_-(\mathcal{V}) \leq \sqrt{2n\epsilon'}$$

Summary: formulations for min

$$\min_{S \subseteq \mathcal{V}} F(S) \;=\; \min_{x \in [0,1]^n} f(x)$$

subgradient descent

primal: take positive coordinates

$$\min_{x} \quad f(x) + \tfrac{1}{2}\|x\|^2$$

subgradient descent

$$\max_{y \in \mathcal{B}_F} \sum_{i=1}^{n} \max\{y_i, 0\}$$

combinatorial

dual: take negative coordinates

$$\max_{y \in \mathcal{B}_F} -\tfrac{1}{2}\|y\|^2$$

Frank-Wolfe
or
Fujishige-Wolfe (faster)

- usually fastest in practice: usually Fujishige-Wolfe
- alternative algorithms for special cases: symmetric functions, cuts, ...

Connections

$$\min_x \quad f(x) + \tfrac{1}{2}\|x\|^2$$

subgradient descent

$$\max_{y \in \mathcal{B}_F} -\tfrac{1}{2}\|y\|^2$$

conditional gradient

subgradient:

$$g^t = \underbrace{\arg\max_{s \in \mathcal{B}_F} \langle s, x^t \rangle} \quad + x^t$$

$$x^{t+1} = x^t - \alpha g^t$$

$$= (1-\alpha)x^t - \alpha s^t$$

direction:

$$s^t = \arg\min_{s \in \mathcal{B}_F} \langle s, y^t \rangle$$

same if $x^t = -y^t$

$$y^{t+1} = (1-\gamma)y^t + \gamma s^t$$

now recall: $x^* = -y^*$

Submodularity and convexity

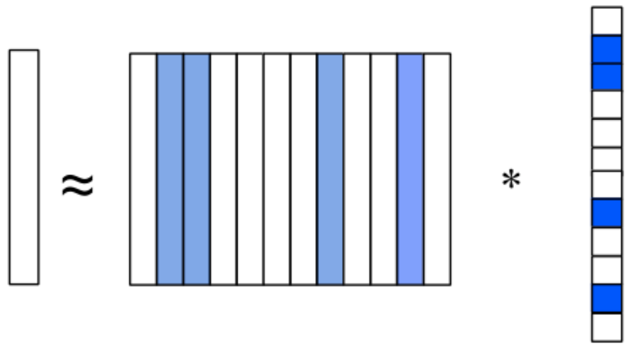
- convex Lovasz extension
  - easy to compute due to greedy algorithm (special polyhedra!)

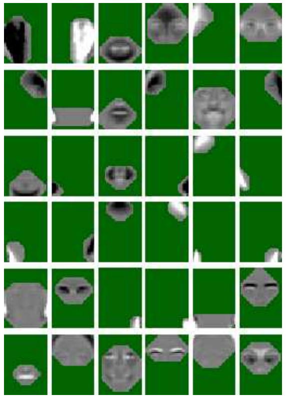
- submodular minimization via convex optimization: fast algorithms for many applications
- duality results

- structured sparsity *(Bach 2010)*
- decomposition & parallel algorithms *(Jegelka et al 2013, Nishihara et al 2014, Ene & Nguyen 2015)*
- variational inference *(Djolonga & Krause 2014)*
- …

Structured sparsity and submodularity

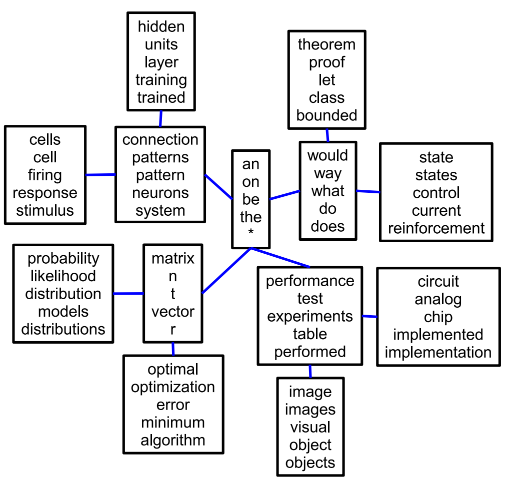
$$y = M\textcolor{red}{x} + \text{noise}$$





Structured sparse PCA

$$\min_x \; \|y - Mx\|^2$$



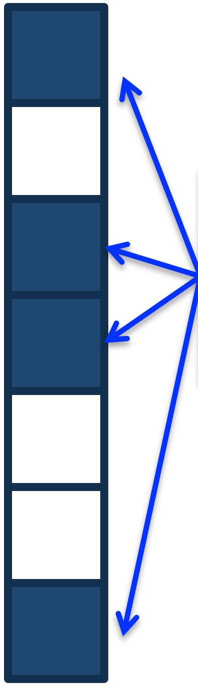
Sparse reconstruction

$$\min_x \|y - Mx\|^2 \quad + \lambda\Omega(x)$$

Assumption:
x is sparse

discrete regularization on support *S* of *x*

$$\Omega(x) = \|x\|_0 \quad = |S|$$



subset selection:
S = {1,3,4,7}

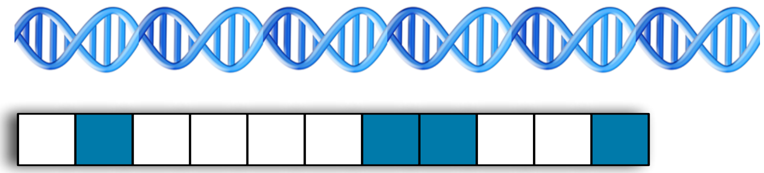
relax to convex envelope

$$\Omega(x) = \|x\|_1$$

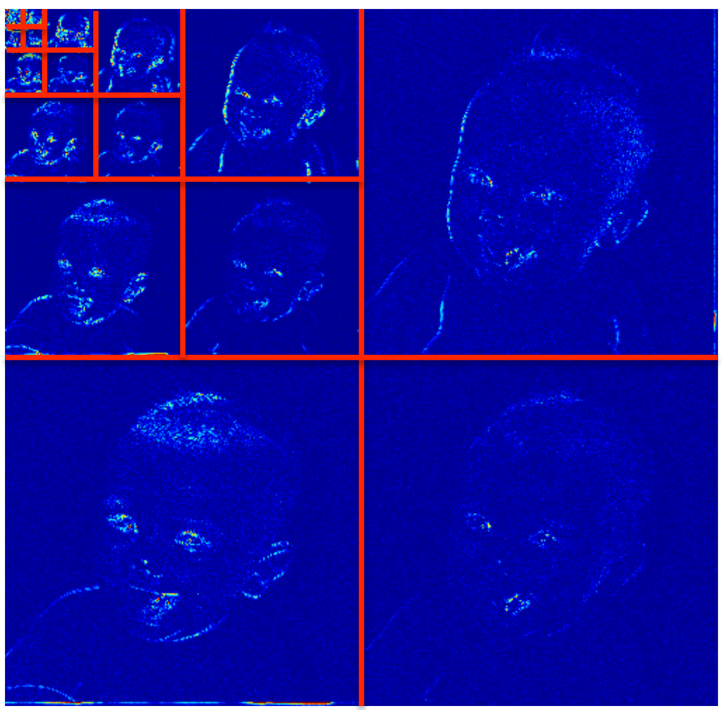
sparsity pattern often not random…

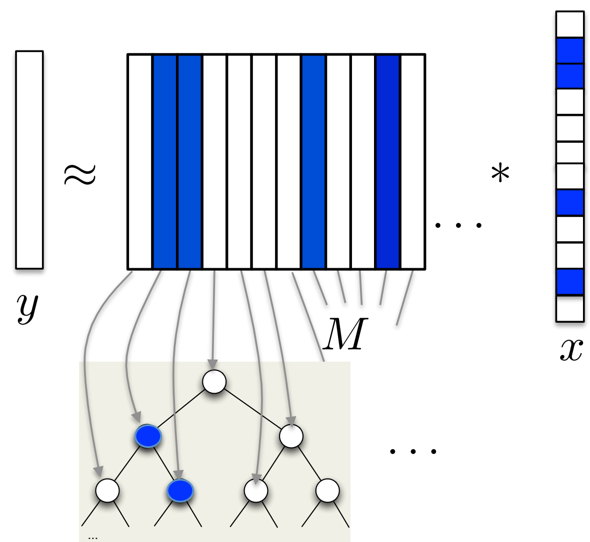
Structured sparsity

Assumption:
support of x
has structure



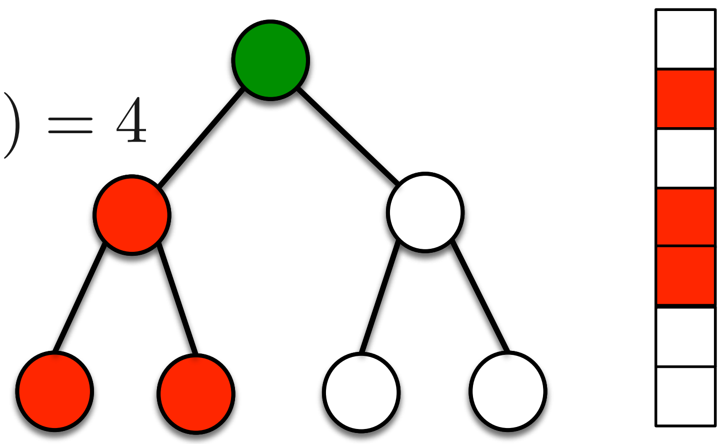
express by set function!



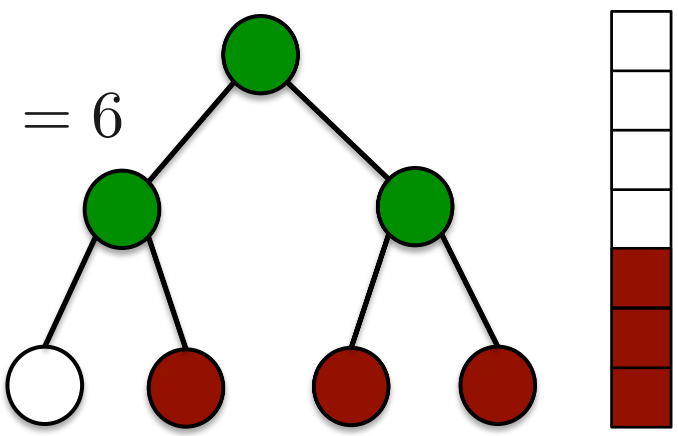


Preference for trees

$F(T) = 4$



$F(S) = 6$



Set function:

$$F(T) < F(S)$$

if T is a tree and S not

|S| = |T|

$$F(S) = \left| \bigcup_{s \in S} \text{ancestors}(s) \right|$$

use as regularizer?

Sparsity

$$\min_x \ \|y - Mx\|^2 \ + \lambda\Omega(x)$$

- x sparse
- x structured sparse

discrete regularization on support S of x

submodular function

$$\Omega(x) = \|x\|_0 \quad = |S|$$

$$\Omega(x) = F(S)$$

relax to convex envelope

➔ Lovász extension

$$\Omega(x) = \|x\|_1$$

$$\Omega(x) = f(|x|)$$

Optimization: submodular minimization (min-norm)

(Bach2010)

Norms from submodular functions

$$\Omega(x) = f(|x|)$$

Proposition

For monotone increasing *f*, $f(|x|)$ is a norm.

(Exercise: show this)

Special cases: $F(S) = |S| \quad \Rightarrow \quad f(|x|) = \|x\|_1$

$$F(S) = \min\{|S|, 1\} \quad \Rightarrow \quad f(|x|) = \|x\|_\infty$$

$$F(S) = \sum_{j=1}^{k} \min\{|S \cap G_j|, 1\} \quad \Rightarrow \quad f(|x|) = \sum_{j=1}^{k} \|x_{G_j}\|_\infty$$

Special case

- minimize a sum of submodular functions

$$F(S) = \sum_{i=1}^{r} F_i(S)$$

$$\min_S \; F_i(S)$$

"easy"

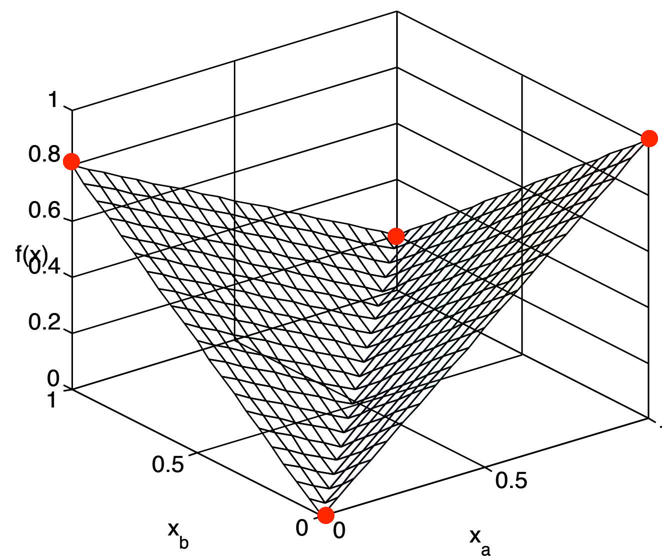
- combinatorial algorithms
  (Kolmogorov 12, Fix-Joachims-Park-Zabih 13, Fix-Wang-Zabih 14)

- convex relaxations

Relaxation

$$F: 2^{\mathcal{V}} \to \mathbb{R} \quad \equiv \quad F: \{0,1\}^n \to \mathbb{R} \qquad\qquad f: \mathbb{R}^n_+ \to \mathbb{R}$$

- convex Lovász extension: tight relaxation



$$\min_{S\subseteq\mathcal{V}} \sum_i F_i(S) \quad = \quad \min_{x\in[0,1]^n} \sum_i f_i(x)$$

$$\min_x \sum_{i=1}^r f_i(x) + \tfrac{1}{2}\|x\|^2$$

- dual decomposition: parallel algorithms

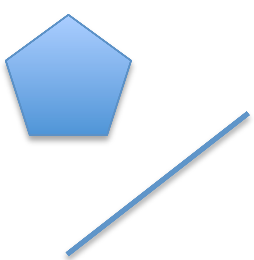
(Komodakis-Paragios-Tziritas 11, Savchynskyy-Schmidt-Kappes-Schnörr 11, J-Bach-Sra 13)

Results: dual decomposition

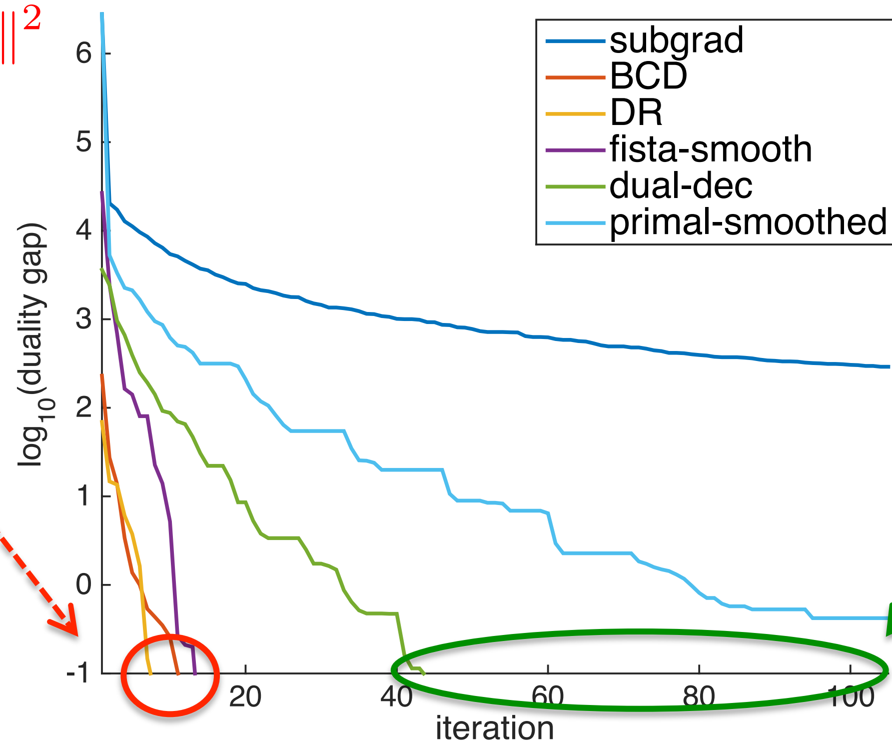
relax II

$$\min_x \sum_i f_i(x) + \frac{1}{2}\|x\|^2$$

smooth dual



convergence discrete problem



relaxation I

$$\min_{x \in [0,1]^n} \sum_i f_i(x)$$

non-smooth dual

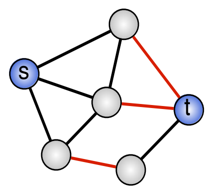
faster
parallel
algorithms ☺

(Jegelka, Bach, Sra 2013; Nishihara, Jegelka, Jordan 2014)

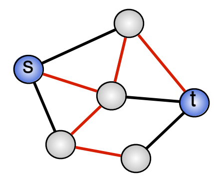
What if ... ?

$$\min F(S) \text{ s.t. constraints on } S$$

e.g.

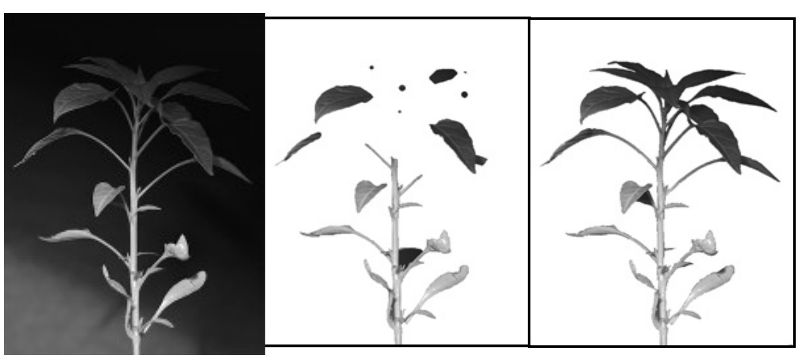


S is a cut



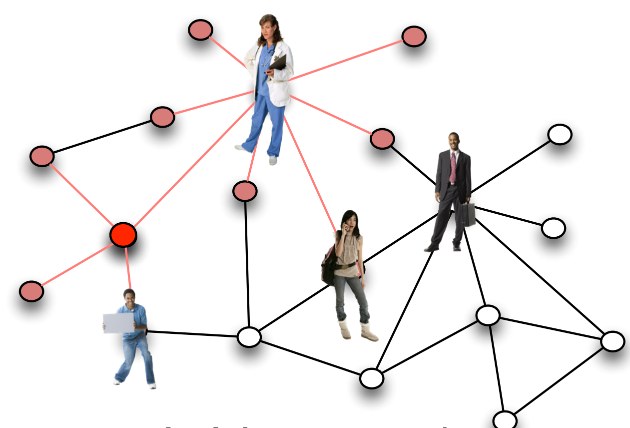
S is a spanning tree

$$|S| = k$$



segmentation

(Jegelka, Bilmes CVPR 2011)



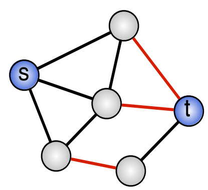
optimizing networks

(Khalil, Dilkina, Song KDD 2014)

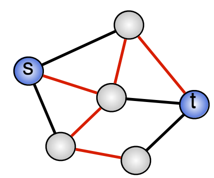
Constrained minimization

$$\min F(S) \text{ s.t. constraints on } S$$

e.g.



S is a cut



S is a spanning tree

$$|S| = k$$

in most cases very hard. ➔ approximations

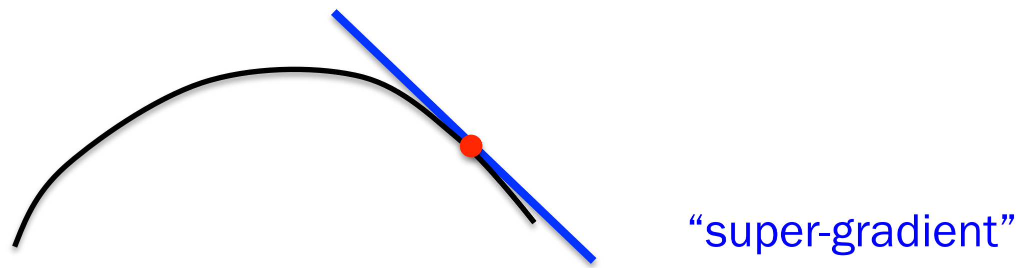
convex relaxation
(not exact!)

approximate *F*

(Goel et al. 09, Iwata & Nagano 09, Goemans et al. 09, Jegelka & Bilmes 11, Iyer et al. ICML 13, Kohli et al 13...)

A practical algorithm

idea: submodularity = discrete concavity



For i = 1,2,...

- compute linear upper bound $\widehat{F}_i$ with $\widehat{F}_i(C_i) = F(C_i)$
- find tree/path/... C_{i+1} with minimum $\widehat{F}_i(C)$.

familiar ☺
e.g. min-cut

fast: only need to solve linear optimization problem!

(Jegelka & Bilmes 2011; Iyer, Jegelka, Bilmes 2013)

Different approximation of *F*

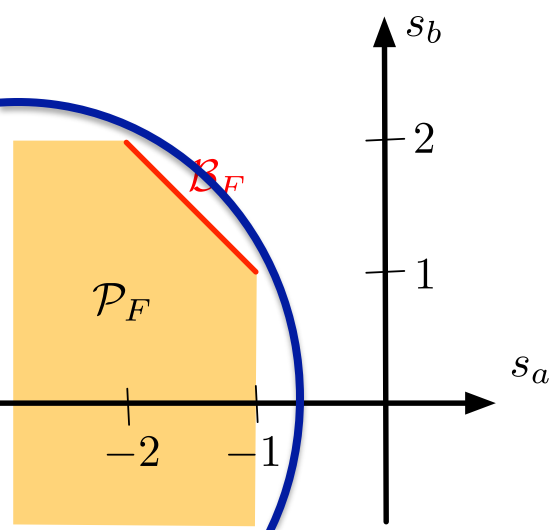
- recall: for $x = 1_S$

$$F(S) = f(x) = \max_{y \in \mathcal{B}_F} y^\top x$$

$$\widehat{F}(S) = \max_{y \in \mathcal{E}_F} y^\top x$$

$$= \sqrt{\sum_{e \in S} w(e)}$$

approximate polyhedron by ellipsoid

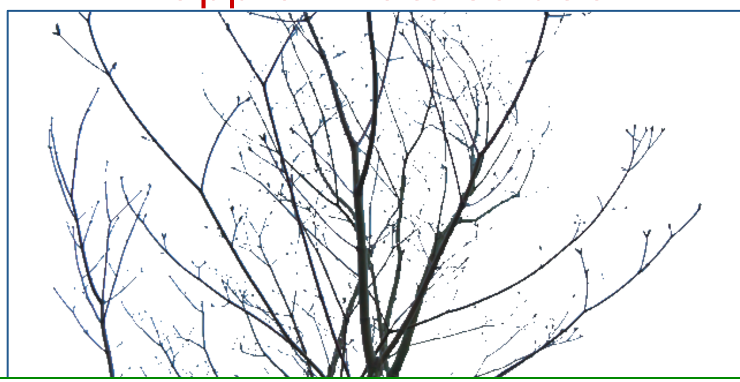


(Goemans et al 2009)

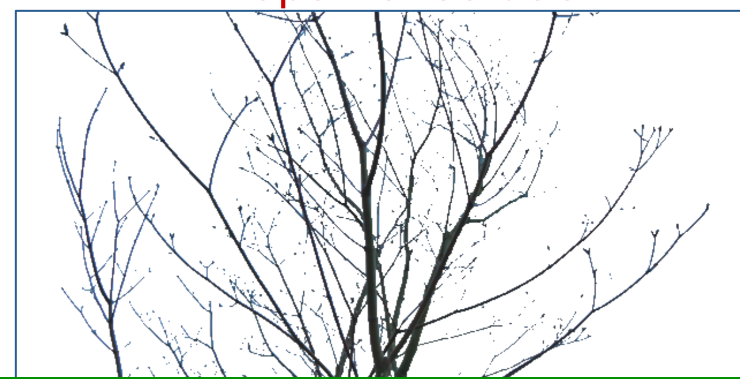
Does it work?

minimum cut solution

approximate solution



optimal solution



- often works well in practice
- theory: approximation guarantees depending on curvature of *F*

  (Iyer et al 2013)
- special cases: exact solution *(Kohli et al 2013)*

Submodular Optimization in a nutshell

Maximization (NP-hard)

- greedy algorithms:
  exploit discrete concavity = diminishing returns
  - accommodate many constraint types
  - scalable algorithms being developed

Minimization (poly-time if unconstrained)

- convex optimization
  exploit ‘discrete’ convexity: polyhedra, convex extension
  - constraints are hard in the worst case. Use majorize-minimize, relaxations or approximations of F

Other recent & ongoing developments

- Faster maximization (streaming, …)
- Faster minimization
- Online submodular optimization
- Beyond binary: integer-submodular functions

- Learning a submodular function

- Profiting from submodularity in distributions defined by submodular functions

Submodularity and machine learning

distributions over labels, sets
log-submodular/
supermodular probability
e.g. "attractive" graphical models,
determinantal point processes

submodularity
& machine
learning!

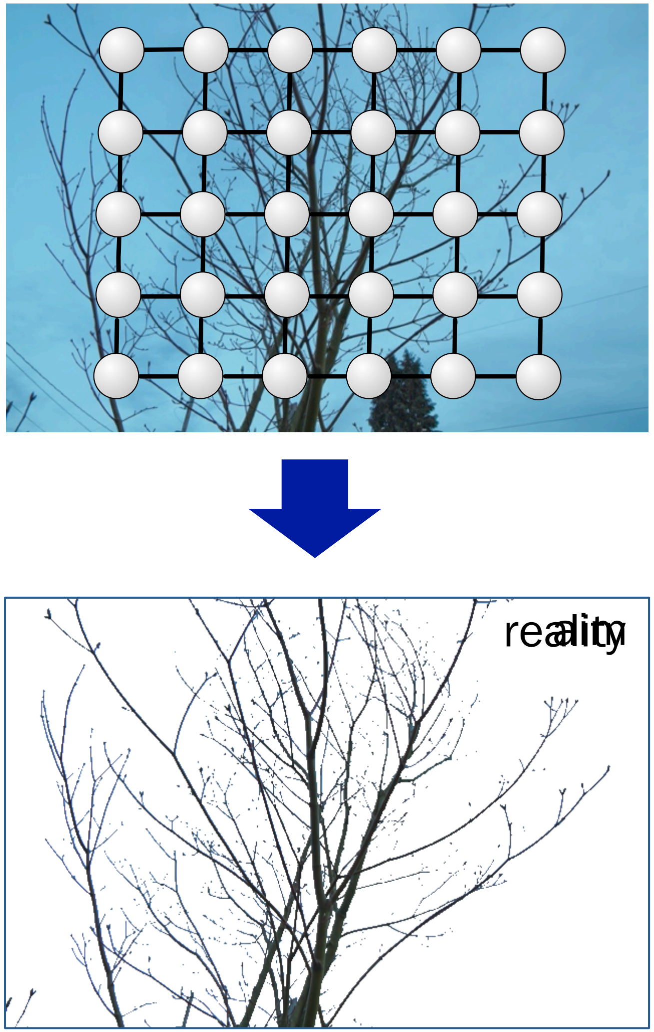
diffusion processes,
covering, rank,
connectivity,
entropy,
economies of scale,
summarization, …
submodular
phenomena

(convex) regularization
submodularity: "discrete
convexity"
e.g. combinatorial sparse estimation

More on constrained minimization

the following slides are extra material on constrained minimization.

Recall: MAP and cuts

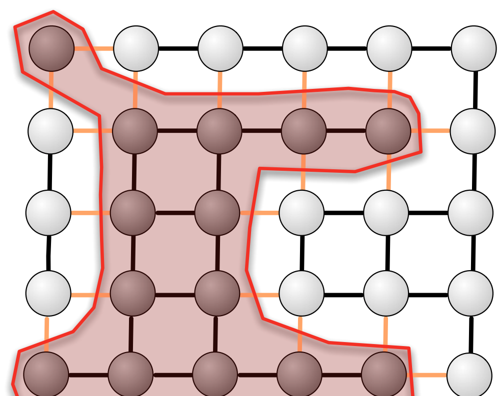


binary labeling: $x = 1_A$

pairwise random field:

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

What's the problem?



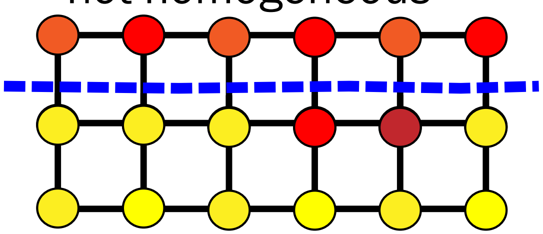
minimum cut: prefer
short cut = short object boundary

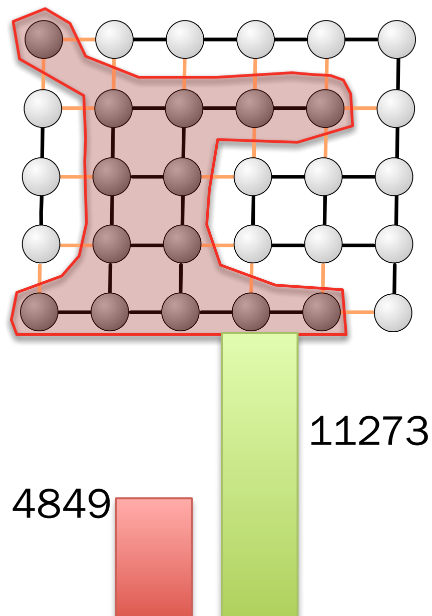
What’s wrong?

we get ...

local coherence
= short cut

not homogeneous



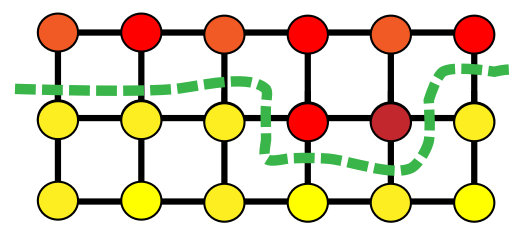


cut weight
= energy

ideally ...

homogeneous cut
global dependencies!

homogeneous



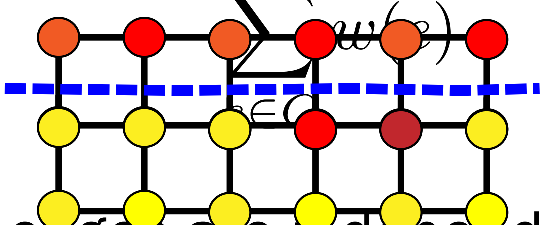
Cooperative cuts

ideally …

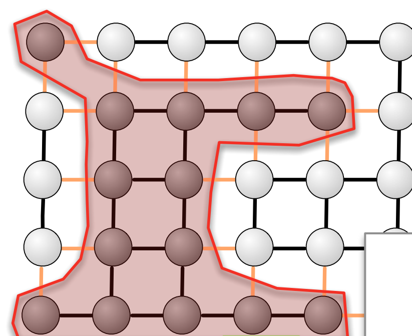
local coherence
= short cut

cost of a cut $C \subseteq \mathcal{E}$:

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



edges are independent



homogeneous cut
global dependencies!

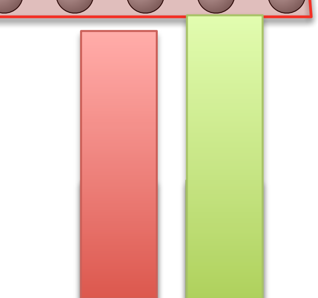
cooperative graph cut

cost of a cut $C \subseteq \mathcal{E}$:

submodular function

$$F(C)$$

edges are not independent



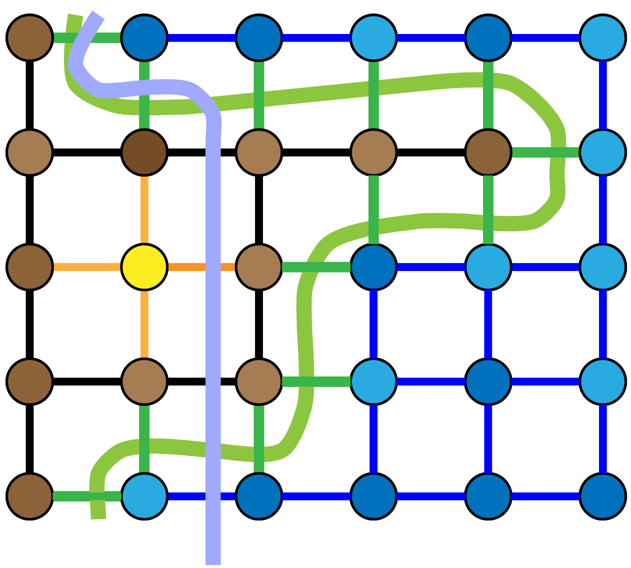
cut weight
= energy

Homogeneity via group sparsity

sum of weights:
use few edges

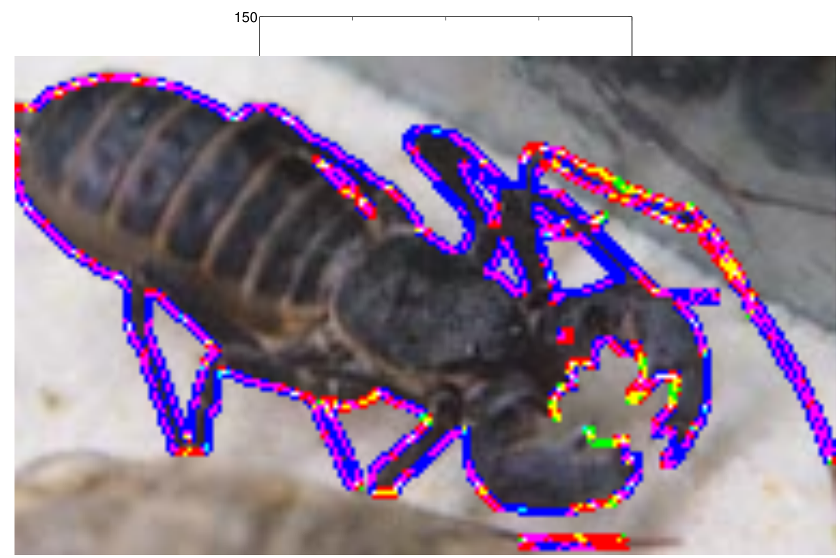
→

submodular cost function:
use few types of edges



One type (13 edges)
Many types (6 edges)

$$F(\mathrm{Cut}) = \sum_{\text{type } k} F_k(\mathrm{Cut})$$



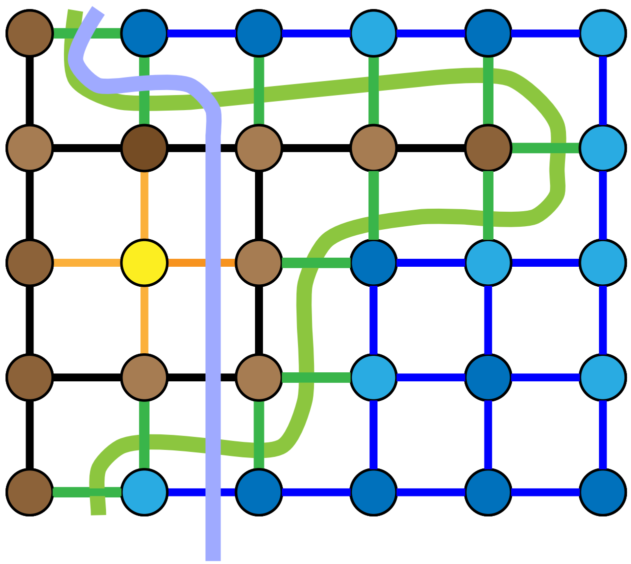
(Jegelka & Bilmes 2011)

Homogeneity via group sparsity

sum of weights:
use few edges

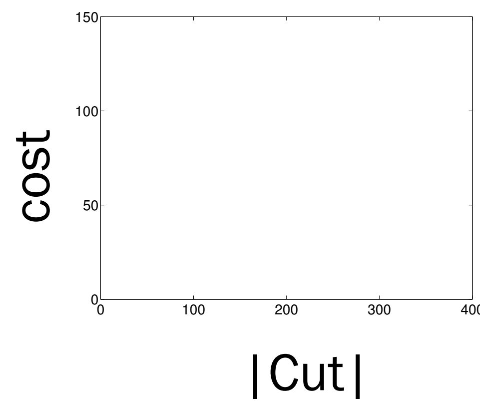
→

submodular cost function:
use few types of edges



One type (13 edges)
Many types (6 edges)

$$F(\mathrm{Cut}) = \sum_{\text{type } k} F_k(\mathrm{Cut})$$



(Jegelka & Bilmes 2011)

Results

Random Walker
[Grady 06]

Curvature Reg.
[El-Zehiry & Grady 10]

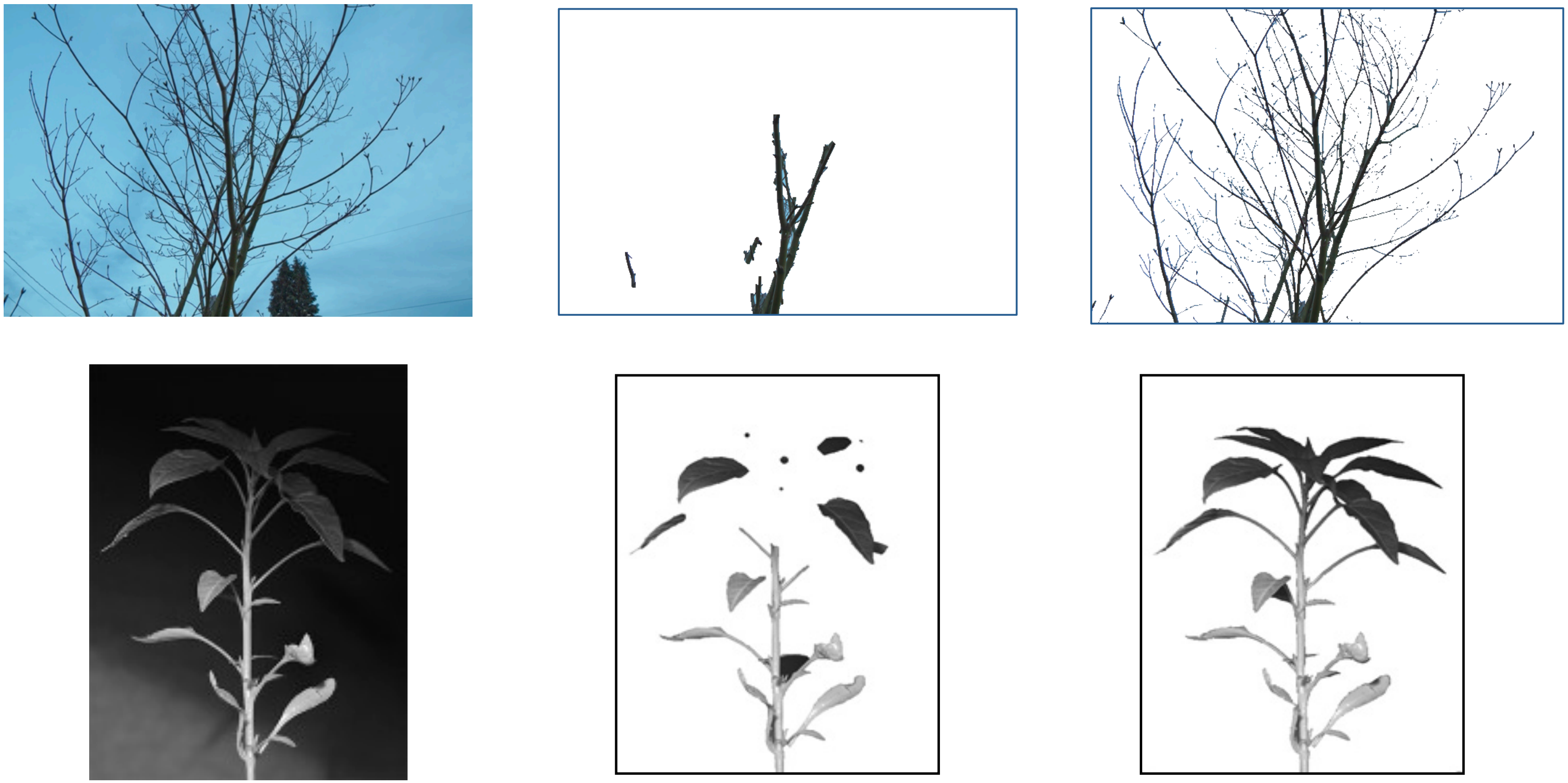
Graph Cut
[Boykov & Jolly 01]

Cooperative Cut

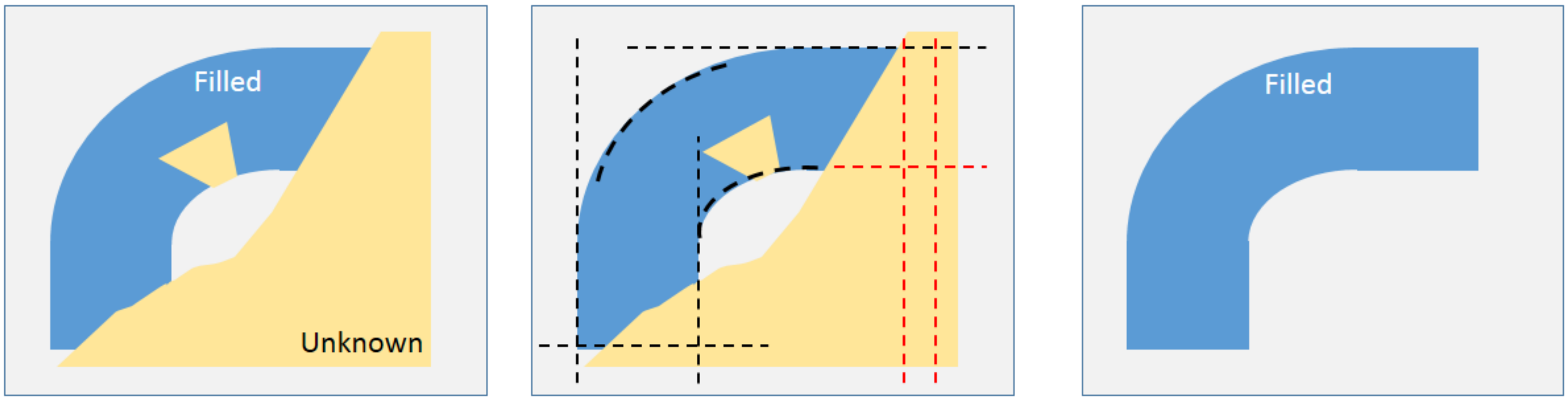
Quantitatively: up to 70% reduction in error!

Results

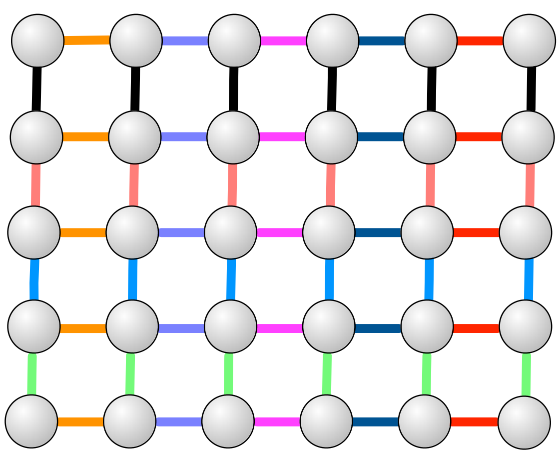
Graph cut | Cooperative cut



Similarly: contour completion RF



(a) (b) (c)

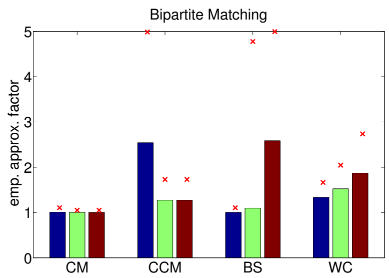


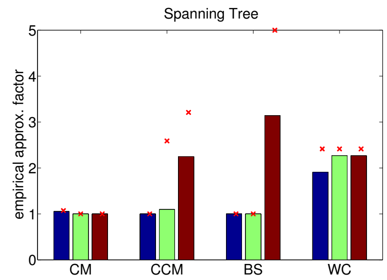
geometric edge groups:

- straight lines
- parabolas

(Silberman et al 2014)

Theory and practice

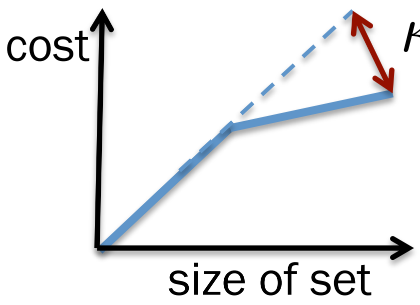




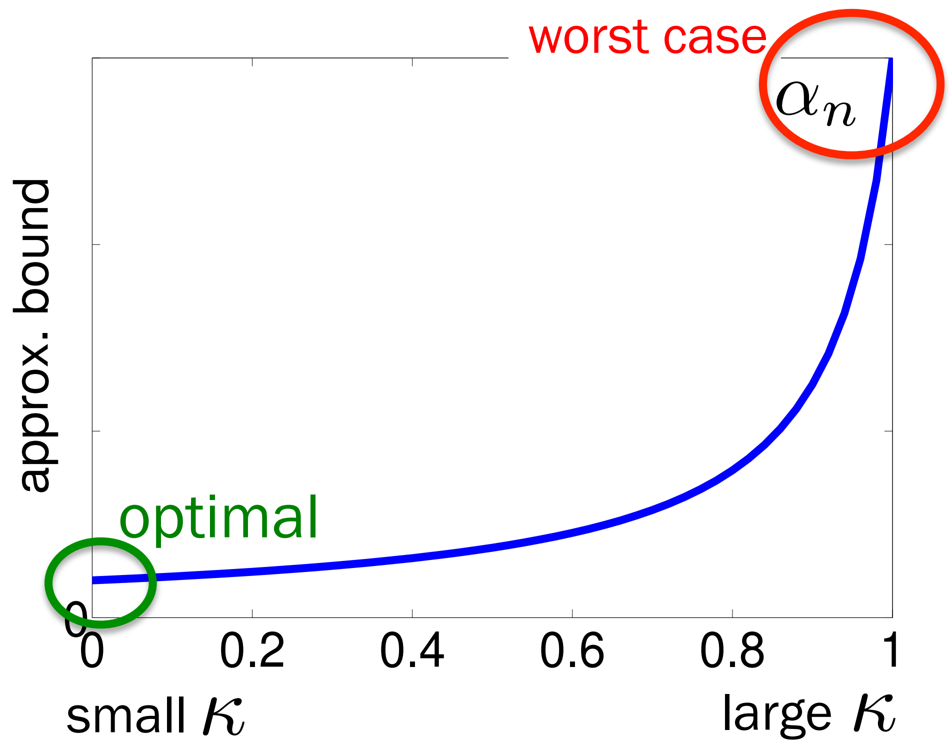
Good approximations in practice ☺ .... BUT not in theory?

What makes some (practical) problems easier than others?

Instance-dependent analysis



κ = "deviation from linearity"



Theorem *(IJB 2013)*.
Tightened upper & lower bounds for constrained minimization, approximation, learning:

$$\frac{\alpha_n}{1 + (\alpha_n - 1)(1 - \kappa)} \leq \frac{1}{1 - \kappa}$$

(Iyer, Jegelka, Bilmes 2013)