Submodularity in Machine Learning

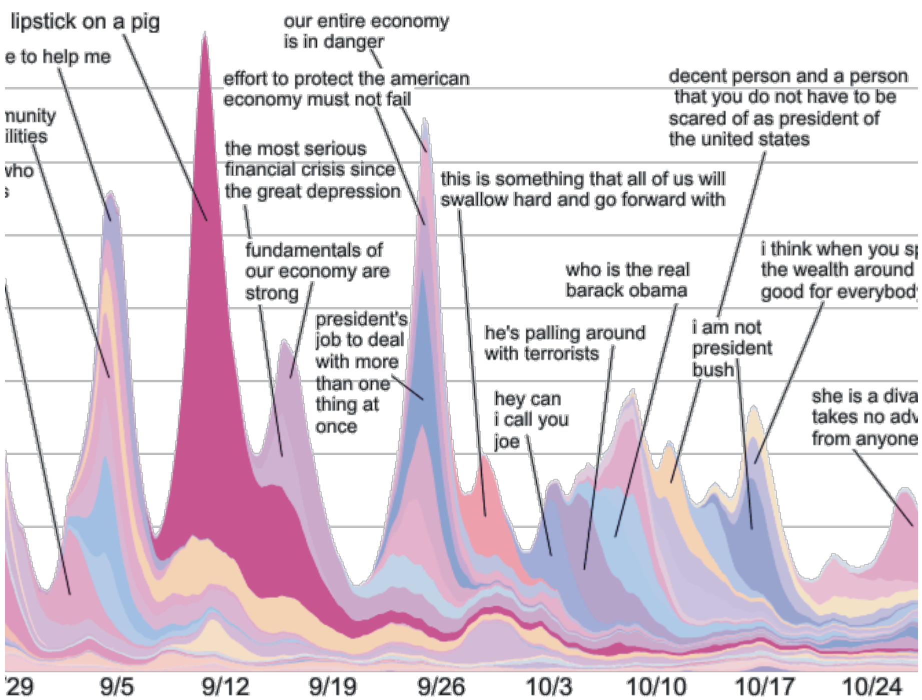
- New Directions -

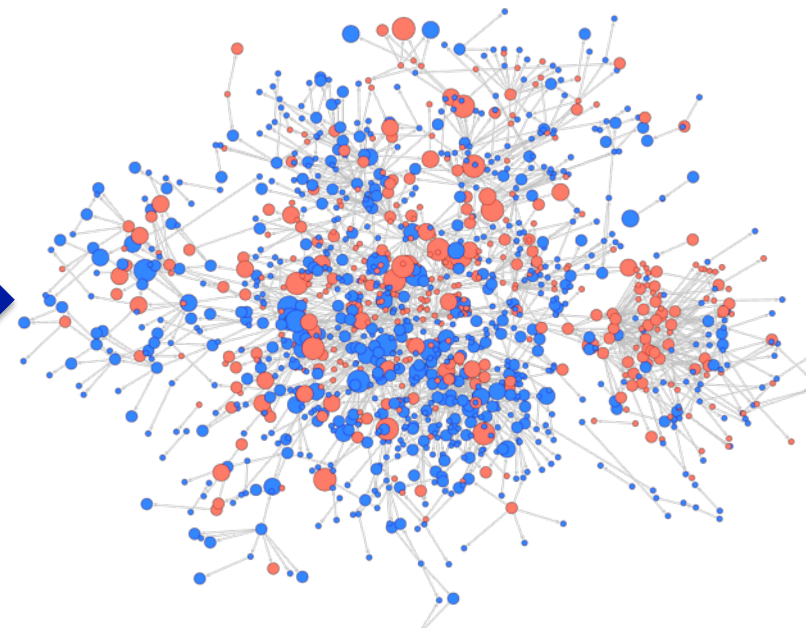
Andreas Krause

Stefanie Jegelka

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Network Inference



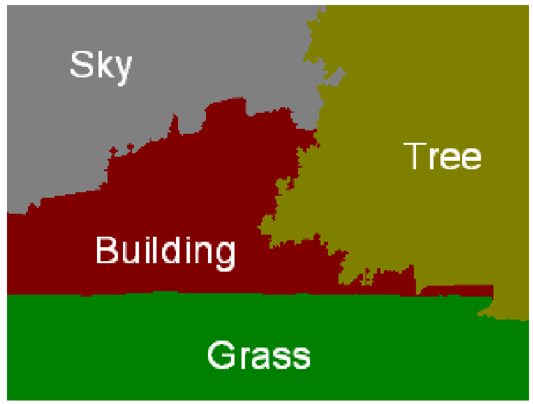


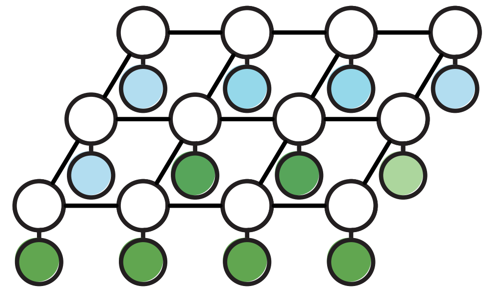
How learn who influences whom?

Summarizing Documents

How select representative sentences?

MAP inference





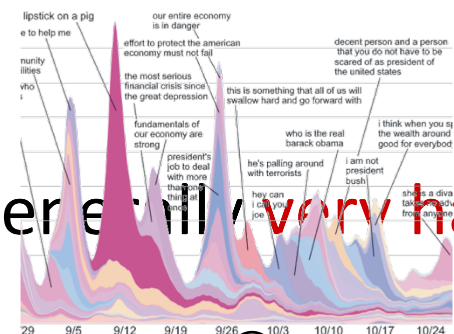
$$\max_x \quad p(x \mid z)$$

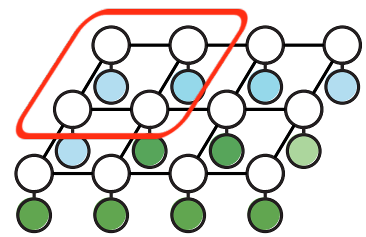
How find the MAP labeling in discrete graphical models *efficiently*?

What’s common?

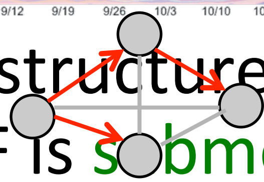
- Formalization:

  Optimize a set function F(S) under constraints





- generally very hard



- but: structure helps!
  … if F is submodular, we can …
  - solve optimization problems with strong guarantees
  - solve some learning problems

Outline

many new results! ☺

- What is submodularity?
- Optimization
  - Minimization
  - Maximization
- Learning
- Learning for Optimization: new settings

Part I

Break

Part II

Outline

many new results! ☺

- What is submodularity?
- Optimization
  - Minimization: new algorithms, constraints
  - Maximization: new algorithms (unconstrained)
- Learning
- Learning for Optimization: new settings

Part I

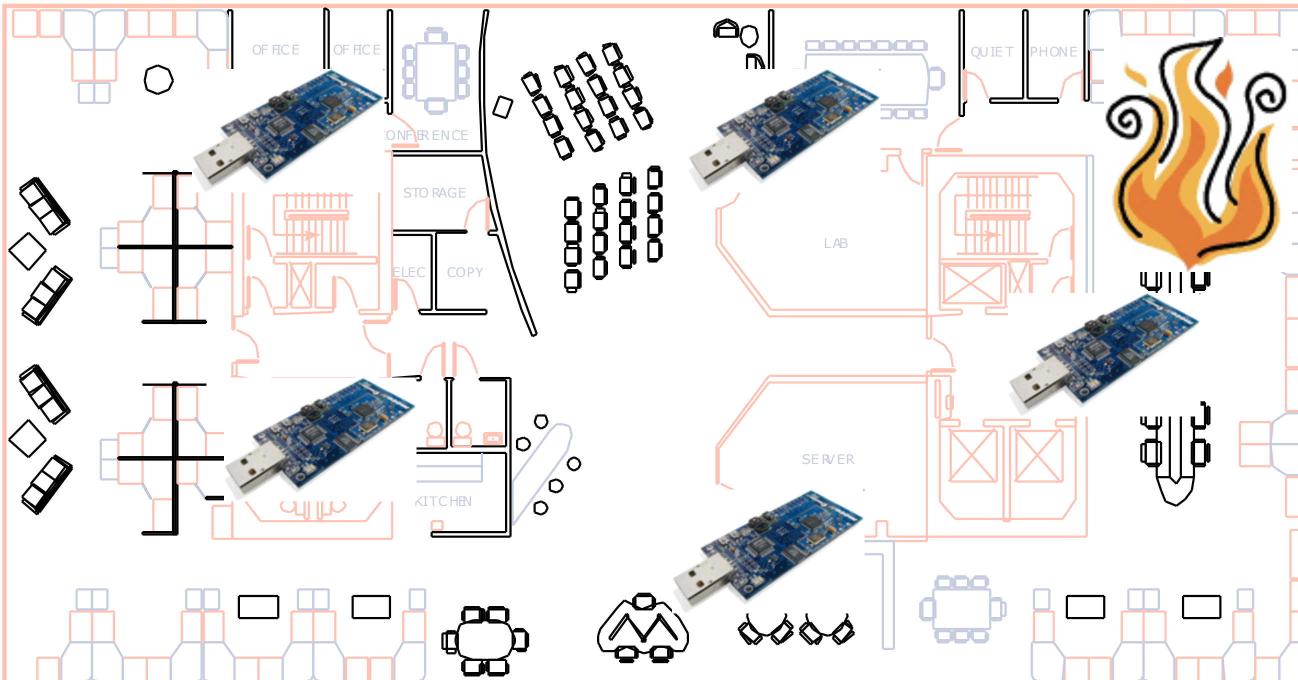
Part II

… and many new applications!

submodularity.org

slides, links, references, workshops, …

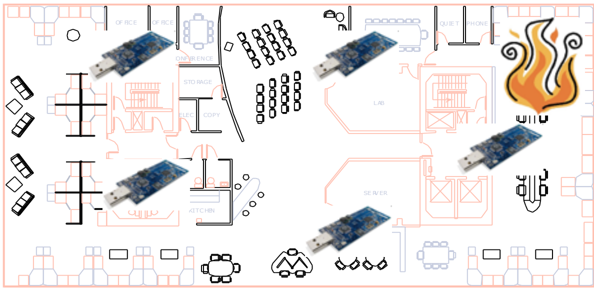
Example: placing sensors



Place sensors to monitor temperature

Set functions

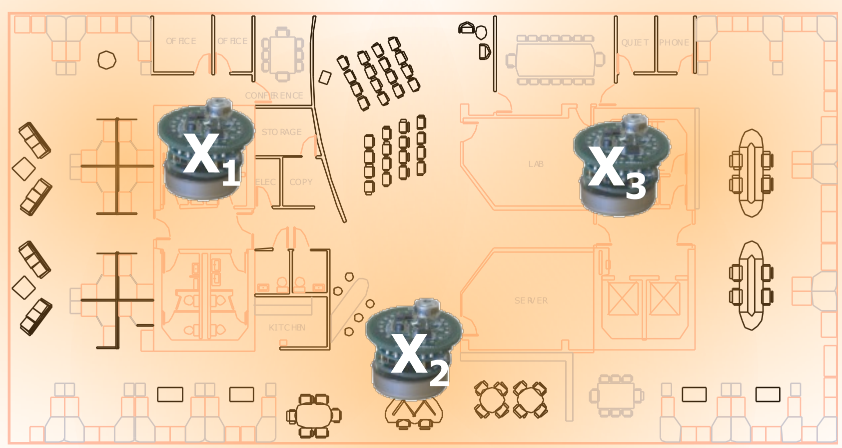
- finite ground set $V = \{1, 2, \ldots, n\}$
- set function $F : 2^V \to \mathbb{R}$



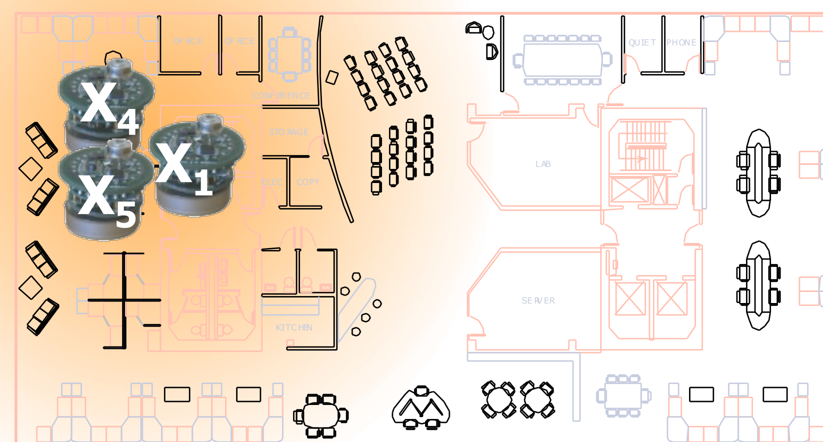
- will assume $F(\emptyset) = 0$ (w.l.o.g.)
- assume black box that can evaluate $F(A)$ for any $A \subseteq V$

Example: placing sensors

Utility $F(A)$ of having sensors at subset A of all locations



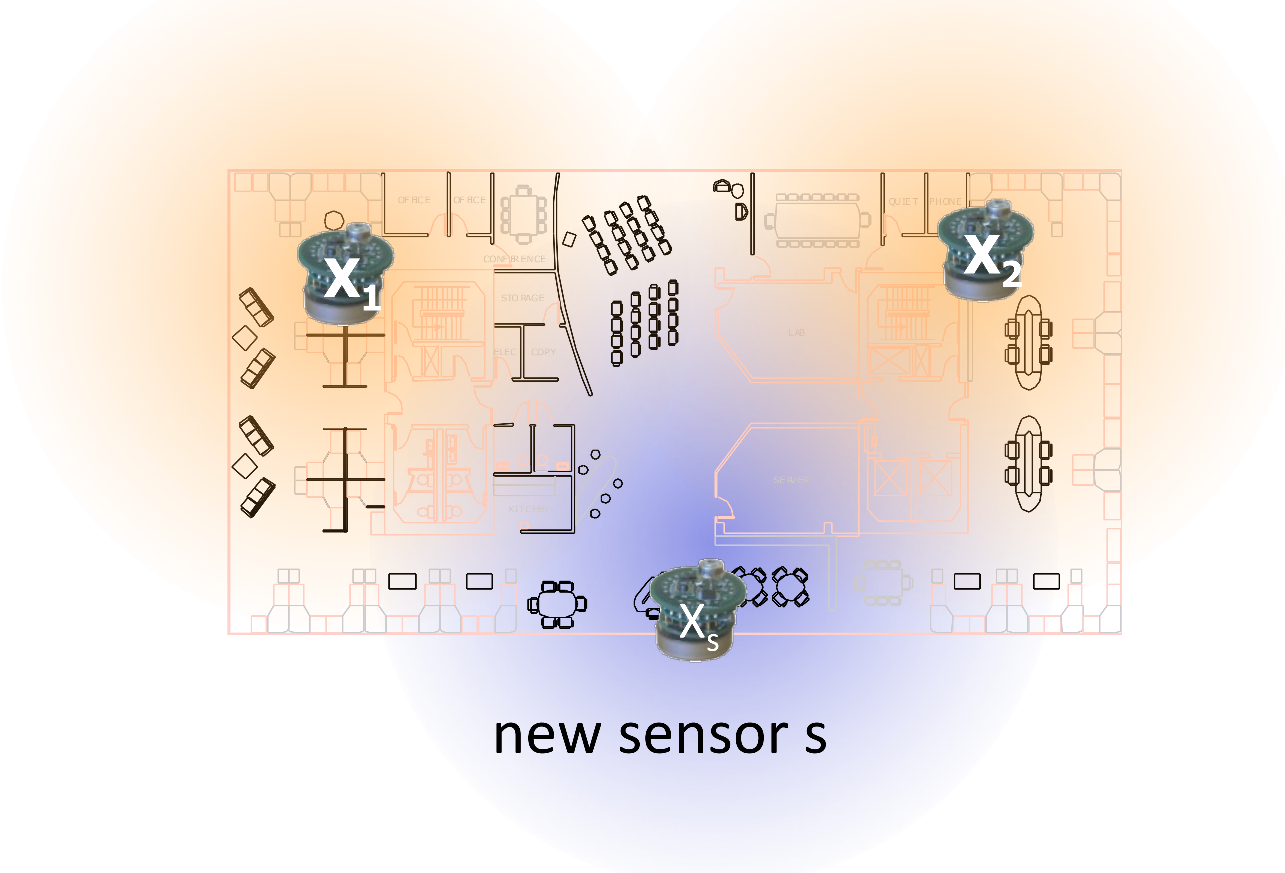
A={1,2,3}: Very informative
High value F(A)



A={1,4,5}: Redundant info
Low value F(A)

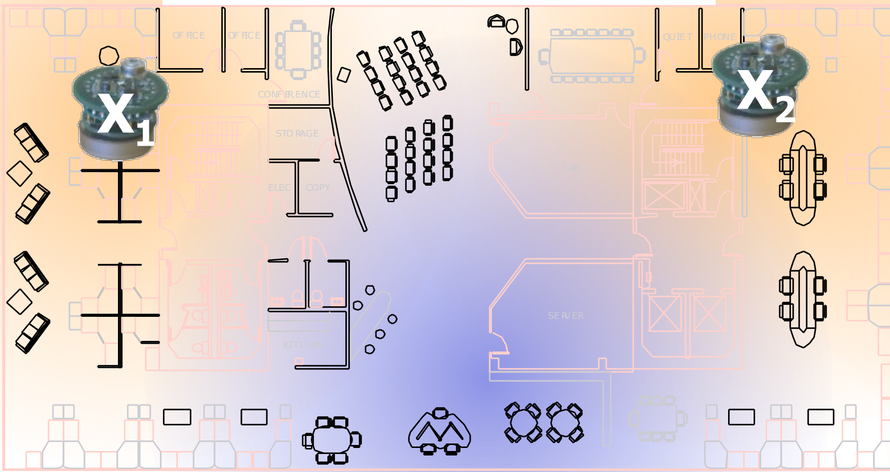
Marginal gain

- Given set function $F : 2^V \rightarrow \mathbb{R}$
- Marginal gain: $\Delta_F(s \mid A) = F(\{s\} \cup A) - F(A)$



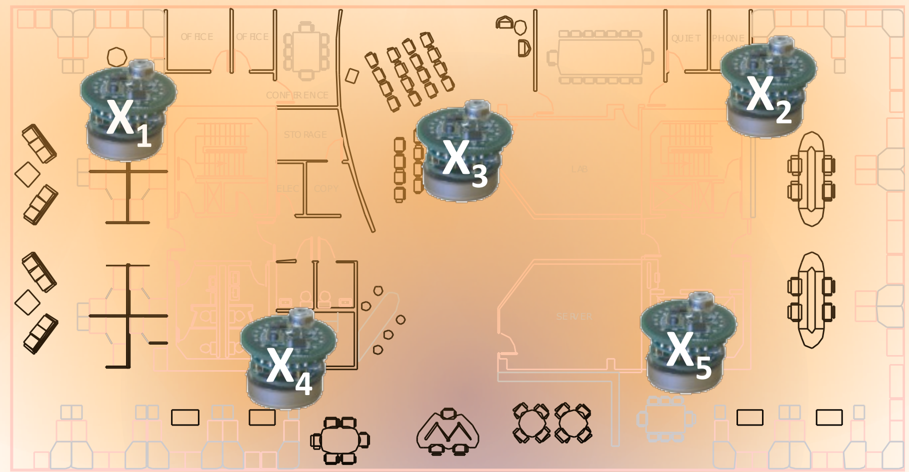
Decreasing gains: submodularity

placement A = {1,2}

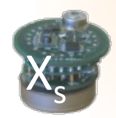


Big gain

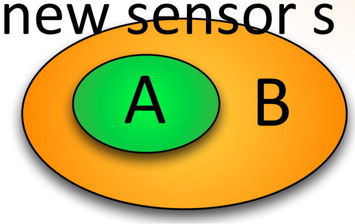
placement B = {1,…,5}



small gain



new sensor s



+ • s

+ • s

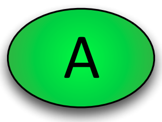
$$A \subseteq B$$

$$F(A \cup s) - F(A)$$

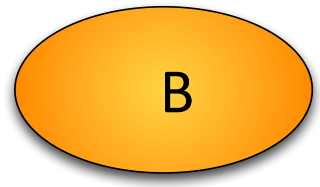
$$\Delta(s \mid A)$$

Equivalent characterizations

- Diminishing gains: for all $A \subseteq B$



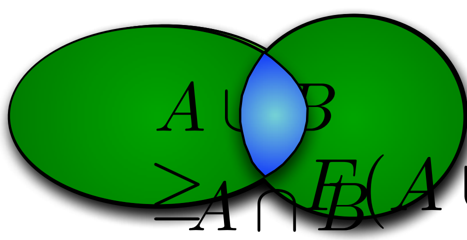
+ • s



+ • s

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

- Union-Intersection: for all $A, B \subseteq V$



$$F(A) + F(B) \geq F(A \cup B) + F(A \cap B)$$

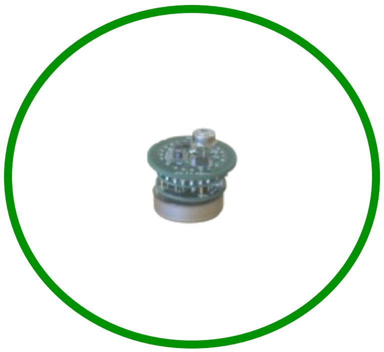
Questions

How do I prove my problem is submodular?

Why is submodularity useful?

Example: Set cover

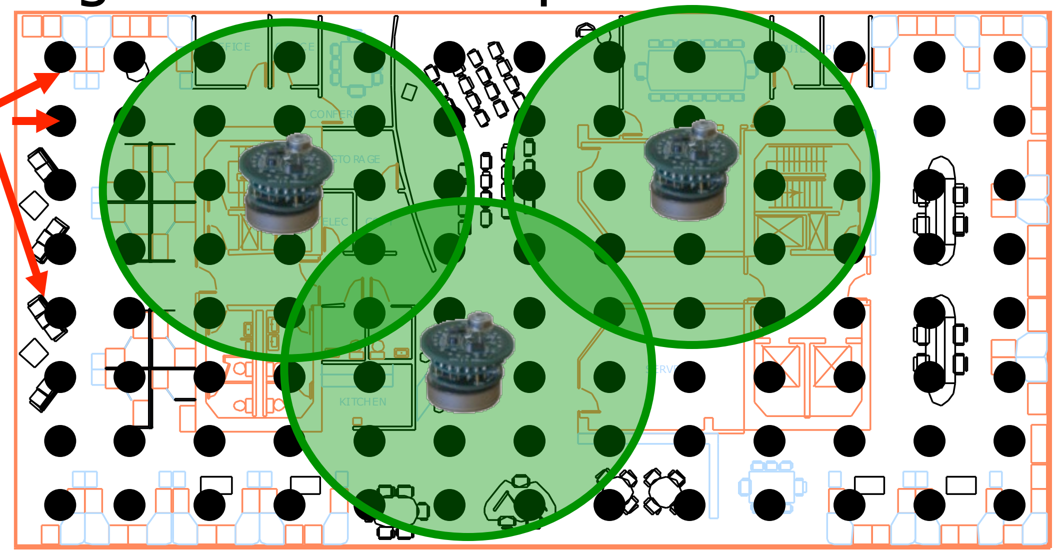
place sensors
in building



Node predicts
values of positions
with some radius

Possible
locations
V

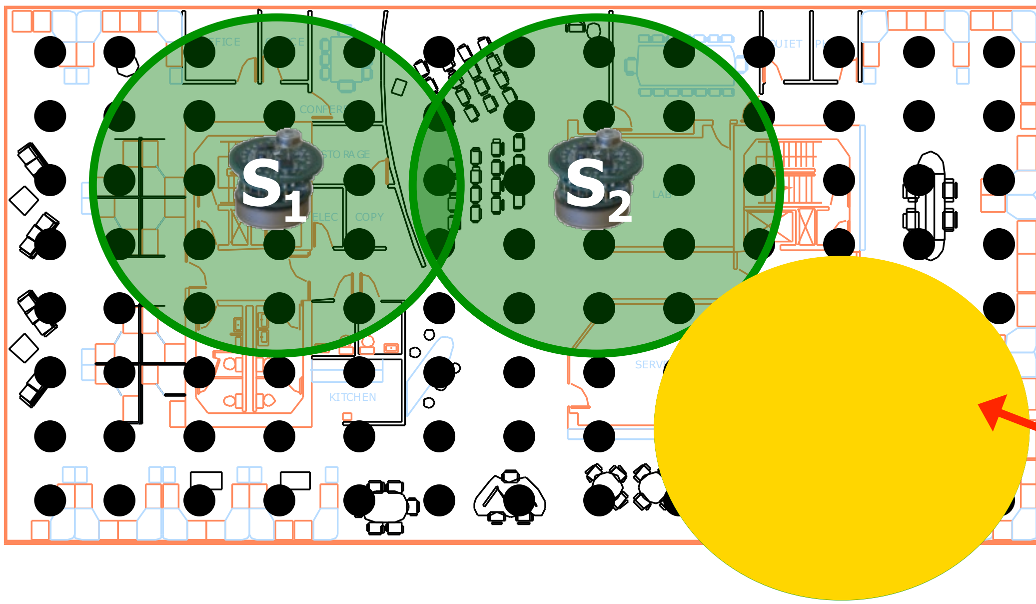
goal: cover floorplan with discs



$A \subseteq V$: $F(A) =$
“area covered by sensors placed at A”

Formally:
Finite set W, collection of n subsets $S_i \subseteq W$
For $A \subseteq V$ define $F(A) = |\bigcup_{i \in A} S_i|$

Set cover is submodular

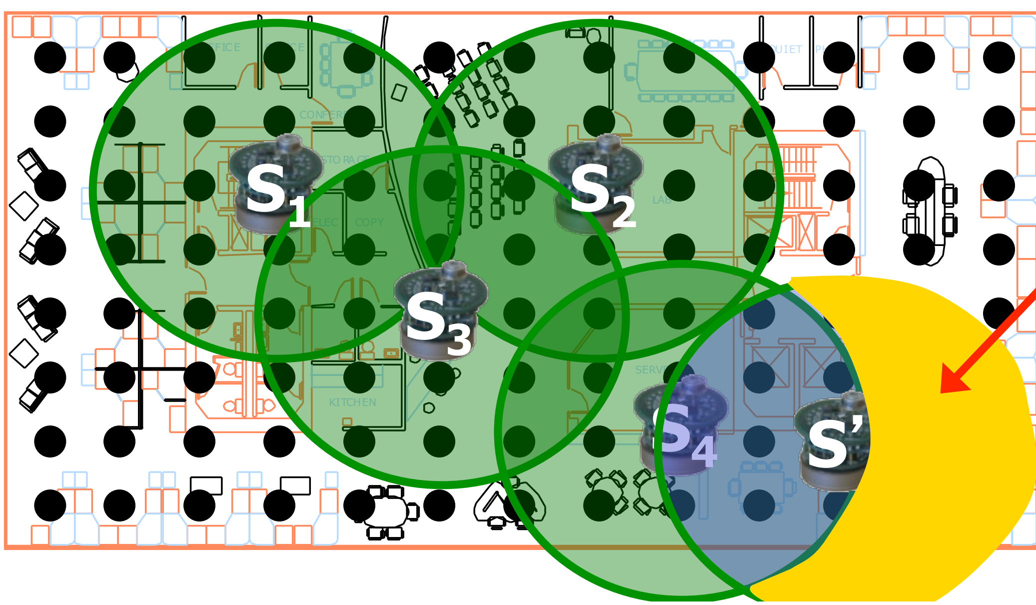


$A=\{s_1,s_2\}$

$F(A \cup \{s'\}) - F(A)$

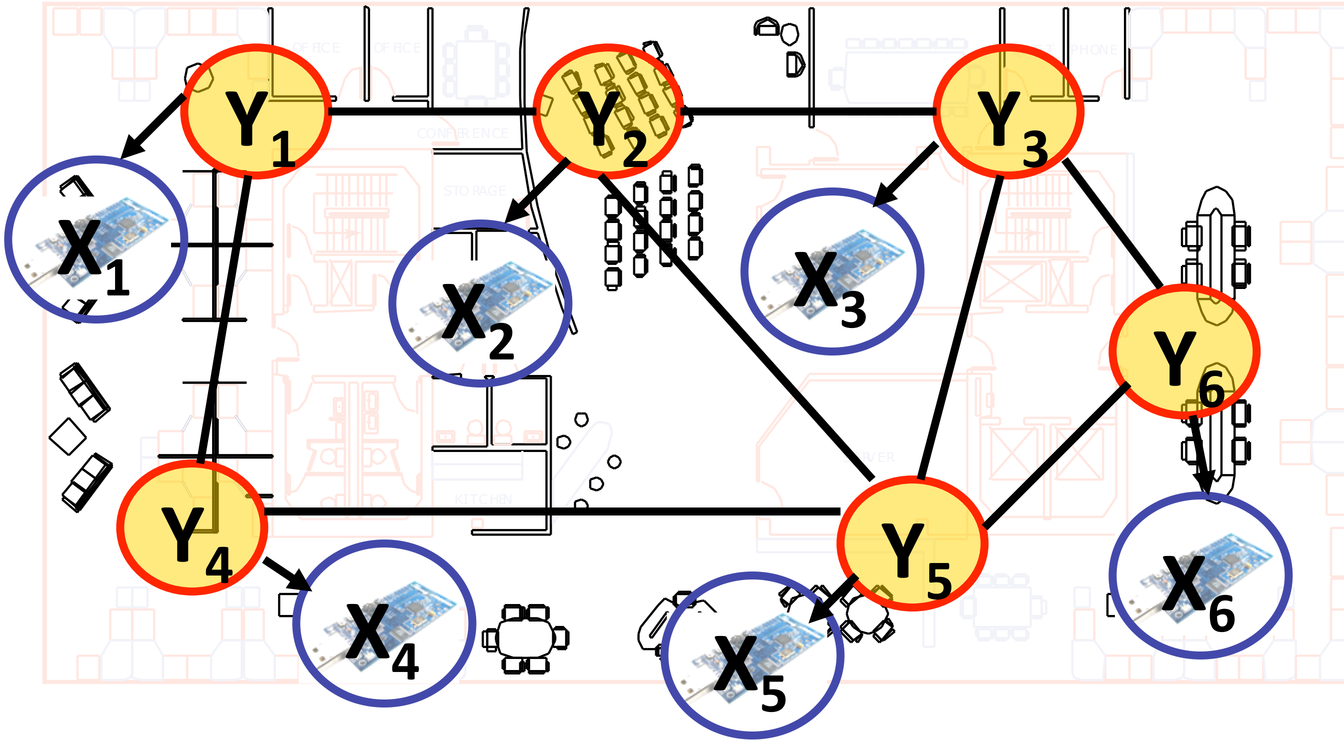
$\geq$

$F(B \cup \{s'\}) - F(B)$



$B = \{s_1,s_2,s_3,s_4\}$

More complex model for sensing



Y_s: temperature at location s

X_s: sensor value at location s

$X_s = Y_s + \text{noise}$

Joint probability distribution

$$P(X_1,\ldots,X_n,Y_1,\ldots,Y_n) = \underbrace{P(Y_1,\ldots,Y_n)}_{\textbf{Prior}}\ \underbrace{P(X_1,\ldots,X_n \mid Y_1,\ldots,Y_n)}_{\textbf{Likelihood}}$$

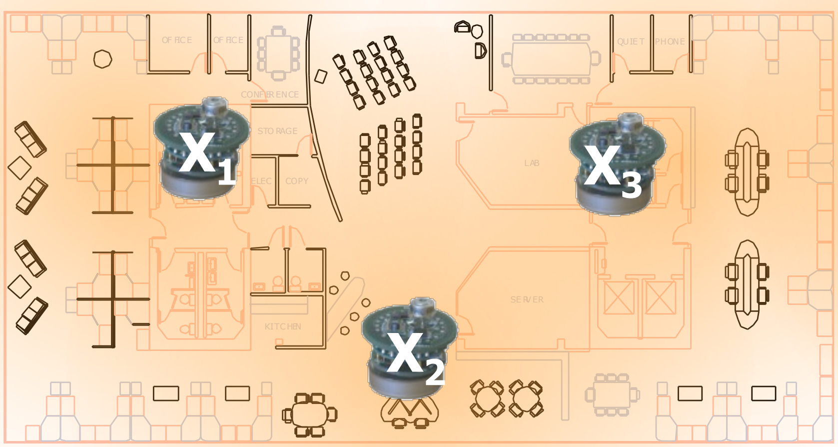
Example: Sensor placement

Utility of having sensors at subset A of all locations

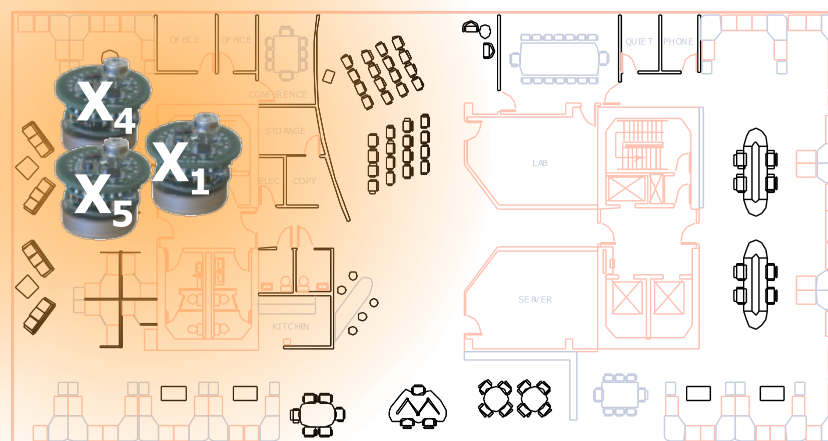
$$F(A) = H(\mathbf{Y}) - H(\mathbf{Y} \mid \mathbf{X}_A)$$

Uncertainty
about temperature Y
before sensing

Uncertainty
about temperature Y
after sensing



A={1,2,3}: High value F(A)



A={1,4,5}: Low value F(A)

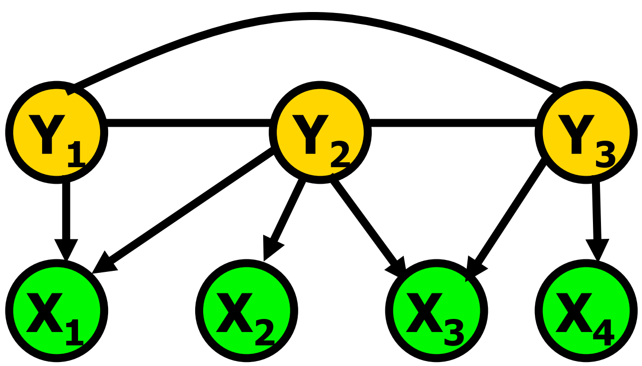
Submodularity of Information Gain

$Y_1,\ldots,Y_m, X_1, \ldots, X_n$ discrete RVs

$F(A) = I(Y; X_A) = H(Y)-H(Y \mid X_A)$

- F(A) is NOT always submodular

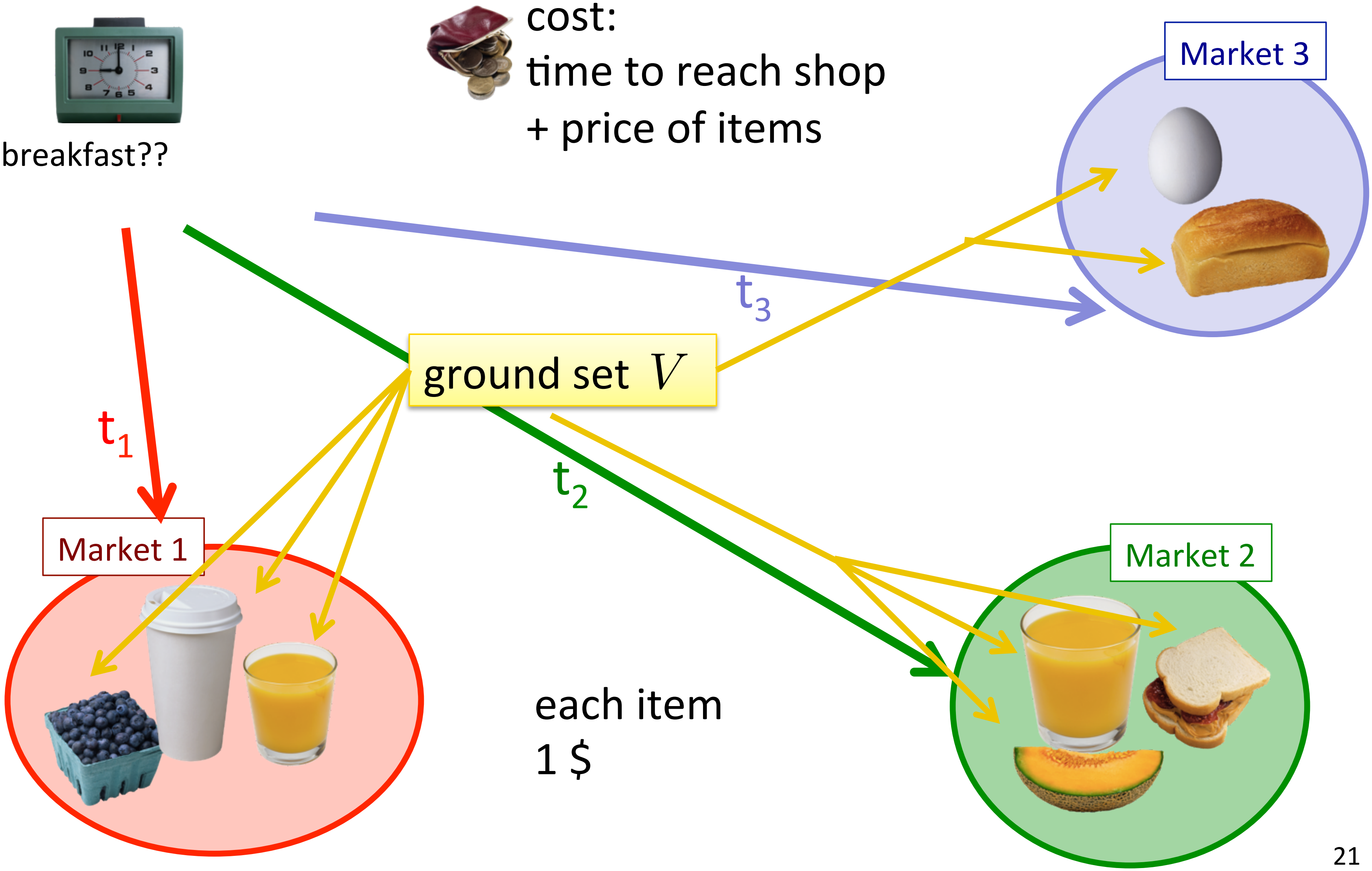
If X_i are all conditionally independent given Y, then F(A) is submodular! [Krause & Guestrin `05]



Proof:
"information never hurts"

Example: costs

cost:
time to reach shop
+ price of items

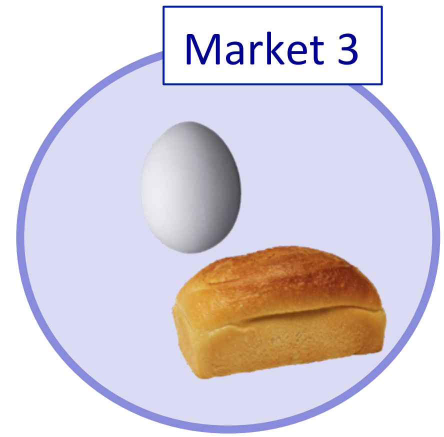


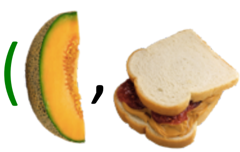
each item
1 $

Example: costs

breakfast??

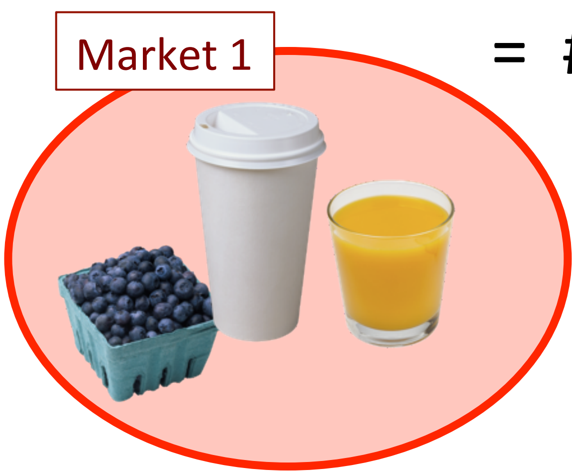
cost:
time to shop
+ price of items



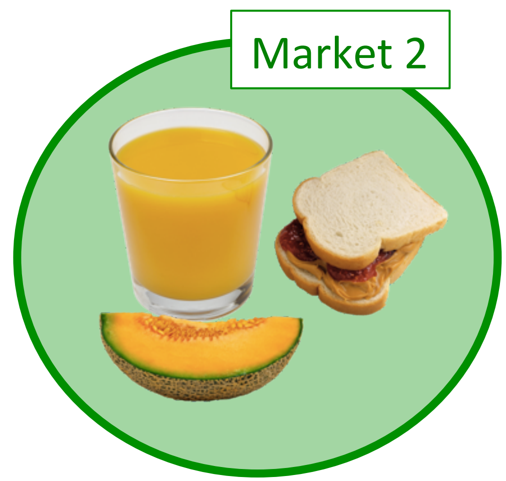
F() = cost() + cost()

$= t_1 + 1 \; + \; t_2 \; + 2$

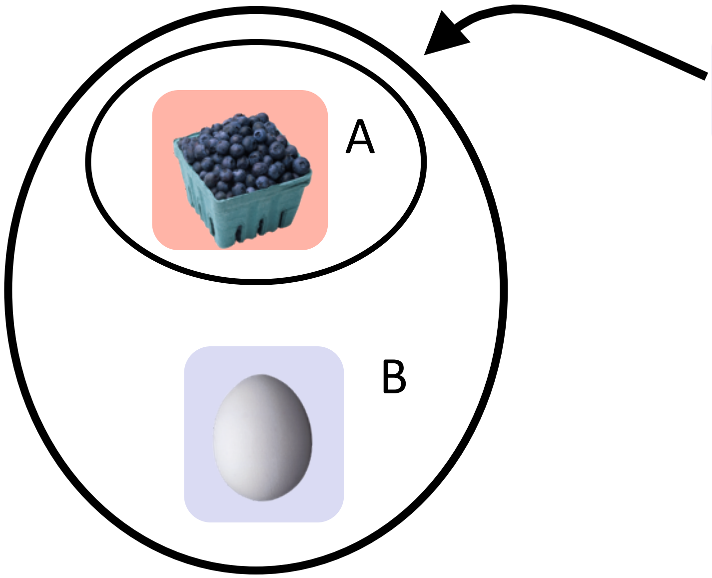
= #shops + #items



submodular?

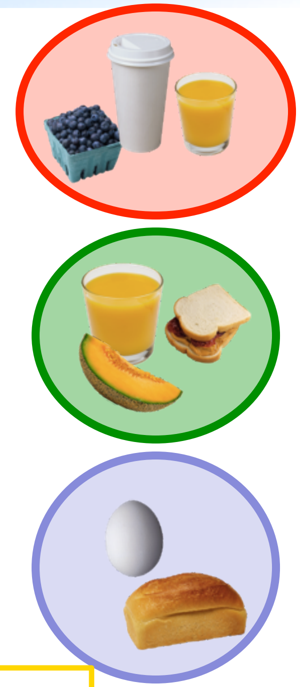


Shared fixed costs



$$\Delta(b \mid A) = 1 + t_3$$

$$\Delta(b \mid B) = 1$$

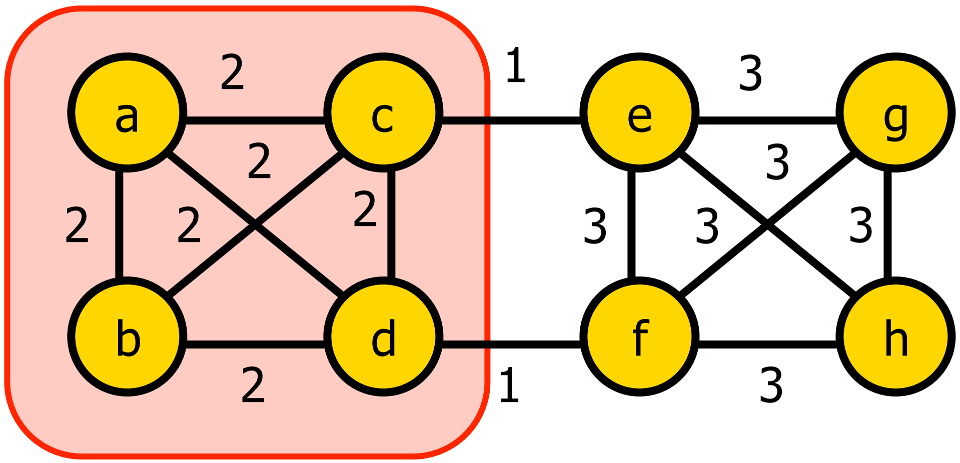


marginal cost: #new shops + #new items

decreasing ➔ cost is submodular!

- shops: shared fixed cost
- economies of scale

Another example: Cut functions

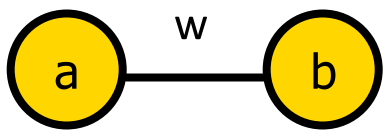


V={a,b,c,d,e,f,g,h}

$$F(A) = \sum_{s \in A, t \notin A} w_{s,t}$$

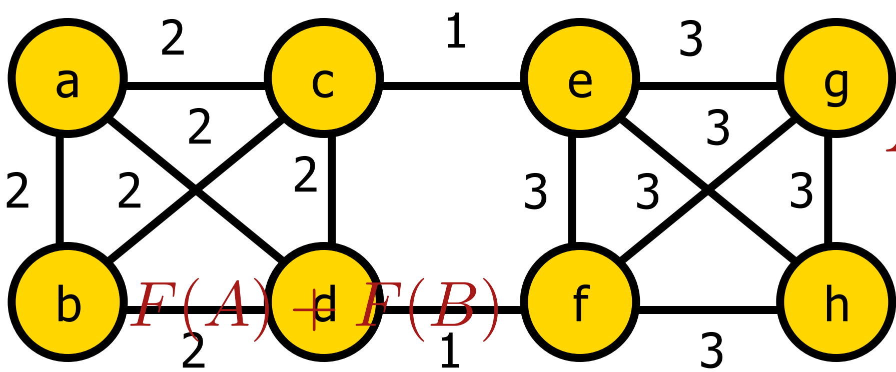
Cut function is submodular!

Why are cut functions submodular?



	S	F_{ab}(S)
$A \cap B$	{}	0
A	{a}	w
B	{b}	w
$A \cup B$	{a,b}	0

Submodular if w ≥ 0!



$$F(A) + F(B) \geq F(A \cap B) + F(A \cup B)$$

$$F(S) = \sum_{(i,j)\in E} \underbrace{F_{i,j}(S \cap \{i,j\})}_{\text{Cut function in subgraph } \{i,j\}}$$

Cut function in subgraph {i,j}

➔ Submodular!

Closedness properties

$F_1,\ldots,F_m$ submodular functions on V and $\lambda_1,\ldots,\lambda_m > 0$

Then: $F(A) = \sum_i \lambda_i F_i(A)$ is submodular

Submodularity closed under nonnegative linear combinations!

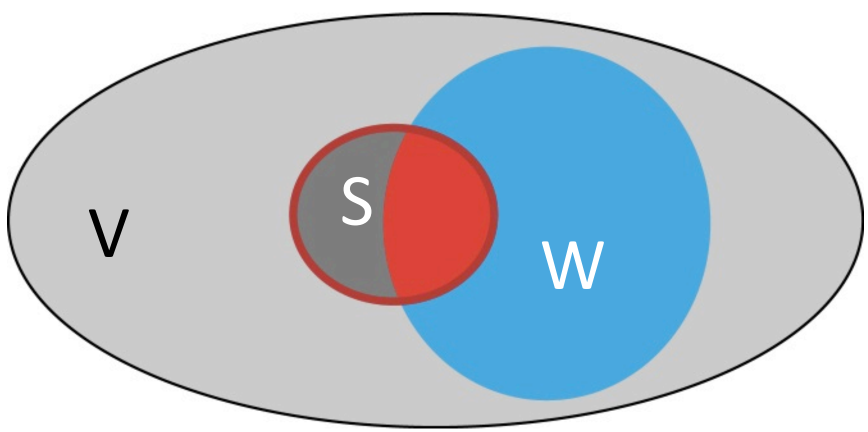
Extremely useful fact:

- $F_\theta(A)$ submodular ➔ $\sum_\theta P(\theta) F_\theta(A)$ submodular!
- Multicriterion optimization
- A basic proof technique! ☺

Other closedness properties

- Restriction: F(S) submodular on V, W subset of V

  Then $F'(S) = F(S \cap W)$ is submodular



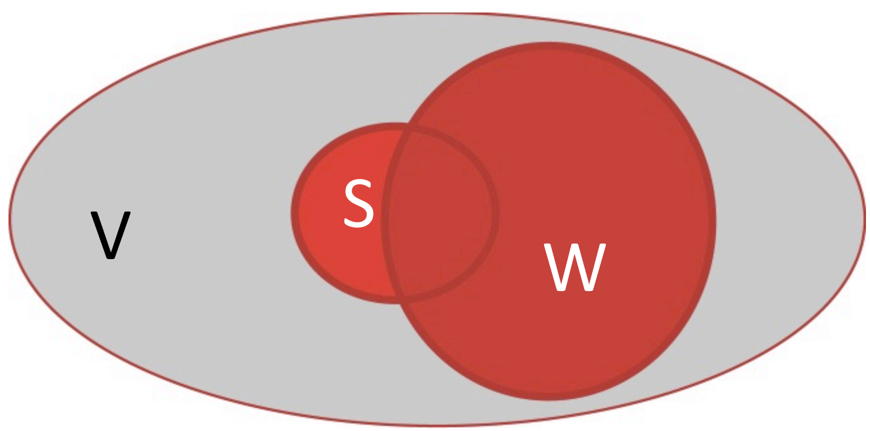
Other closedness properties

- Restriction: F(S) submodular on V, W subset of V

  Then $F'(S) = F(S \cap W)$ is submodular

- Conditioning: F(S) submodular on V, W subset of V

  Then $F'(S) = F(S \cup W)$ is submodular



Other closedness properties

- Restriction: F(S) submodular on V, W subset of V

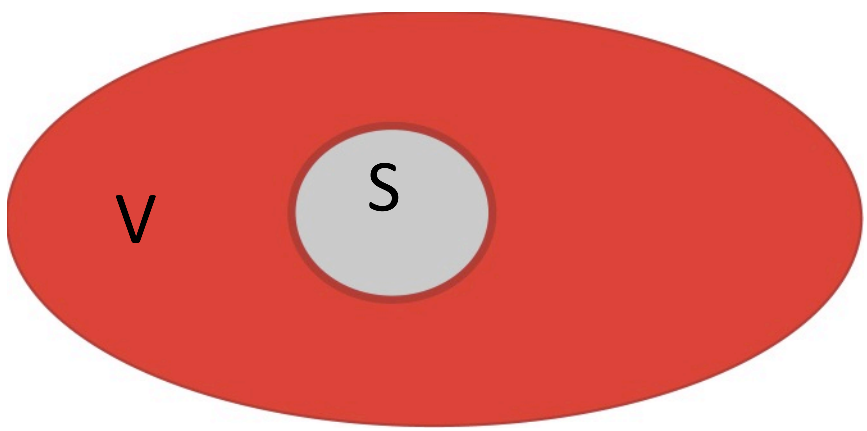
  Then $$F'(S) = F(S \cap W)$$ is submodular

- Conditioning: F(S) submodular on V, W subset of V

  Then $$F'(S) = F(S \cup W)$$ is submodular

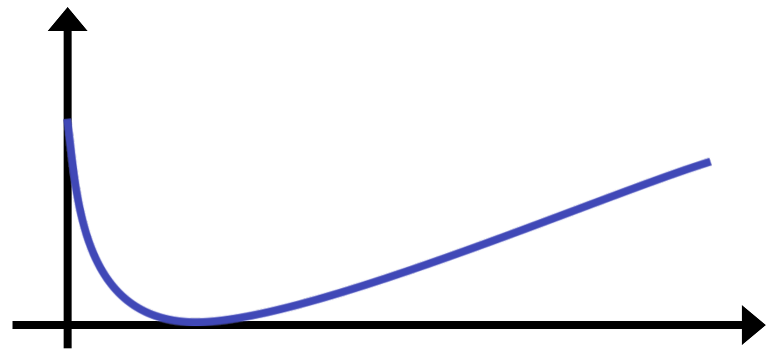
- Reflection: F(S) submodular on V

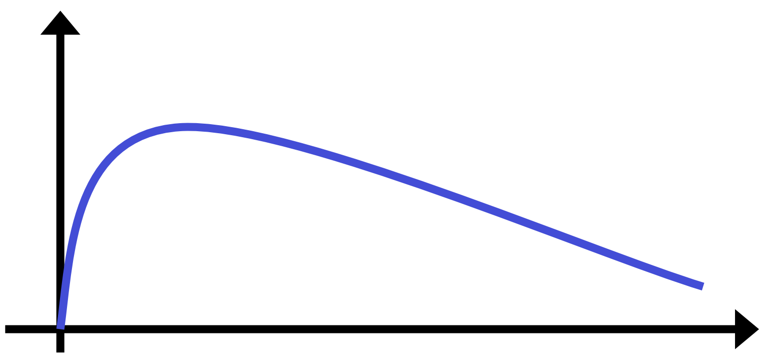
  Then $$F'(S) = F(V \setminus S)$$ is submodular



Submodularity …

discrete convexity ….

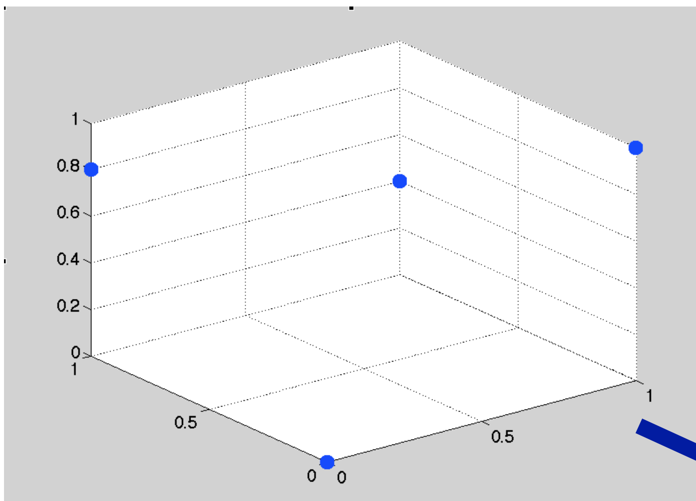


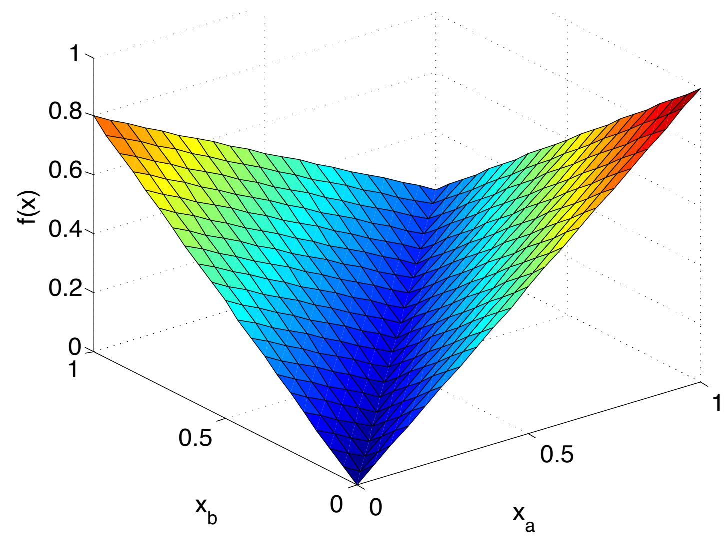


… or concavity?

Convex aspects

- convex extension
  - duality
  - efficient minimization





But this is only half of the story…

Concave aspects

- submodularity:

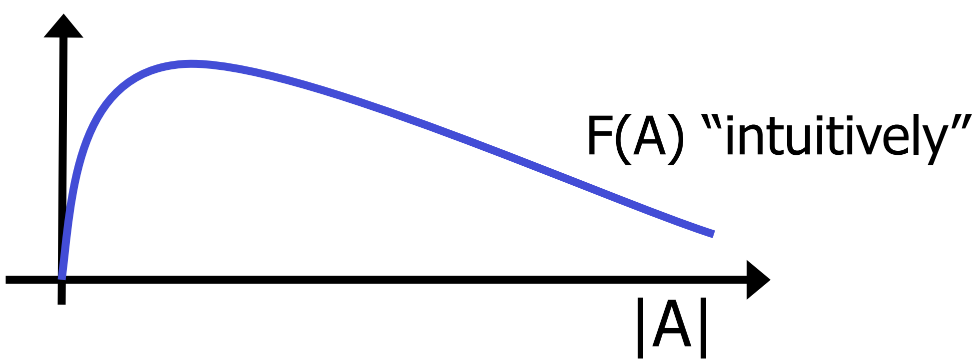
$$A \subseteq B, \;\; s \notin B :$$

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

- concavity:

$$a \leq b, \;\; s > 0 :$$

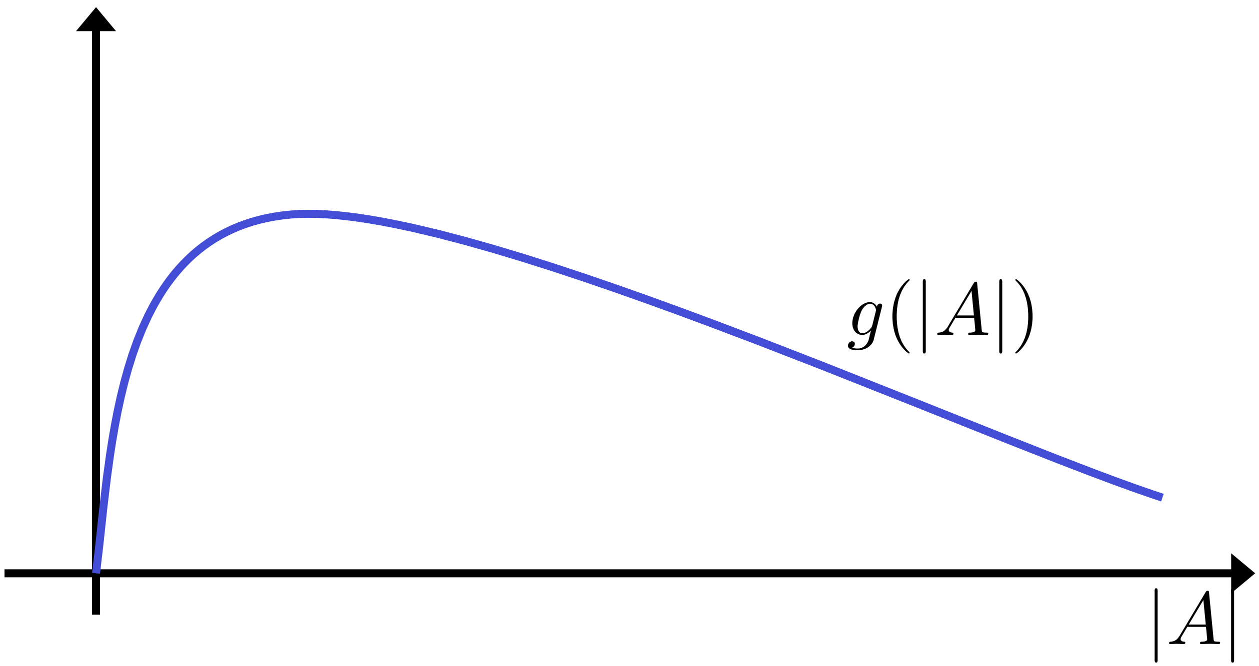
$$f(a+s) - f(a) \quad \geq \quad f(b+s) - f(b)$$



Submodularity and concavity

- suppose $g : \mathbb{N} \to \mathbb{R}$ and $F(A) = g(|A|)$

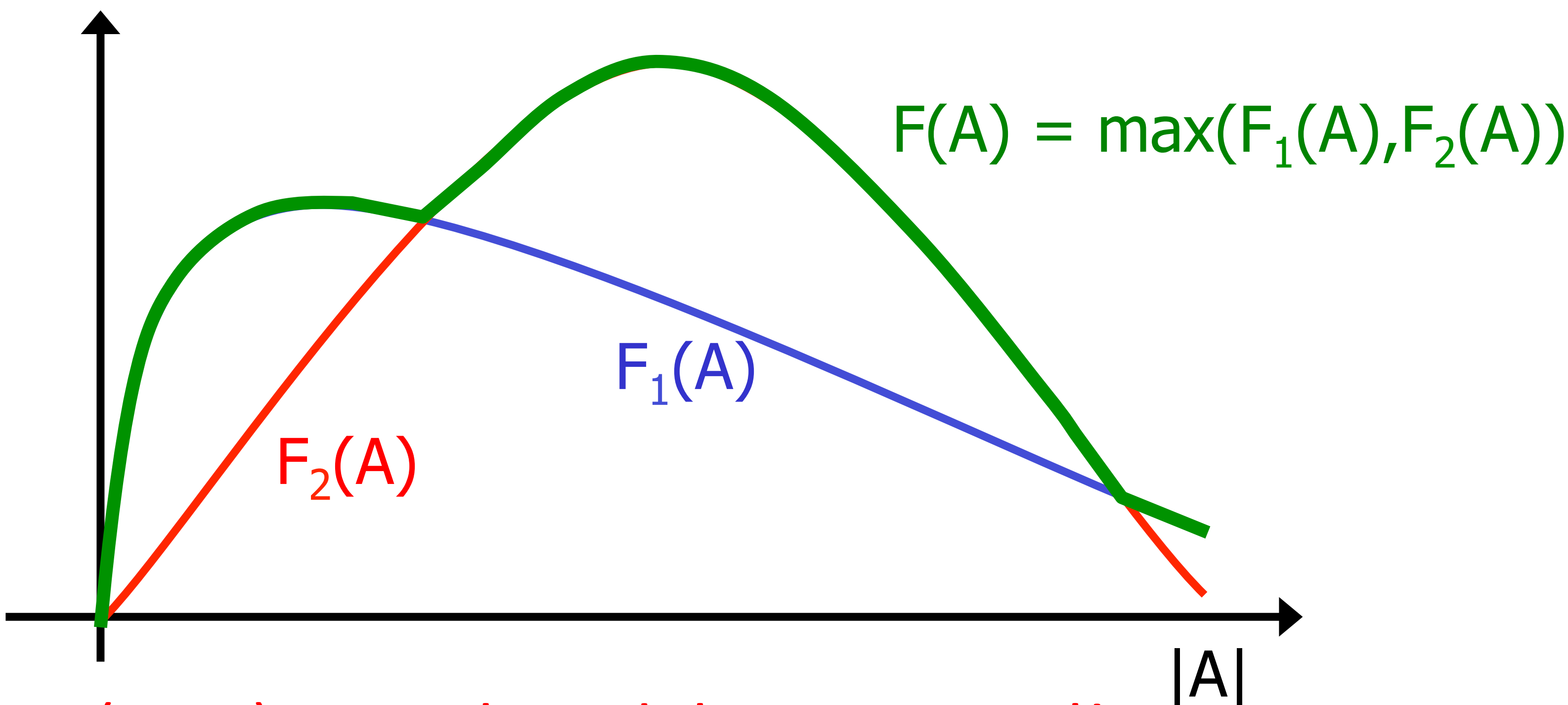
$F(A)$ submodular *if and only if* ... g is concave



Maximum of submodular functions

- $F_1(A), F_2(A)$ submodular. What about

$$F(A) = \max\{\, F_1(A), F_2(A) \,\} \quad ?$$



max(F_1,F_2) not submodular in general!

Minimum of submodular functions

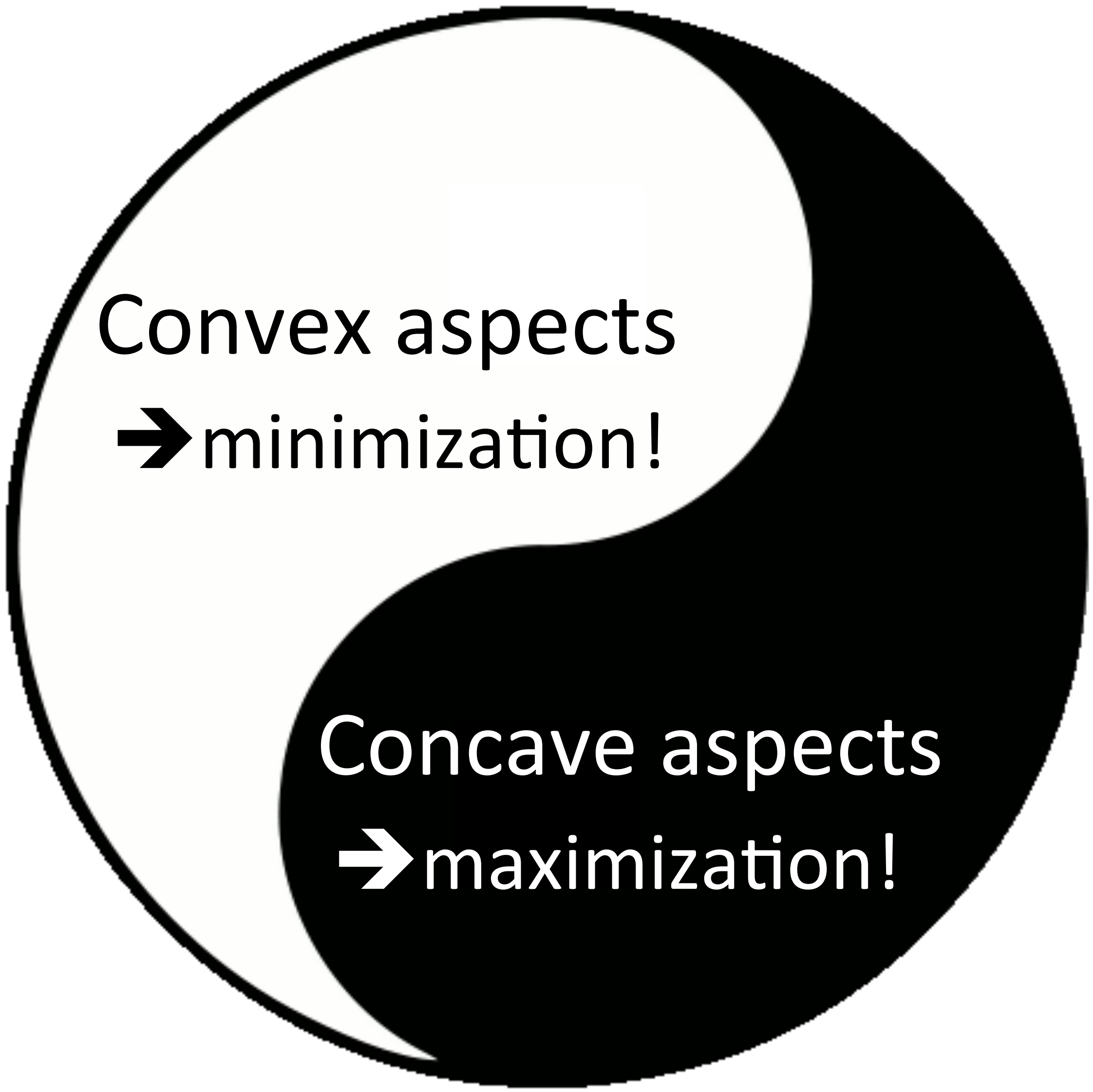
Well, maybe $F(A) = \min(F_1(A), F_2(A))$ instead?

	$F_1(A)$	$F_2(A)$
{}	0	0
{a}	1	0
{b}	0	1
{a,b}	1	1

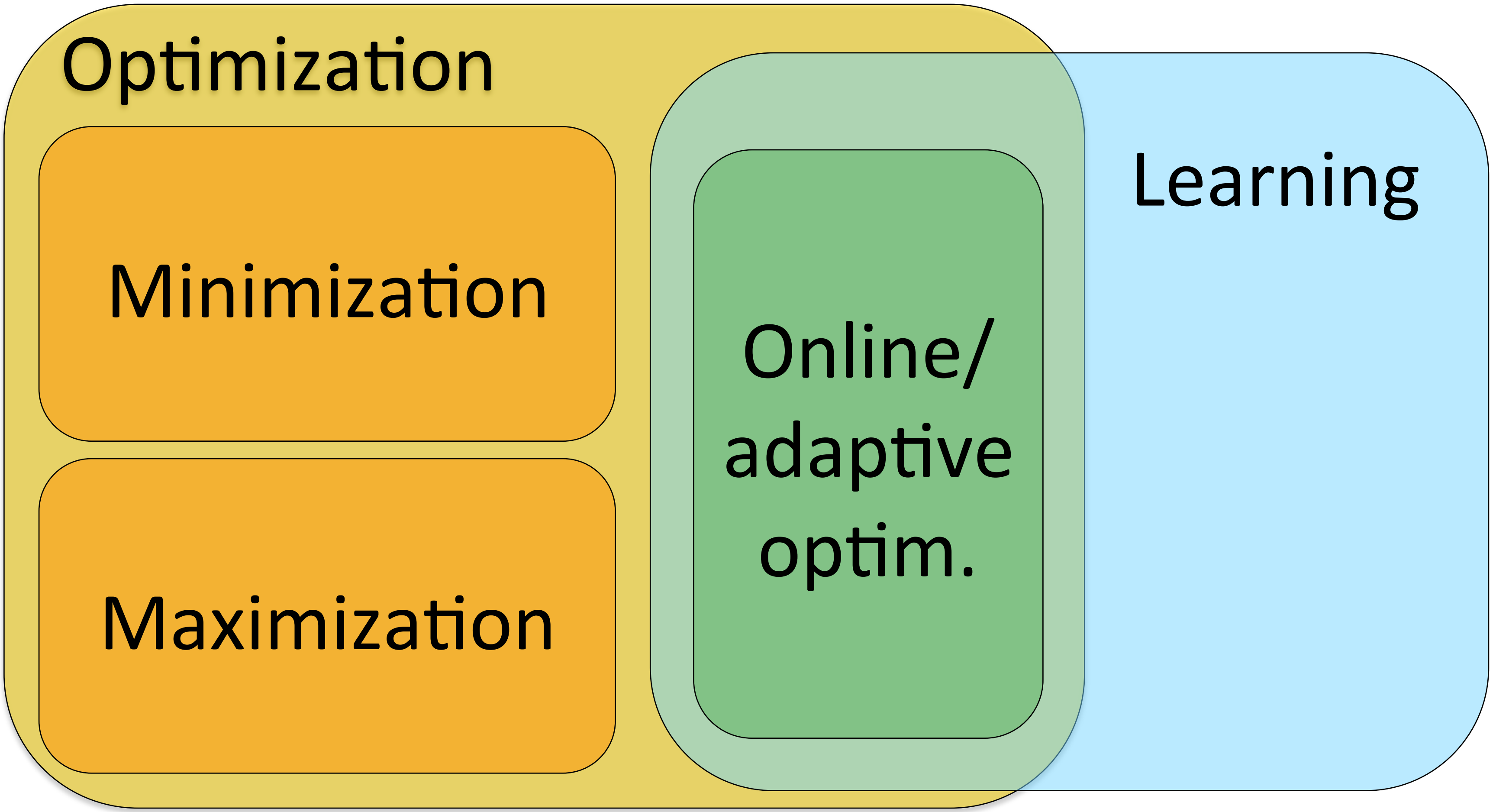
$$F(\{b\}) - F(\{\}) = 0$$
$$<$$
$$F(\{a,b\}) - F(\{a\}) = 1$$

min(F_1,F_2) not submodular in general!

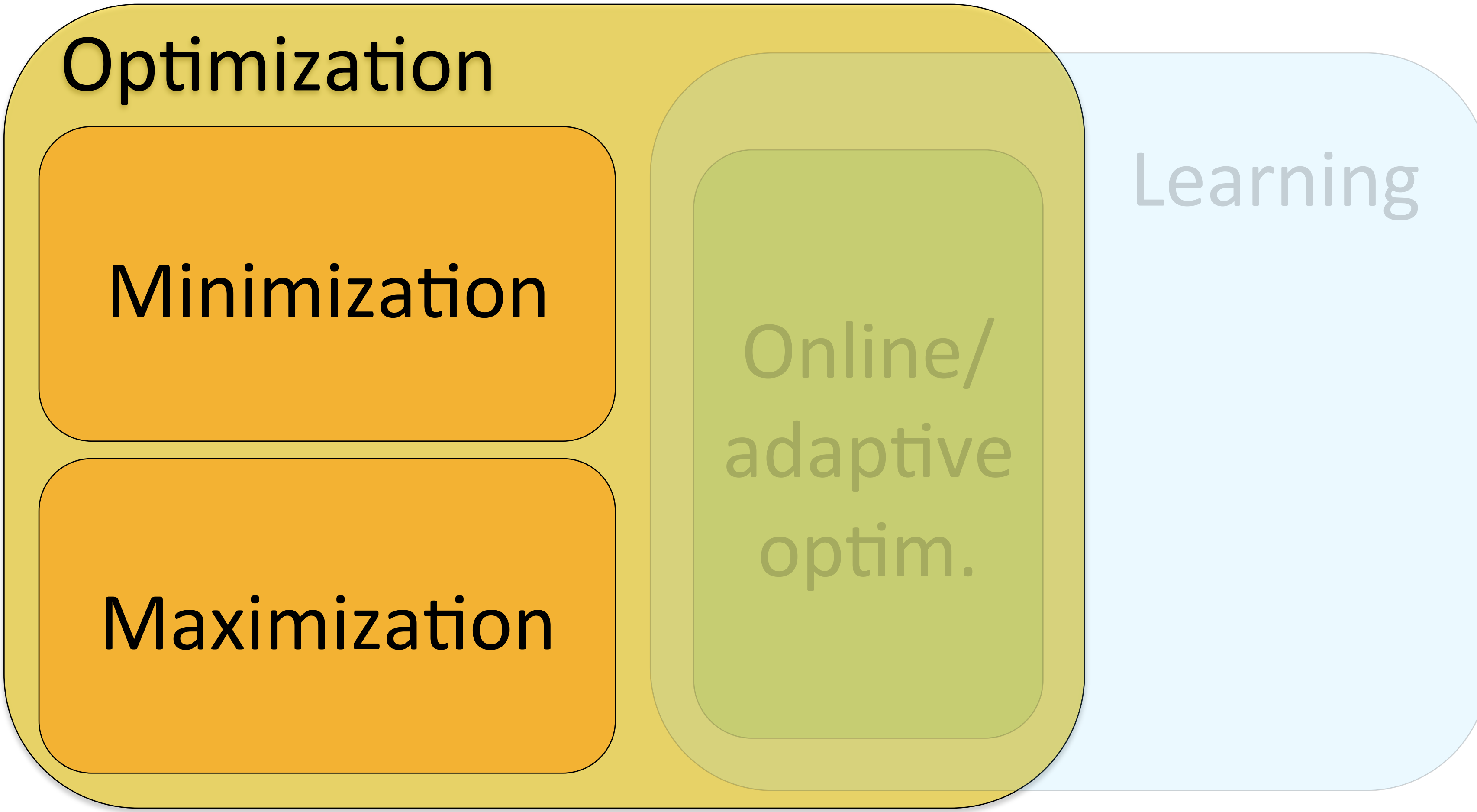
Two faces of submodular functions



What to do with submodular functions

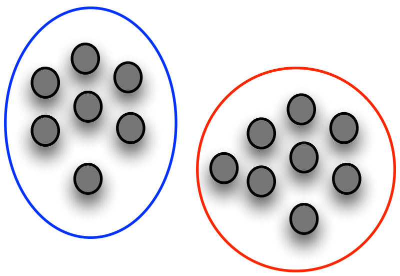


What to do with submodular functions

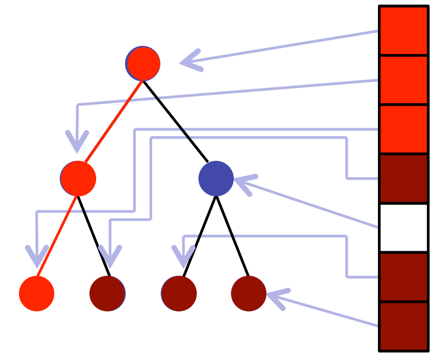


Minimization and maximization not the same??

Submodular minimization

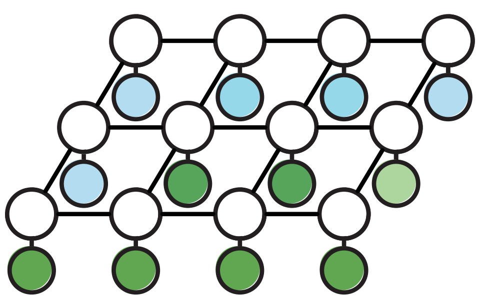


clustering

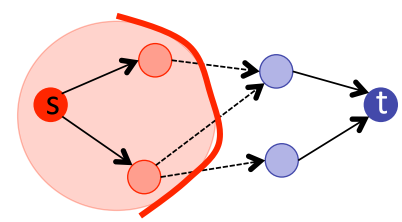


structured sparsity regularization

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



MAP inference



minimum cut

Submodular minimization

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

➔ submodularity and convexity

Set functions and energy functions

any set function

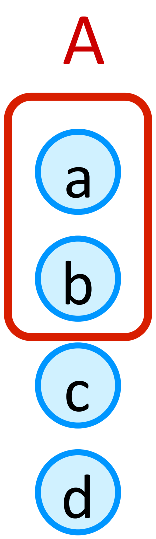
with $|V| = n$

... is a function on binary vectors!

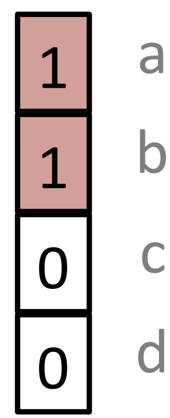
$$F : 2^V \to \mathbb{R}$$

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

$$x = e_A$$



$$\hat{=}$$

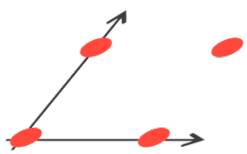


pseudo-boolean function

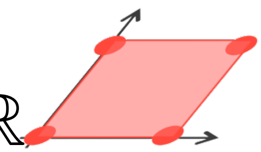
Submodularity and convexity

extension

$$f : [0,1]^n \to \mathbb{R}$$



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



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!

Grötschel, Lovász, Schrijver 1981

Submodularity and convexity

extension

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

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!

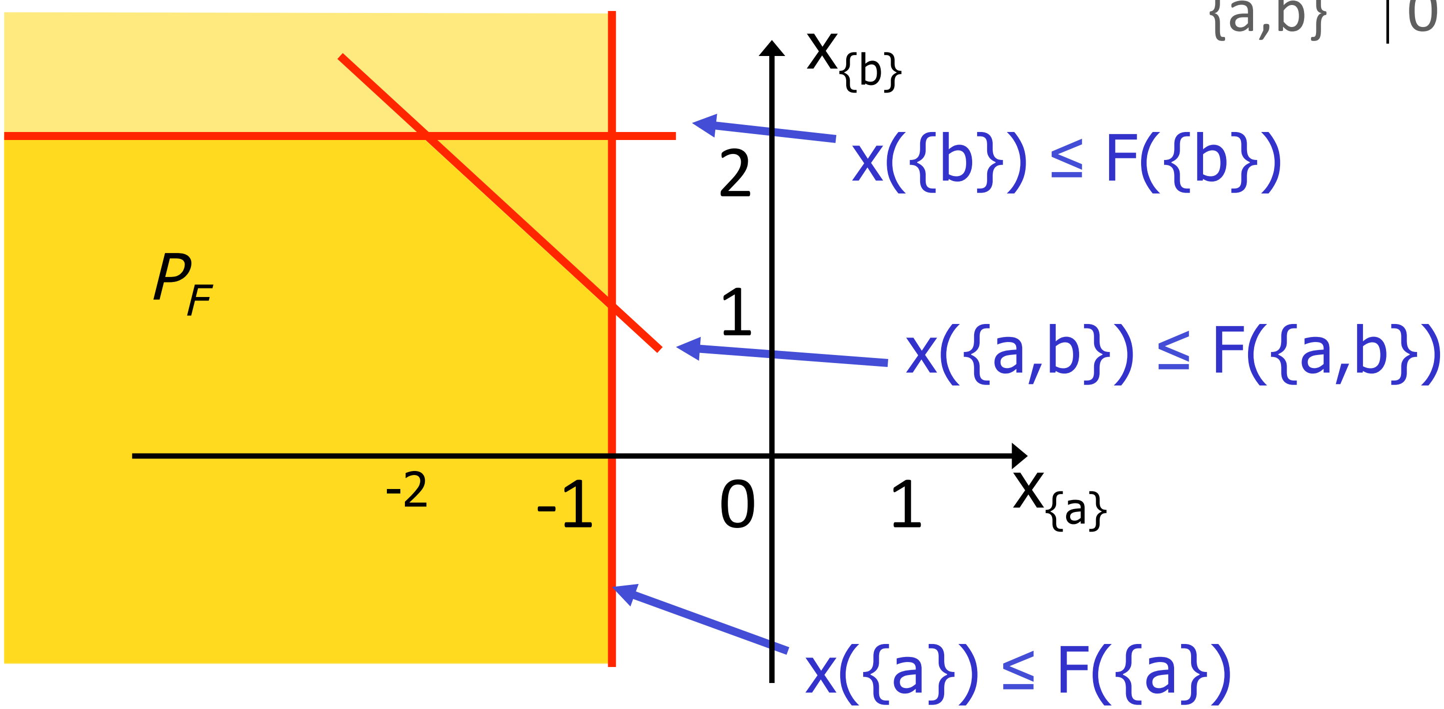
The submodular polyhedron P_F

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

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

Example: V = {a,b}

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



Evaluating the Lovász extension

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

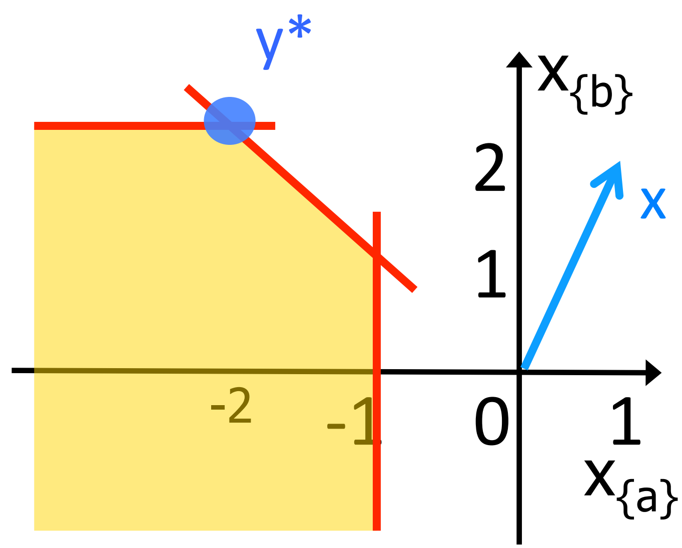
Linear maximization over P_F

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

Exponentially many constraints!!! ☹

Computable in O(n log n) time ☺

[Edmonds '70]



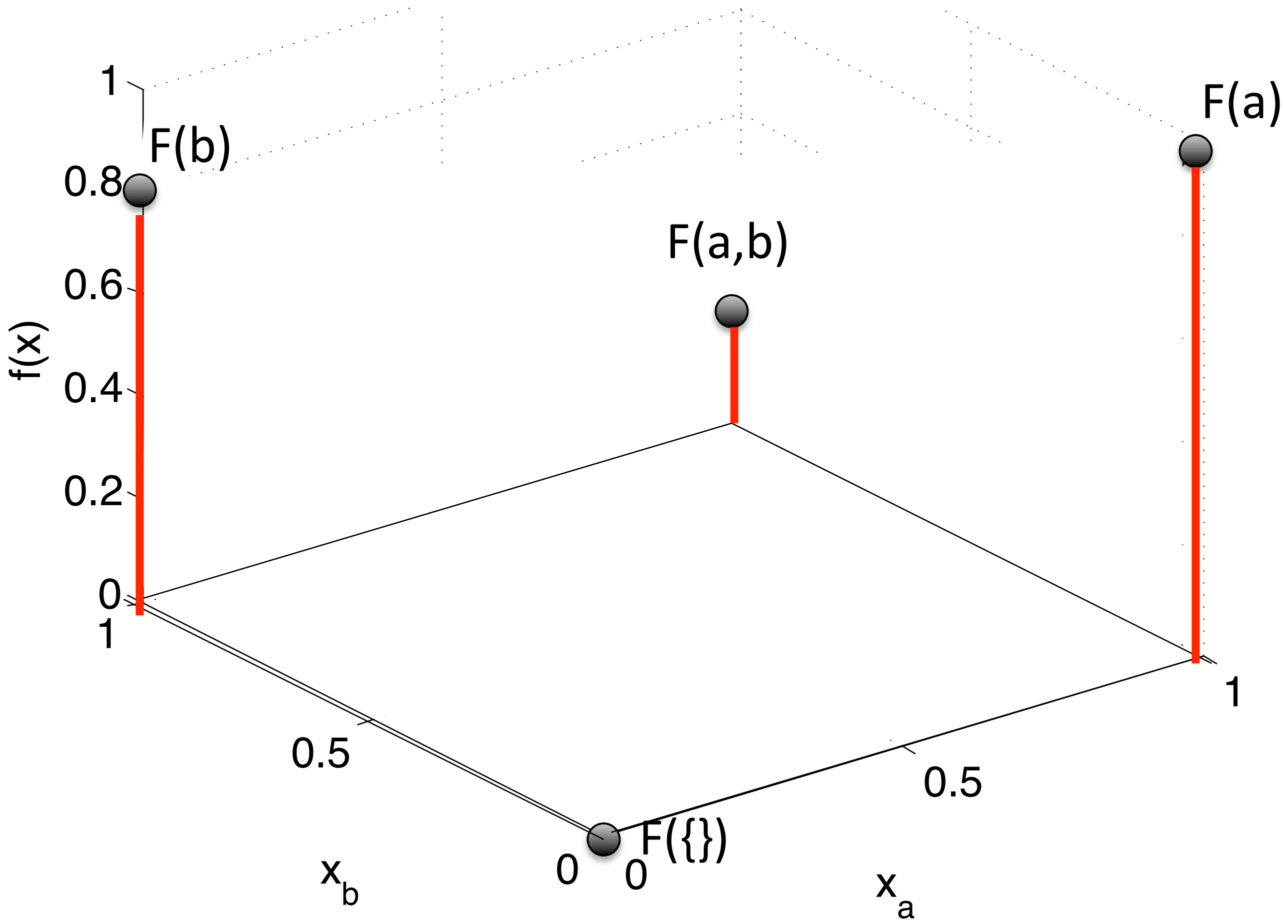
greedy algorithm:

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

⇨

- Subgradient
- Separation oracle

Lovász extension: example



A	F(A)
{}	0
{a}	1
{b}	.8
{a,b}	.2

Submodular minimization

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

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]

T = time for evaluating *F*

The minimum-norm-point algorithm

Example: V = {a,b}

Lovász extension

Regularized problem

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

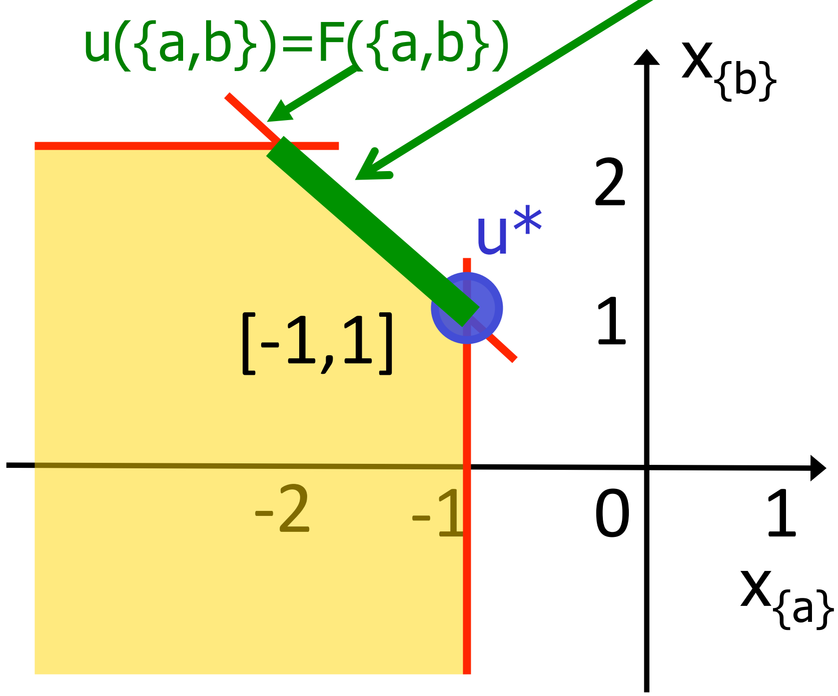
$$\min_{x \in [0,1]^n} f(x) + \frac{1}{2}\|x\|^2$$

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

dual: minimum norm problem

$$u^* = \arg\min_{u \in B_F} \frac{1}{2}\|u\|^2$$

Base polytope B_F



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

minimizes *F*:

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

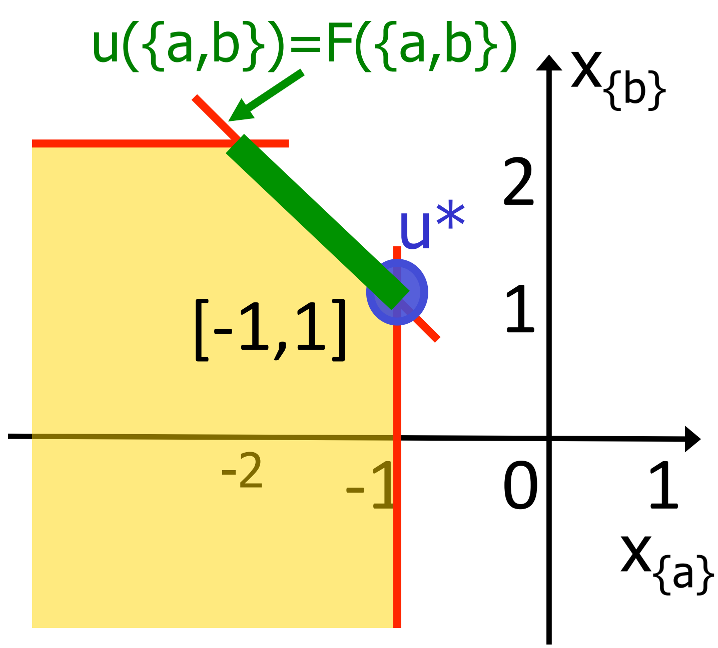
Fujishige '91, Fujishige & Isotani '11

The minimum-norm-point algorithm

1. find $u^* = \arg\min_{u \in B_F} \frac{1}{2}\|u\|^2$

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

can we solve this??



yes! ☺

recall: can solve

linear optimization over P_F

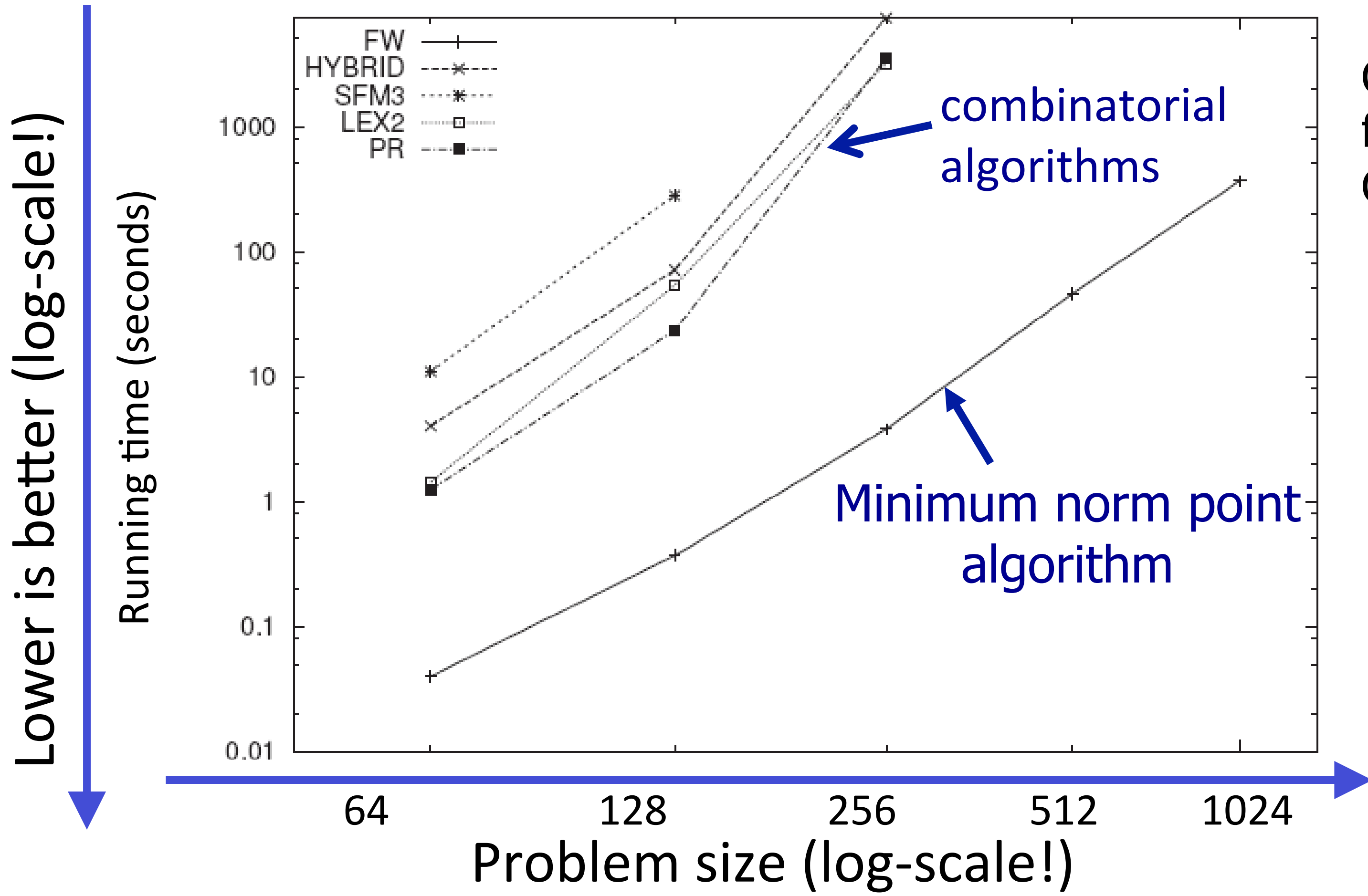
similar: optimization over B_F

➔ can find u^*

(Frank-Wolfe algorithm)

Fujishige '91, Fujishige & Isotani '11

Empirical comparison



Cut functions from DIMACS Challenge

Minimum norm point algorithm: usually orders of magnitude faster

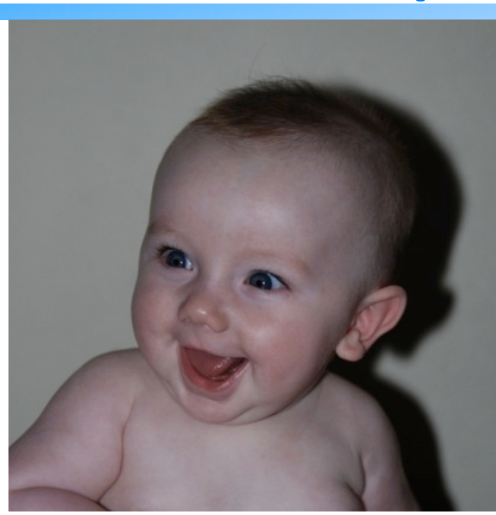
[Fujishige & Isotani '11]

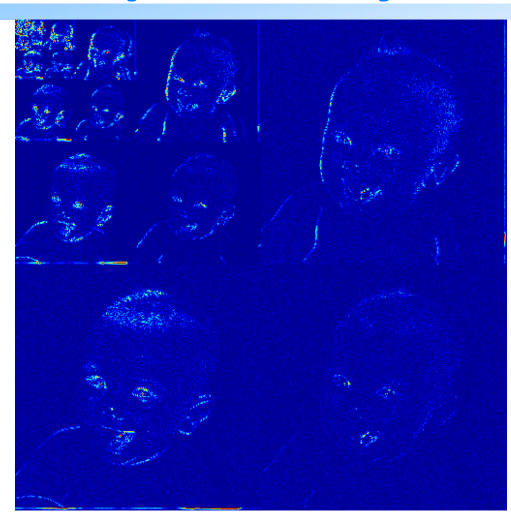
Applications?

Example I: Sparsity

d

pixels



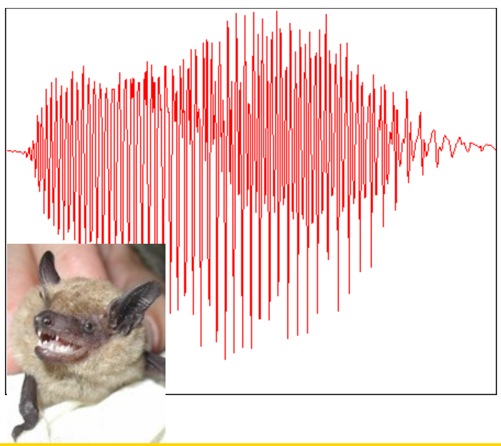


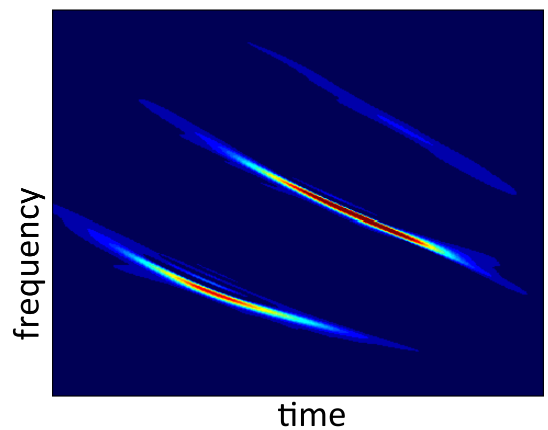
$k \ll d$

large wavelet coefficients

d

wideband signal samples





$k \ll d$

large Gabor (TF) coefficients

Many natural signals sparse in suitable basis.
Can exploit for learning/regularization/compressive sensing...

Sparse reconstruction

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

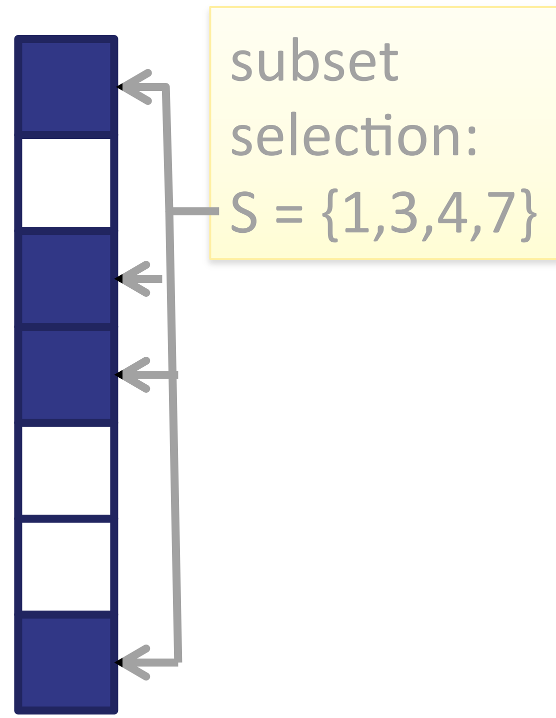
- explain y with few columns of M: few x_i

discrete regularization on support S of x

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

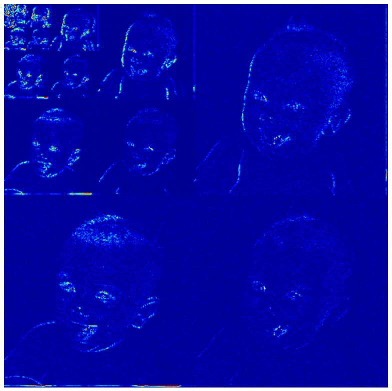
relax to convex envelope

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

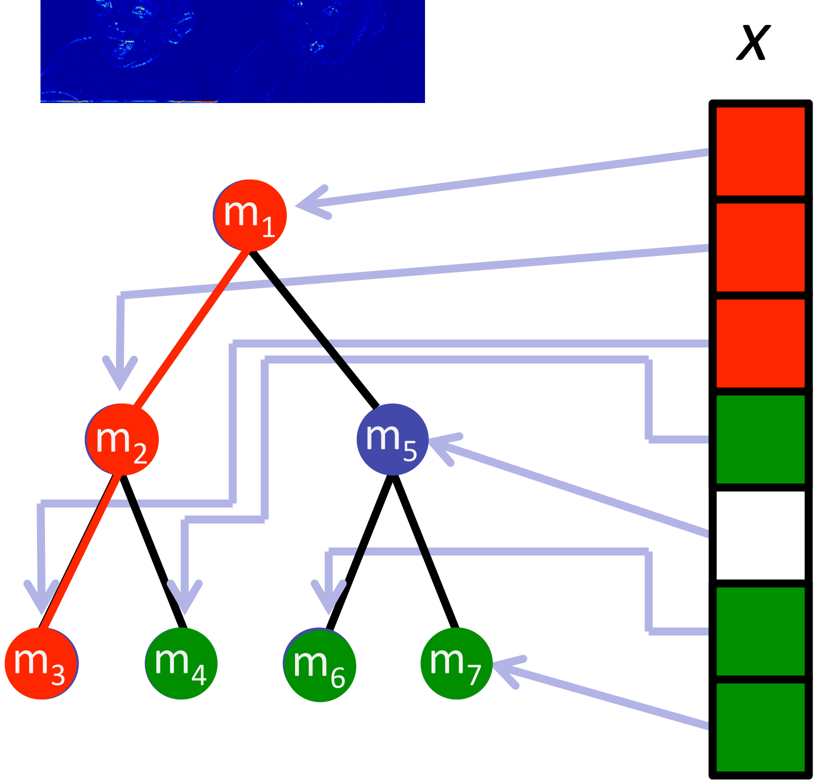


in nature: sparsity pattern often not random…

Structured sparsity



Incorporate tree preference in regularizer?



Set function:

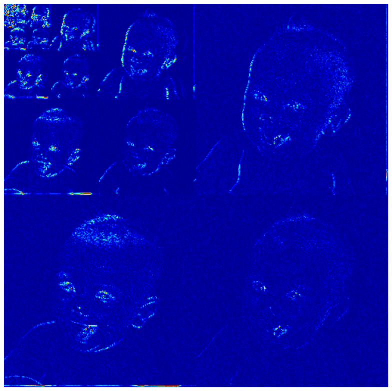
$$F(T) < F(S)$$

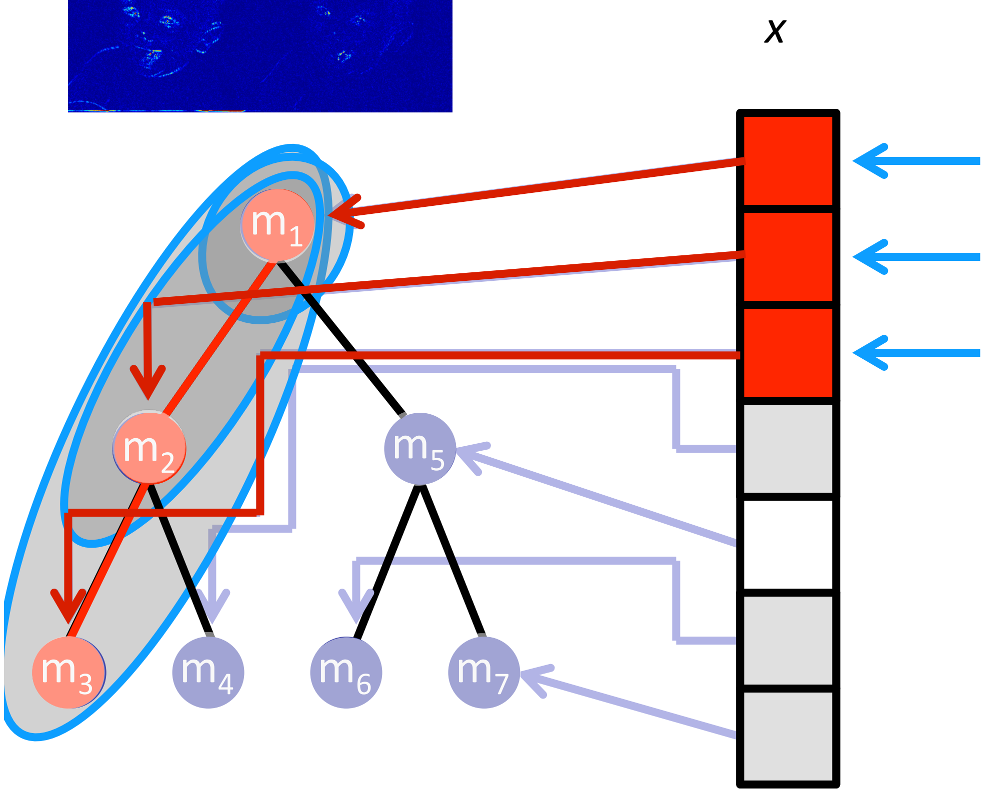
if T is a tree and S not

$|S| = |T|$

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

Structured sparsity





Incorporate tree preference in regularizer?

Set function:

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

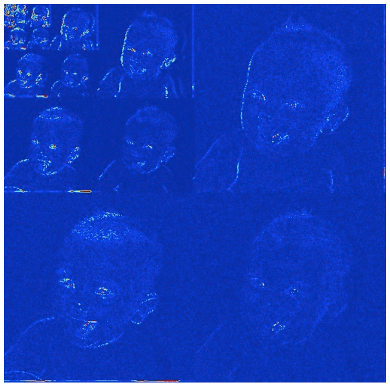
If T is a tree and S not,

$|S| = |T|$

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

$$F(T) = 3$$

Structured sparsity



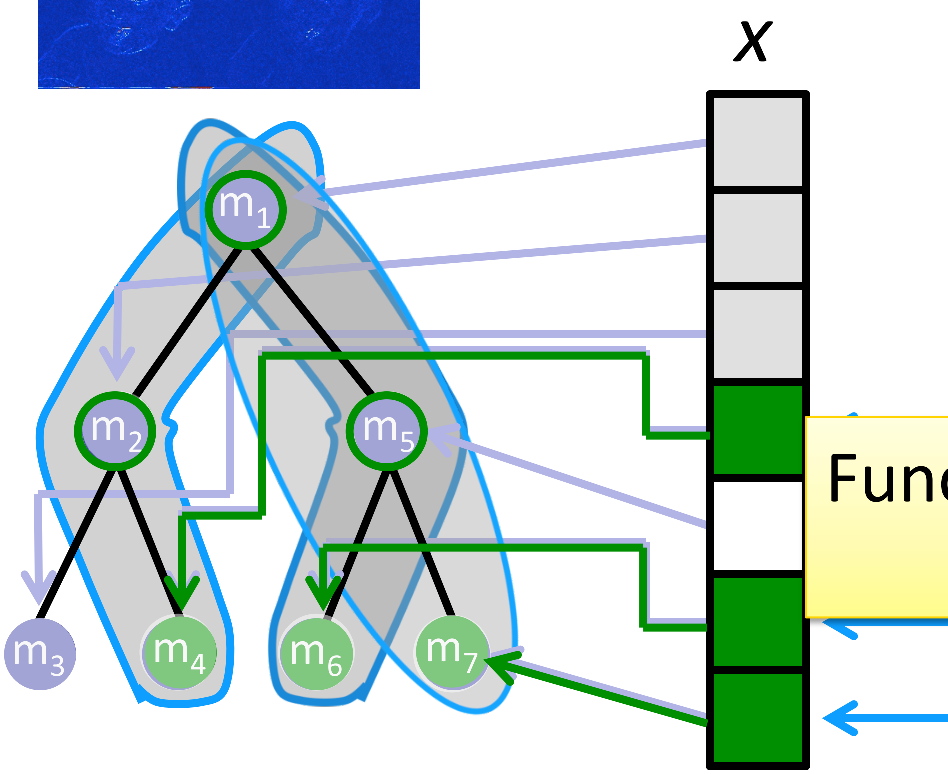
Incorporate tree preference in regularizer?

Set function:

$$F(\color{red}{T}) < F(\color{green}{S})$$

If *T* is a tree and *S* not,

$|S| = |T|$



Function *F* is ...

submodular! ☺

$$F(\color{red}{T}) = 3$$

Sparsity

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

- explain *y* with few columns of *M*: few x_i

- prior knowledge: patterns of nonzeros

discrete regularization on support *S* of *x*

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

- submodular function

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

relax to convex envelope

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

➔ Lovász extension

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

- Optimization: submodular minimization

[Bach`10]

Further connections: Dictionary Selection

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

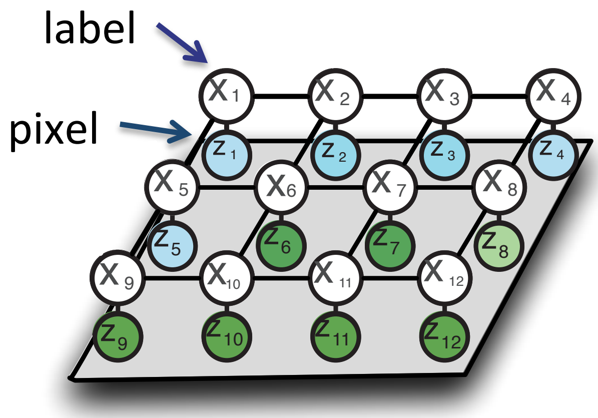
Where does the dictionary M come from?

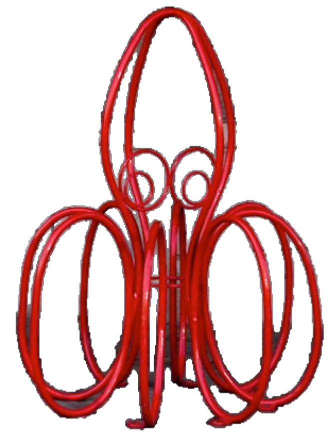
Want to learn it from data: $\{y_1, \ldots, y_n\} \subseteq \mathbb{R}^d$

Selecting a dictionary with near-max. variance reduction
⇔ Maximization of approximately submodular function

[Krause & Cevher '10; Das & Kempe '11]

Example: MAP inference





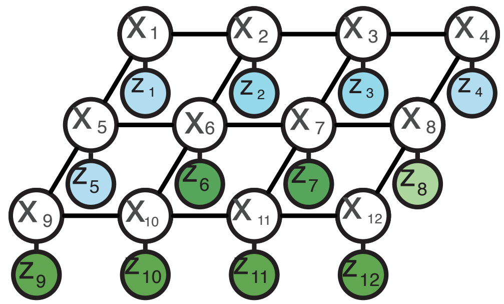
$$\max_{\mathbf{x}\in\{0,1\}^n} P(\mathbf{x} \mid \mathbf{z}) \propto \exp(-E(\mathbf{x};\mathbf{z}))$$

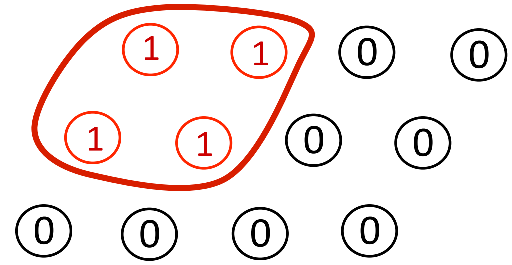
labels

pixel values

$$\Leftrightarrow \quad \min_{\mathbf{x}\in\{0,1\}^n} E(\mathbf{x};\mathbf{z})$$

Example: MAP inference



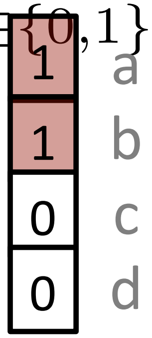


Recall: equivalence

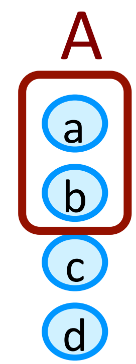
function on binary vectors $\qquad$ set function

$$\max_{\mathbf{x}\in\{0,1\}^n} P(\mathbf{x} \mid \mathbf{z}) \propto \exp(-E(\mathbf{x};\mathbf{z}))$$

$x = e_A$



$$E(e_A;\mathbf{z}) = F(A)$$



if F is submodular (attractive potentials), then $\iff \min_{\mathbf{x}\in\{0,1\}^n} E(\mathbf{x};\mathbf{z})$

MAP inference = submodular minimization!

polynomial-time

Special cases

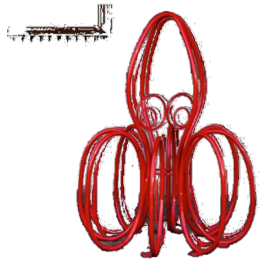
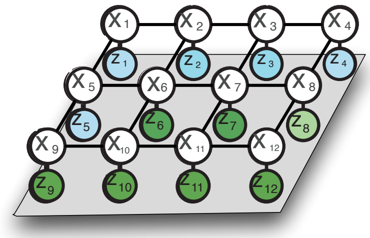
Minimizing general submodular functions:

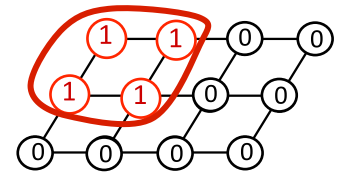
poly-time, but not very scalable

Special structure → faster algorithms

- Symmetric functions
- Graph cuts
- Concave functions
- Sums of functions with bounded support
- ...

MAP inference



$$\min_{\mathbf{x}\in\{0,1\}^n} E(\mathbf{x};\mathbf{z}) = \sum_i E_i(x_i) + \sum_{ij} E_{ij}(x_i, x_j) \quad \equiv \min_{A\subseteq V} F(A)$$

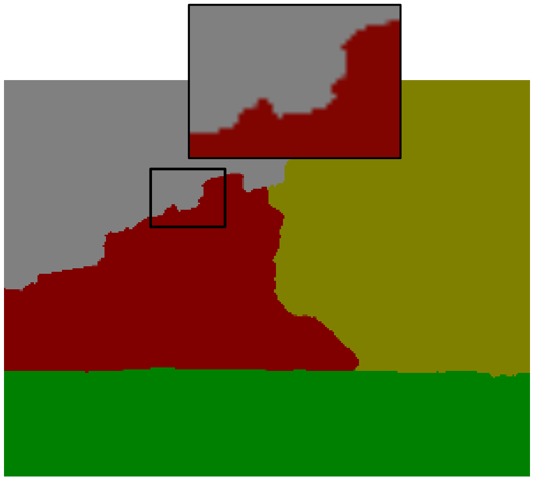
if each E_{ij} is submodular:

$$\underbrace{E_{ij}(1,0)}_{a} + \underbrace{E_{ij}(0,1)}_{b} \geq E_{ij}(0,0) + \underbrace{E_{ij}(1,1)}_{a\ b}$$

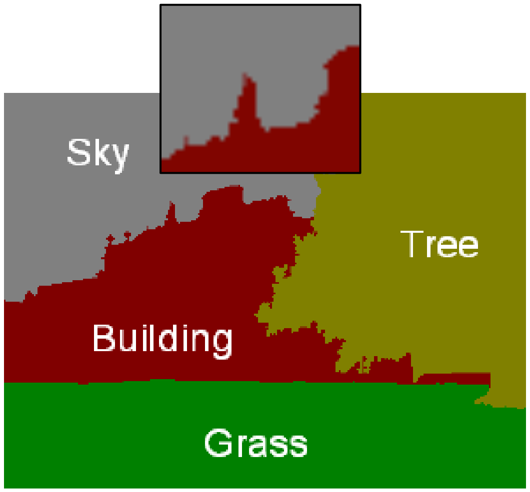
then F is a graph cut function.

$\boxed{\text{MAP inference = Minimum cut: fast ☺}}$

Pairwise is not enough…



color + pairwise



color + pairwise +

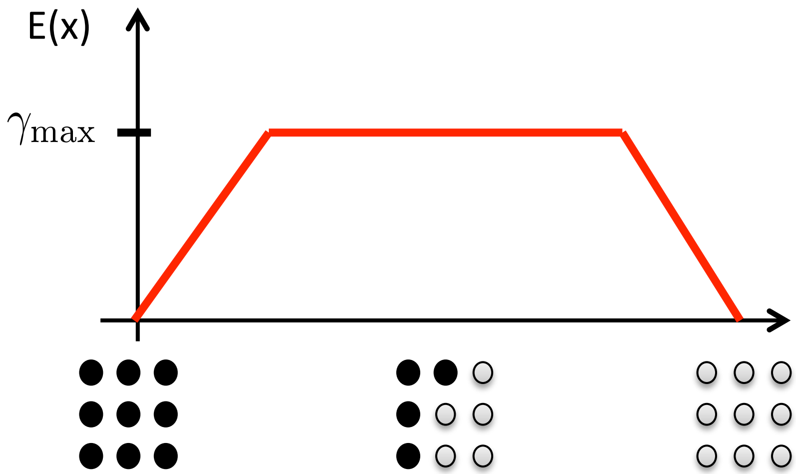
$$E(x) = \sum_i E_i(x_i) + \sum_{ij} E_{ij}(x_i, x_j)$$

Pixels in one tile should have the same label

[Kohli et al.`09]

Enforcing label consistency

Pixels in a superpixel should have the same label



concave function of cardinality ➔ submodular ☺

> 2 arguments: Graph cut ??

Higher-order functions as graph cuts?

$$\sum_i E_i(x_i) + \sum_{ij} E_{ij}(x_i, x_j) \quad \boxed{+ \sum_c E_c(x_c)}$$

General strategy:
reduce to pairwise case by adding auxiliary variables

- works well for some particular $E_c(x_c)$
  [Billionet & Minoux `85, Freedman & Drineas `05, Živný & Jeavons `10,...]

- necessary conditions complex and
  not all submodular functions equal such graph cuts [Živný et al.'09]

Fast approximate minimization

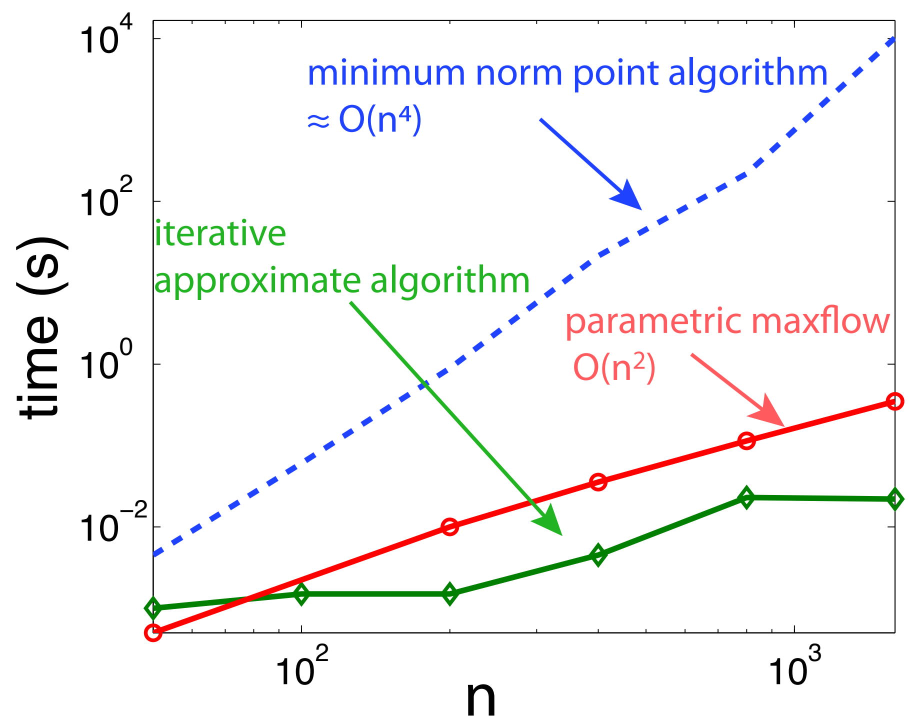
- Not all submodular functions can be optimized as graph cuts
- Even if they can: possibly many extra nodes in the graph ☹

Other options?

- minimum norm algorithm
- other special cases:
  e.g. parametric maxflow
  [Fujishige & Iwata`99]

Approximate! ☺
Every submodular function can be approximated by a series of graph cut functions [Jegelka, Lin & Bilmes `11]

speech corpus selection [Lin&Bilmes `11]



Fast approximate minimization

- Not all submodular functions can be optimized as graph cuts
- Even if they can: possibly many extra nodes in the graph ☹

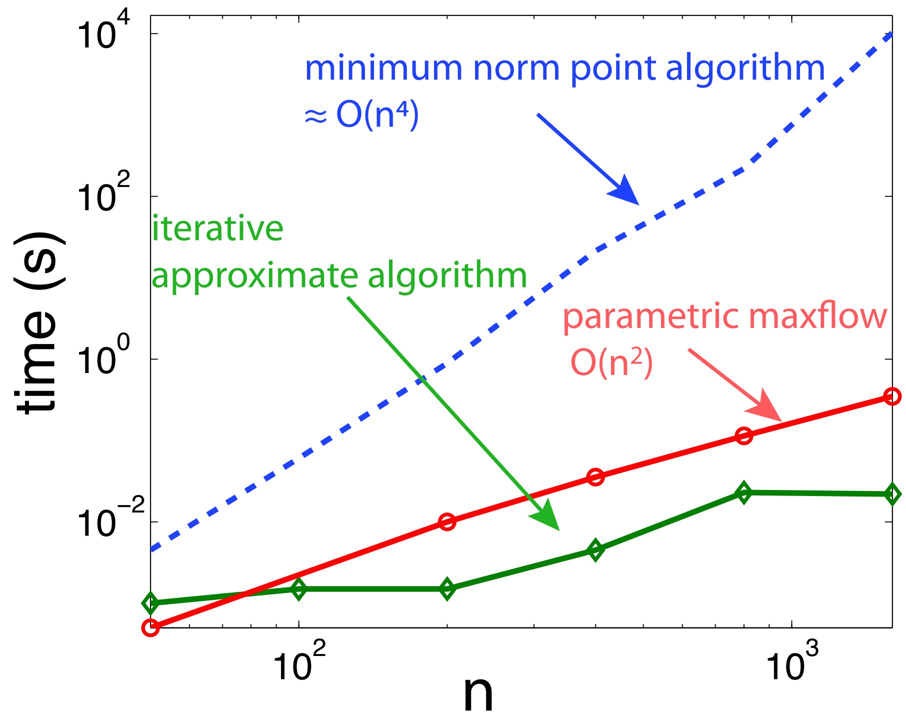
Approximate! ☺

decompose:

- represent as much as possible exactly by a graph
- rest: approximate iteratively by changing edge weights

solve a series of cut problems

speech corpus selection [Lin&Bilmes `11]



Other special cases

- Symmetric: $F(S) = F(V \setminus S)$
  - Queyranne's algorithm: O(n^3)

[Queyranne, 1998]

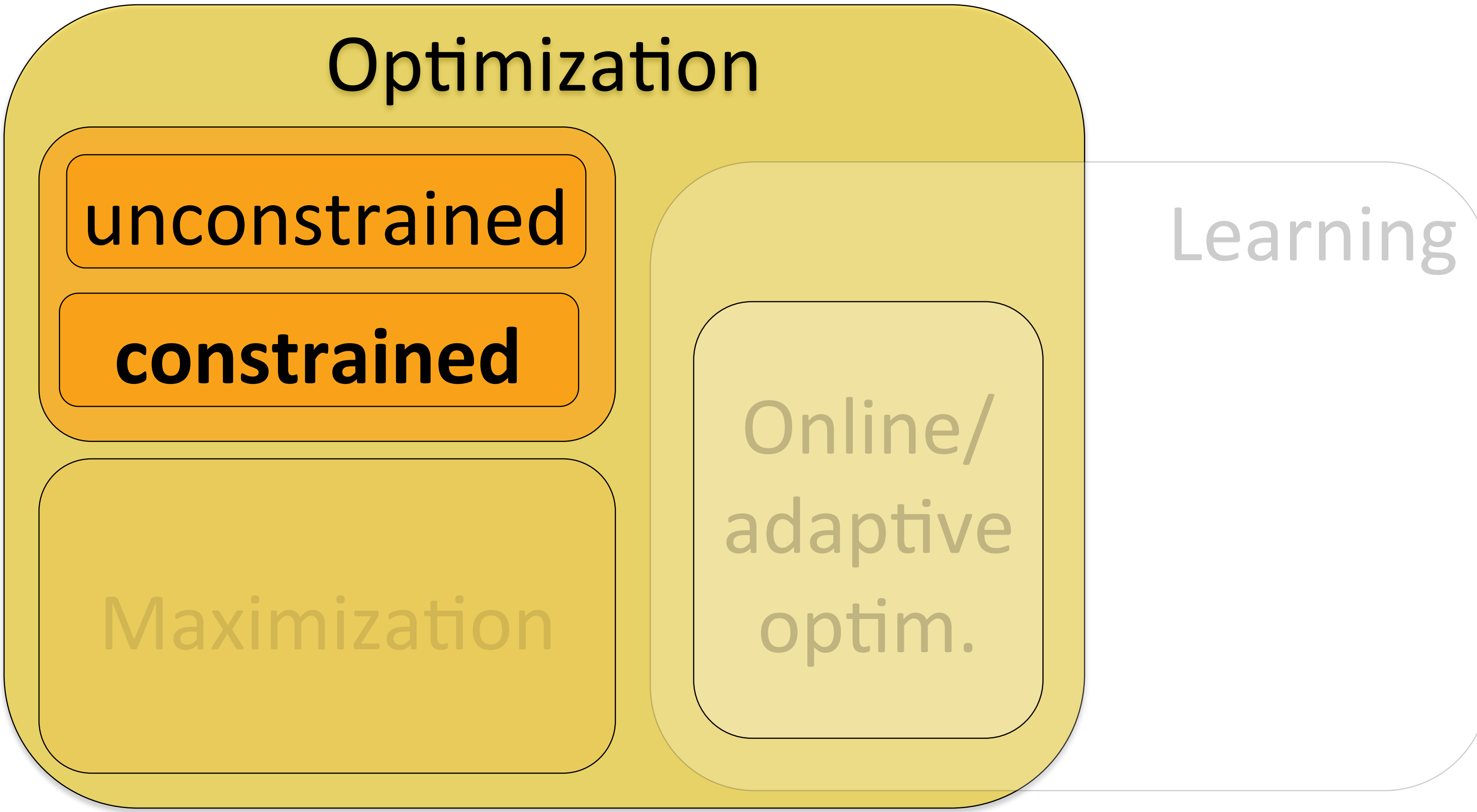
- Concave of modular: $$F(S) = \sum_i g_i\Big(\sum_{s \in S} w(s)\Big)$$

[Stobbe & Krause `10, Kohli et al, `09]

- Sum of submodular functions, each bounded support

[Kolmogorov `12]

Submodular minimization



Submodular minimization

- unconstrained: $\min\ F(A) \quad \text{s.t. } A \subseteq V$
  - nontrivial algorithms, polynomial time

special case: balanced cut

- constraints: e.g. $\min\ F(A) \quad \text{s.t. } |A| \geq k$
  - limited cases doable: $\ldots$
    odd/even cardinality, inclusion/exclusion of a set

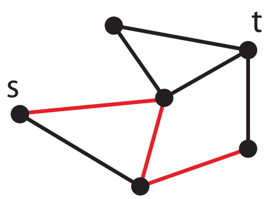
General case: NP hard
- hard to approximate within polynomial factors!
- But: special cases often still work well

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

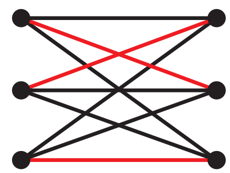
Constraints

minimum...

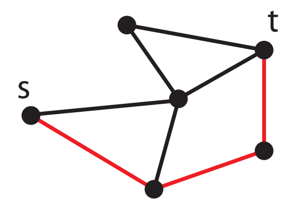
cut



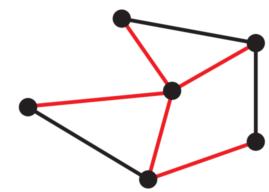
matching



path



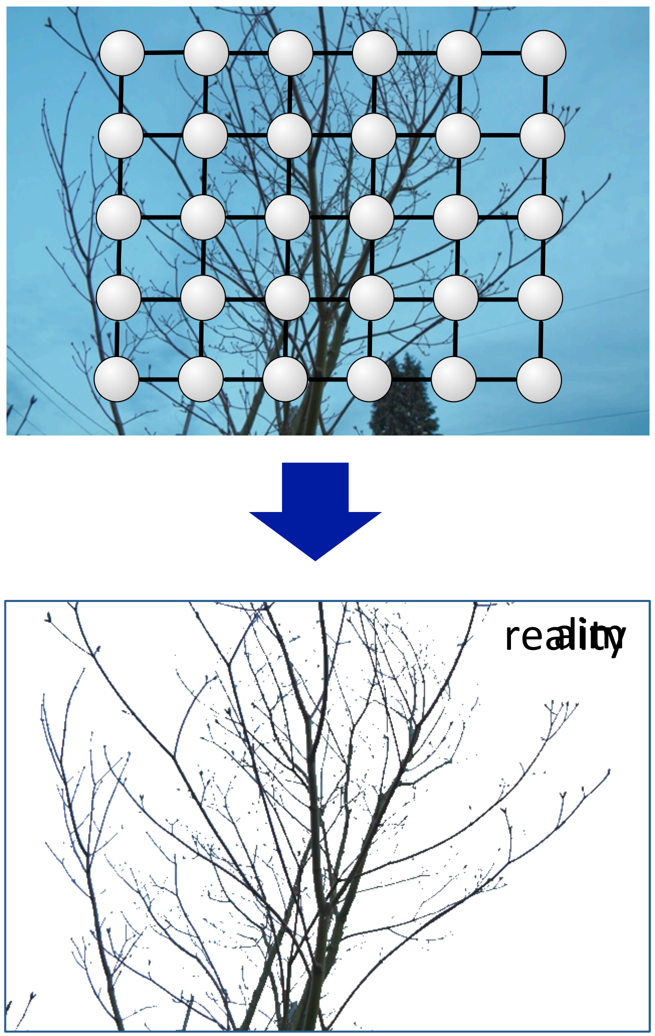
spanning tree



ground set: edges in a graph

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

Recall: MAP and cuts

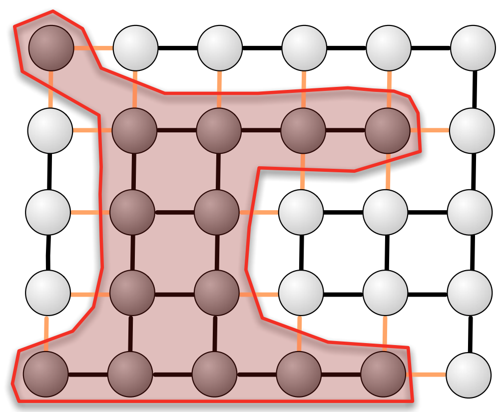


binary labeling: $x = e_A$

pairwise random field:

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

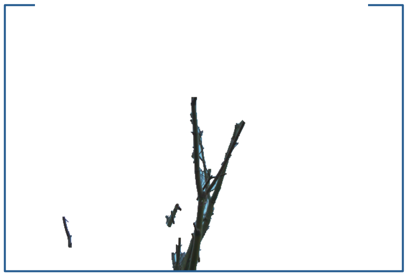
What's the problem?



minimum cut: prefer
short cut = short object boundary

MAP and cuts

Minimum cut



Minimum cooperative cut

implicit criterion:
short cut =
short boundary

minimize
sum of edge weights

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

new criterion:
boundary may be long if the
boundary is homogeneous

minimize
submodular function of edges

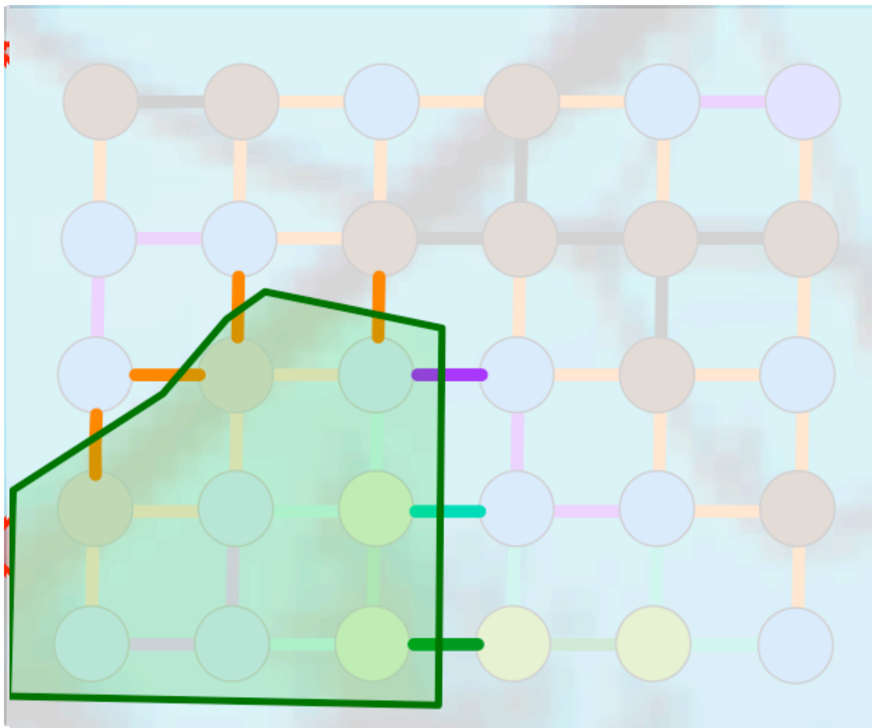
$$F(C)$$

not a sum of
edge weights!

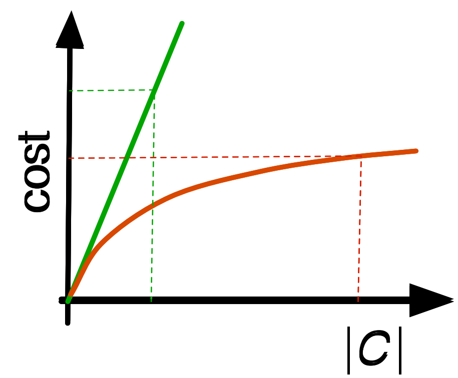
Reward co-occurrence of edges

sum of weights:
use few edges

submodular cost function:
use few groups S_i of edges



$$F(C) = \sum_i F_i(C \cap S_i)$$



25 edges, 1 type

7 edges, 4 types

Results

Graph cut

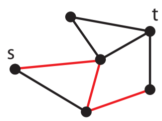
Cooperative cut

Optimization?

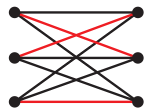
- not a standard graph cut
- MAP viewpoint:
  global, non-submodular energy function

Constrained optimization

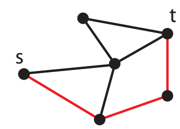
cut



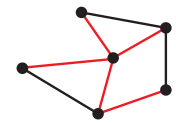
matching



path



spanning tree



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

approximate optimization

convex relaxation

minimize surrogate function

approximation bounds dependent on *F*:

polynomial – constant – FPTAS

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

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

Efficient constrained optimization

minimize a series of surrogate functions

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

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

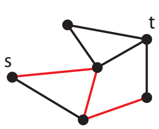
2. Solve easy sum-of-weights problem:

$$S^i = \arg\min_{S \in \mathcal{C}} \hat{F}^i(S)$$

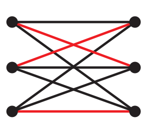
and repeat.

- efficient
- only need to solve sum-of-weights problems
- unifying viewpoint of submodular min and max
  see Wed best student paper talk

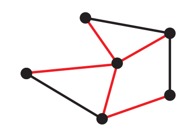
cut



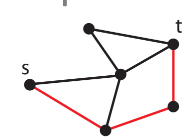
matching



spanning tree



path



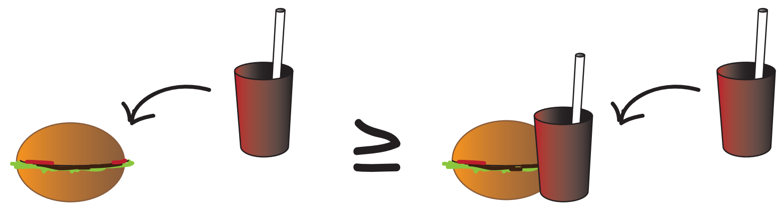
[Jegelka & Bilmes `11, Iyer et al. ICML `13]

Submodular min in practice

- Does a special algorithm apply?
  - symmetric function? graph cut? …. approximately?
- Continuous methods: convexity
  - minimum norm point algorithm
- Other techniques [not addressed here]
  - LP, column generation, …
- Combinatorial algorithms: relatively high complexity
- Constraints: hard
  - majorize-minimize or relaxation

Outline

- What is submodularity?
- Optimization
  - Minimize costs



  - Maximize utility
- Learning
- Learning for Optimization: new settings

Part I

Break!

Part II

see you in half an hour ☺