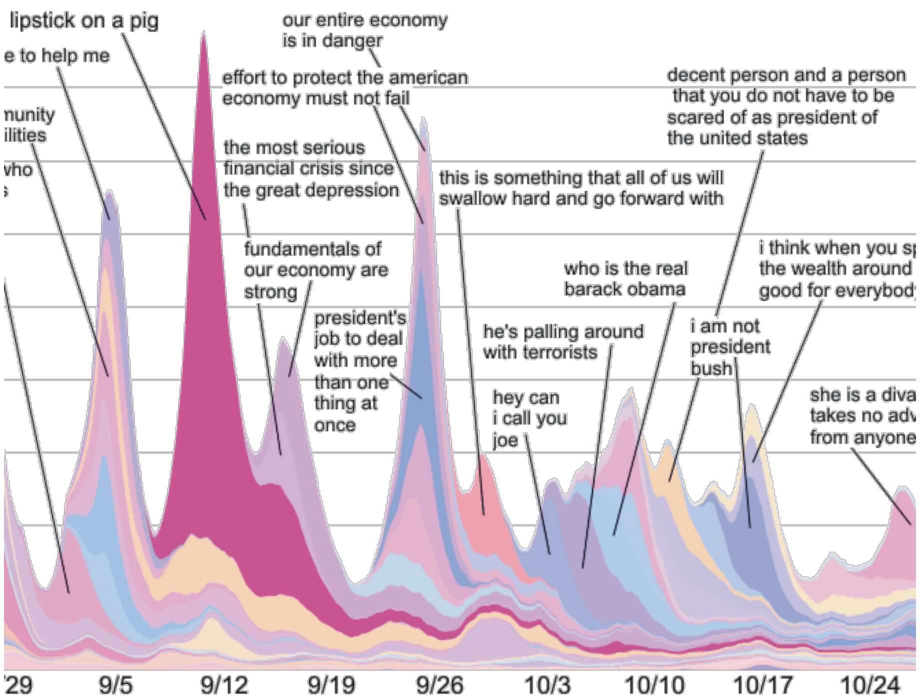
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

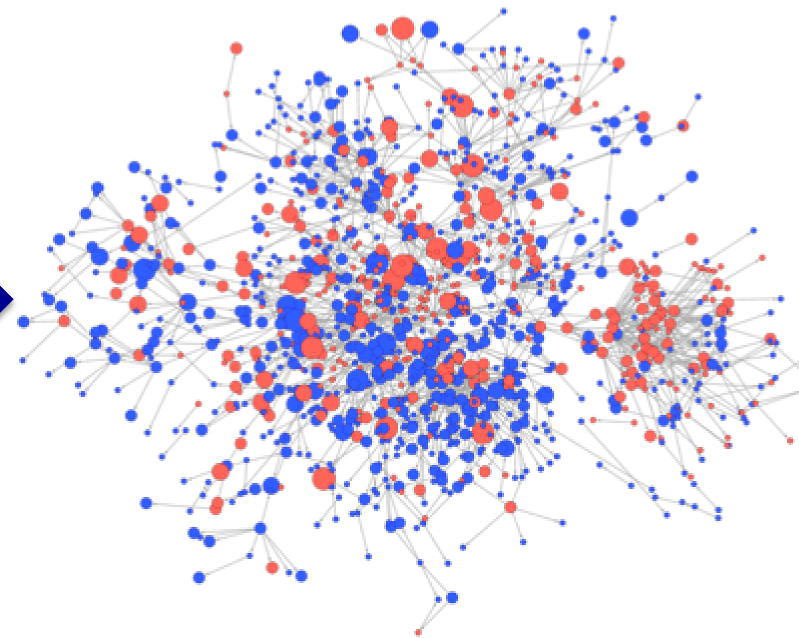
Submodularity in Machine Learning and Vision

Andreas Krause

British Machine Vision Conference (BMVC) 2013
Bristol, UK

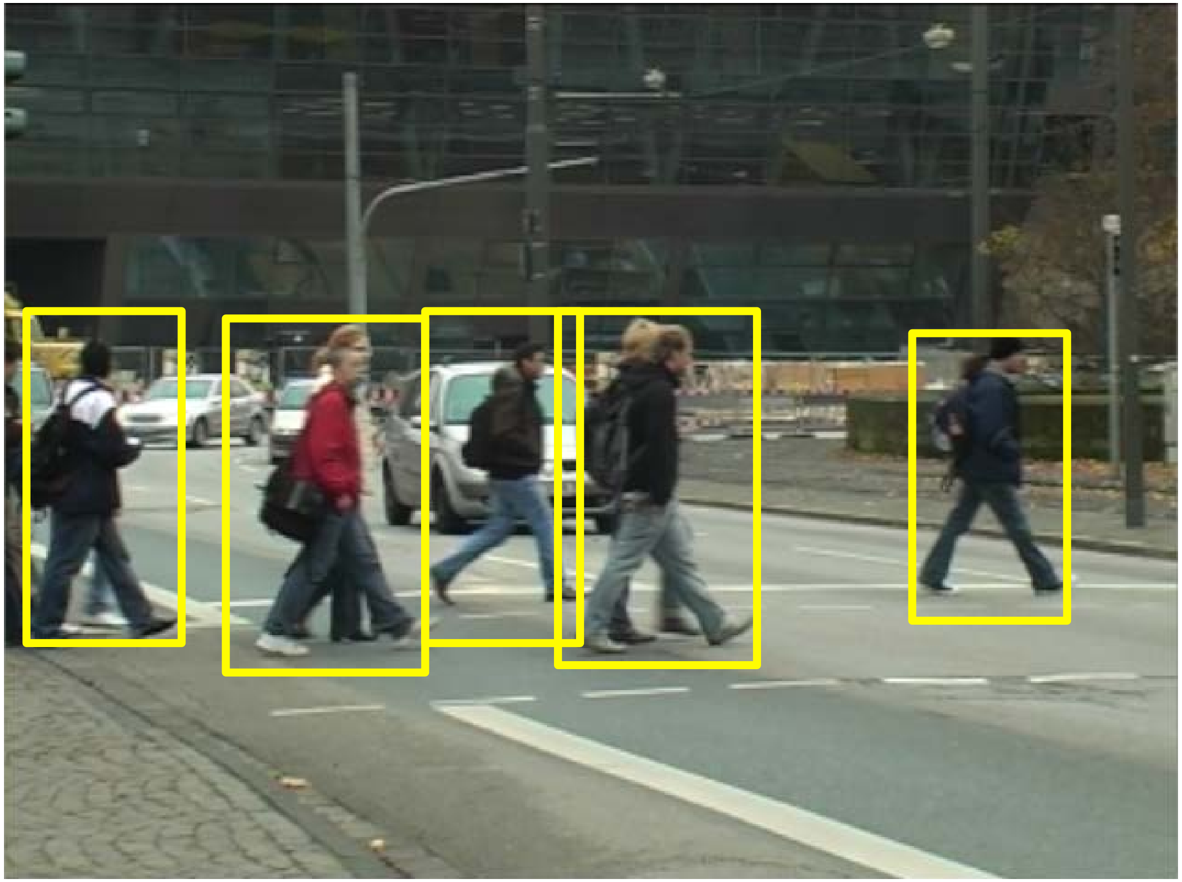
Network Inference





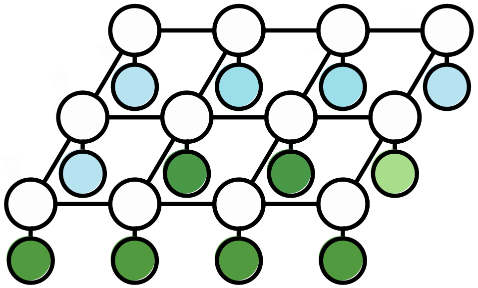
How to learn who influences whom?

Multiple object detection



How to locate multiple objects in an image?

Segmentation & MAP inference



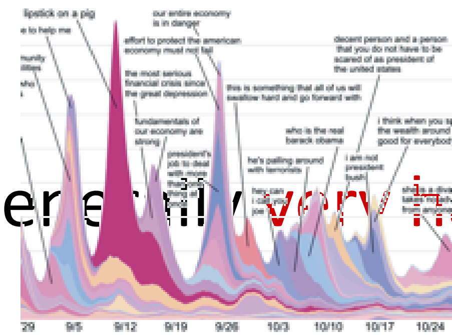
$$\max_x \; p(x \mid z)$$

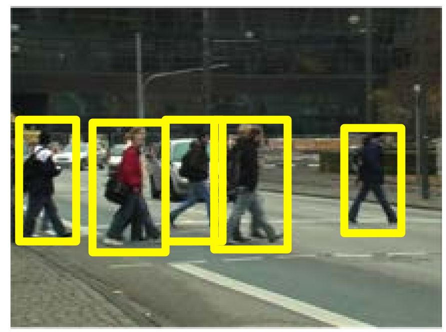
How find the MAP labeling in discrete graphical models *efficiently*?

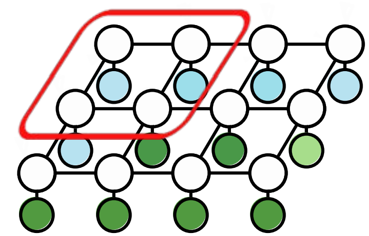
What's common?

- 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 complex structured learning problems

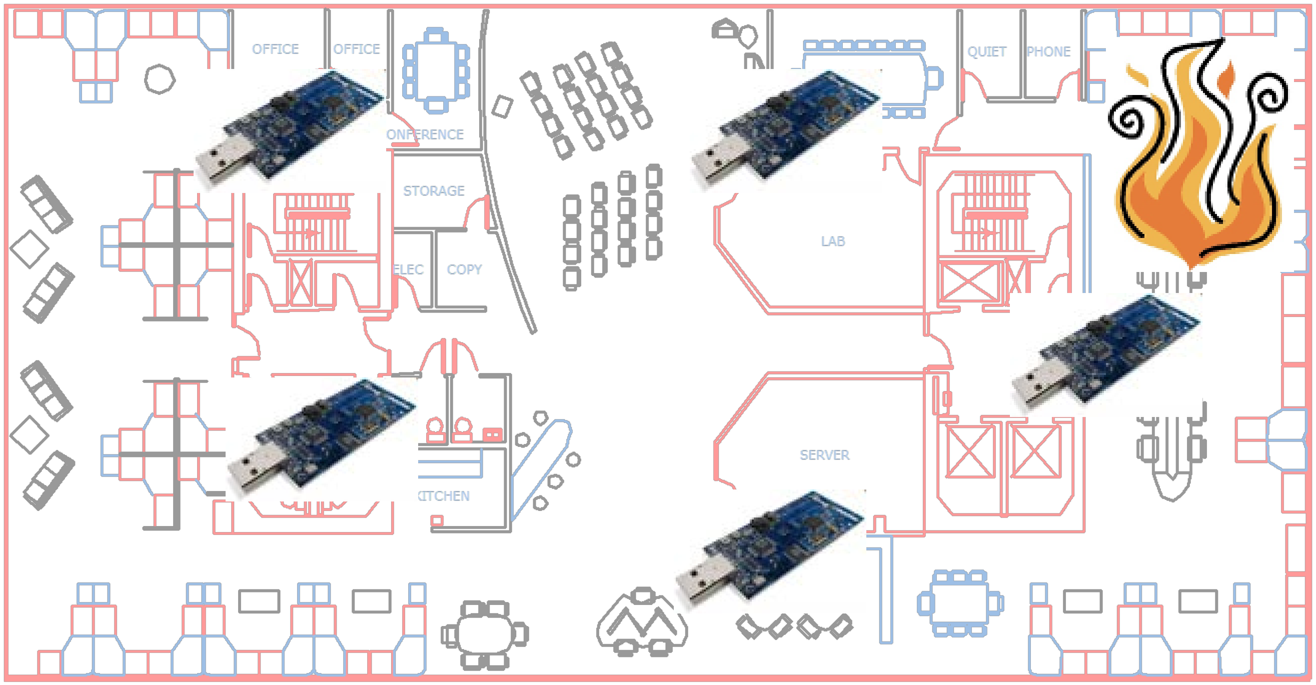
Outline

- What is submodularity?
- Optimization
  - Minimization
  - Maximization
- Applications
- Outlook and pointers

submodularity.org

slides, code, references, workshops, …

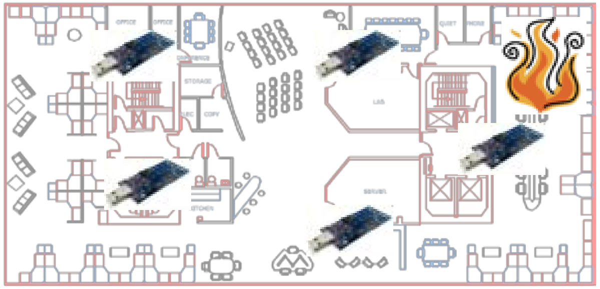
Example: placing sensors



Place sensors to monitor temperature

Set functions

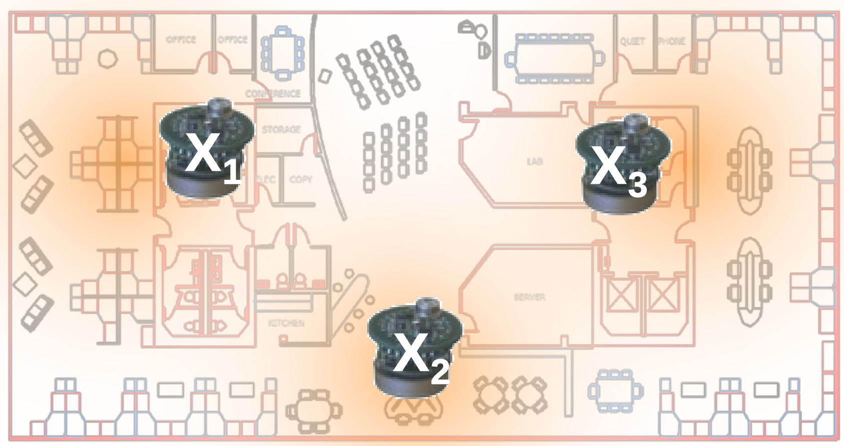
- finite ground set $V = \{1, 2, \ldots, n\}$
- set function $F : 2^V \to \mathbb{R}$



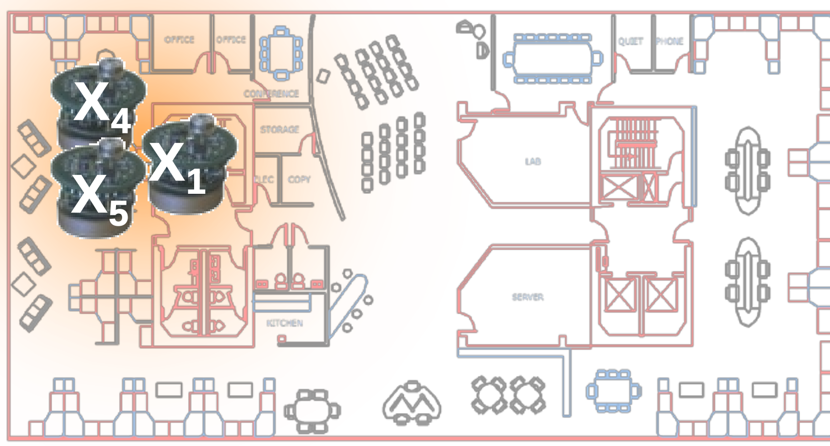
- will assume $F(\emptyset) = 0$ (w.l.o.g.)
- assume black box that can evaluate $F(A)$ for any $A \subseteq V$

Example: placing sensors

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



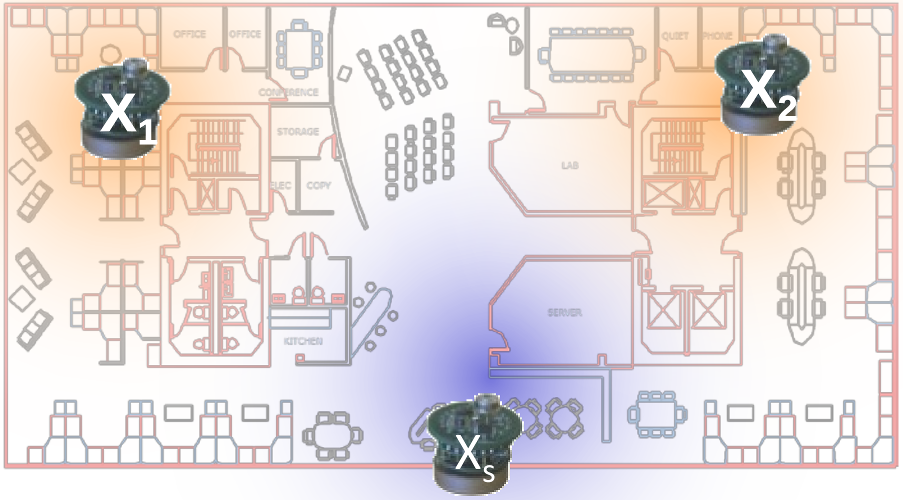
A={1,2,3}: Very informative
High value F(A)



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

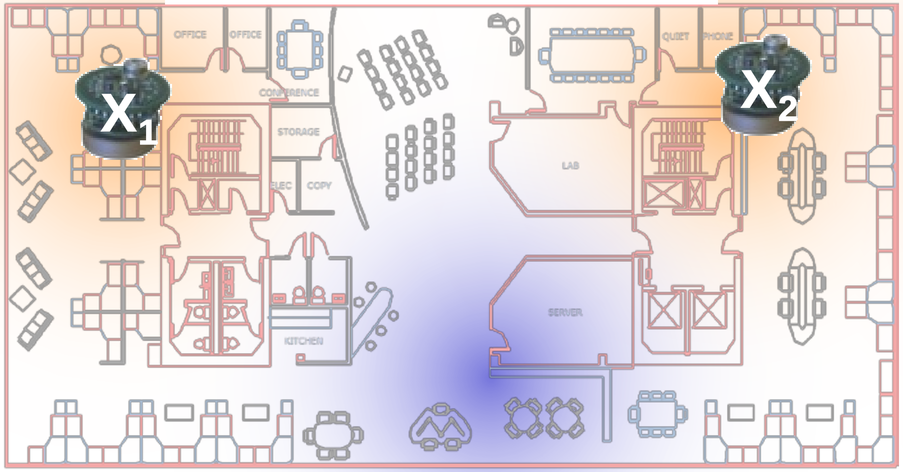
Marginal gain

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

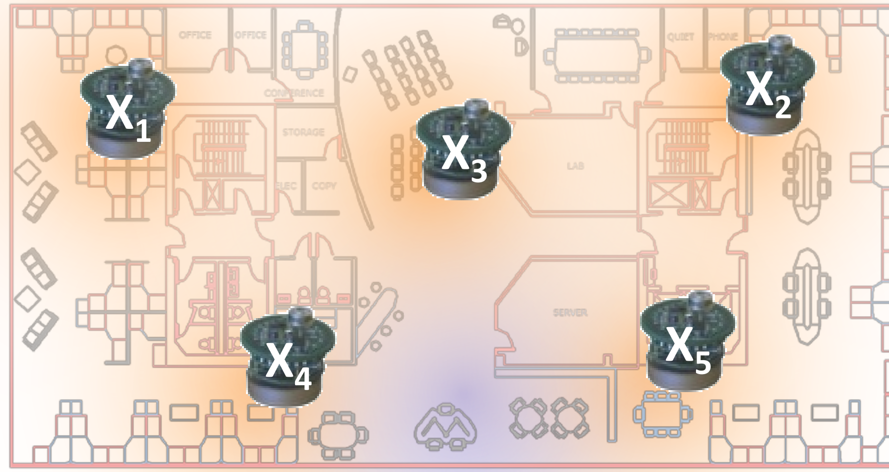


Decreasing gains: submodularity

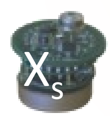
placement A = {1,2}



placement B = {1,…,5}



Big gain

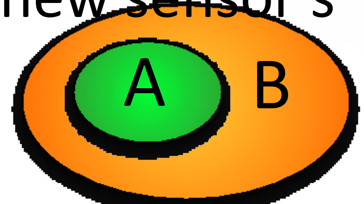


Adding *s* do … uch

small gain

new sensor s

+ • s



+ • s

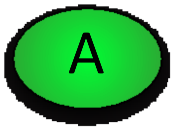
$$A \subseteq B \qquad F(A \cup s) - F(A) \quad \geq \quad F(B \cup s) - F(B)$$

$$s \notin B$$

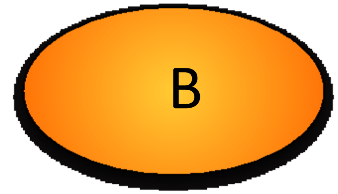
$$\Delta(s \mid A)$$

Equivalent characterizations

- Diminishing returns: for all $A \subseteq B$ and $s \notin B$



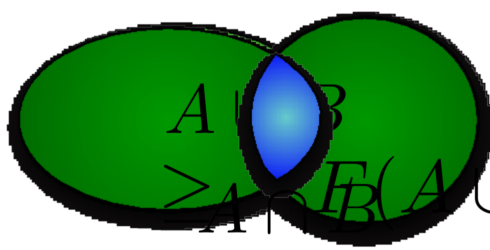
\+ • s



\+ • s

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

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



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

Submodular, modular & supermodular

A set function F is called

- supermodular if -F is submodular
- modular if F is both submodular and supermodular. Such functions can be written as

$$F(A) = \sum_{i \in A} w_i$$

Questions

How do I prove my problem is submodular?

Why is submodularity useful?

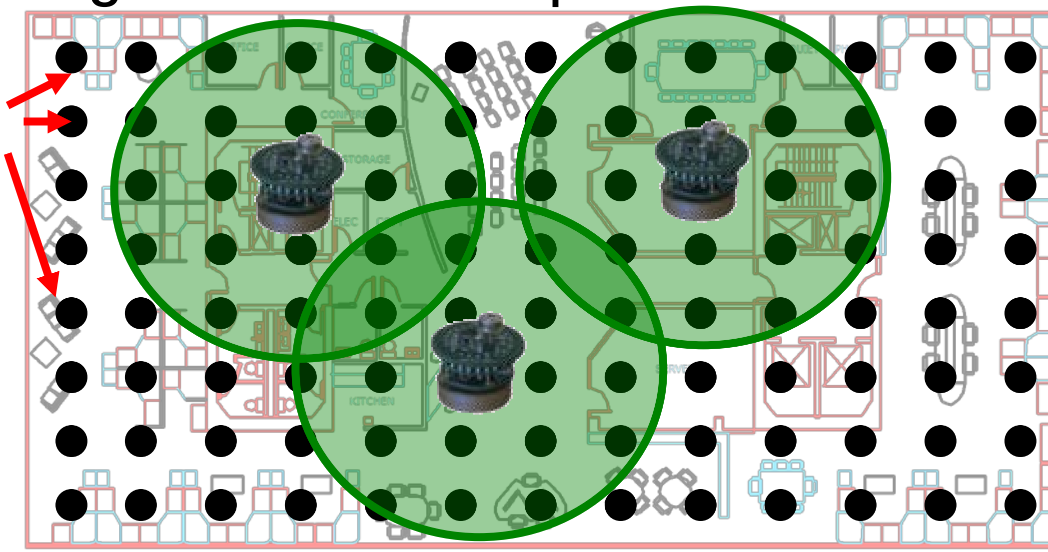
Example: Set cover

place sensors
in building

Node predicts
values of positions
with some radius

goal: cover floorplan with discs

Possible
locations
V



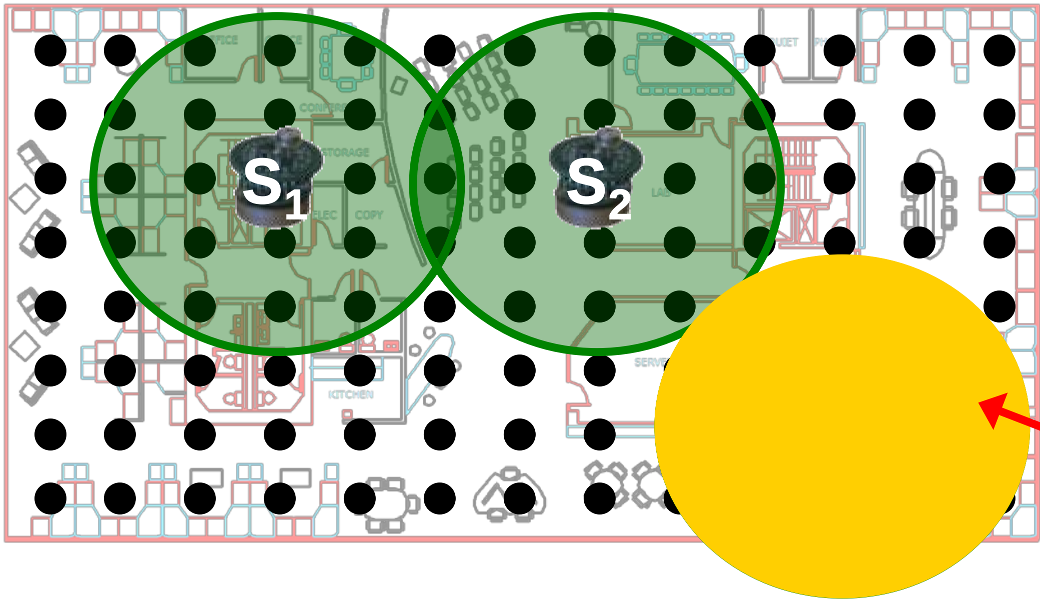
$A \subseteq V$: $F(A) =$
"area covered by sensors placed at A"

Formally:

Finite set W, collection of n subsets $S_i \subseteq W$

For $A \subseteq V$ define $F(A) = \left|\bigcup_{i \in A} S_i\right|$

Set cover is submodular

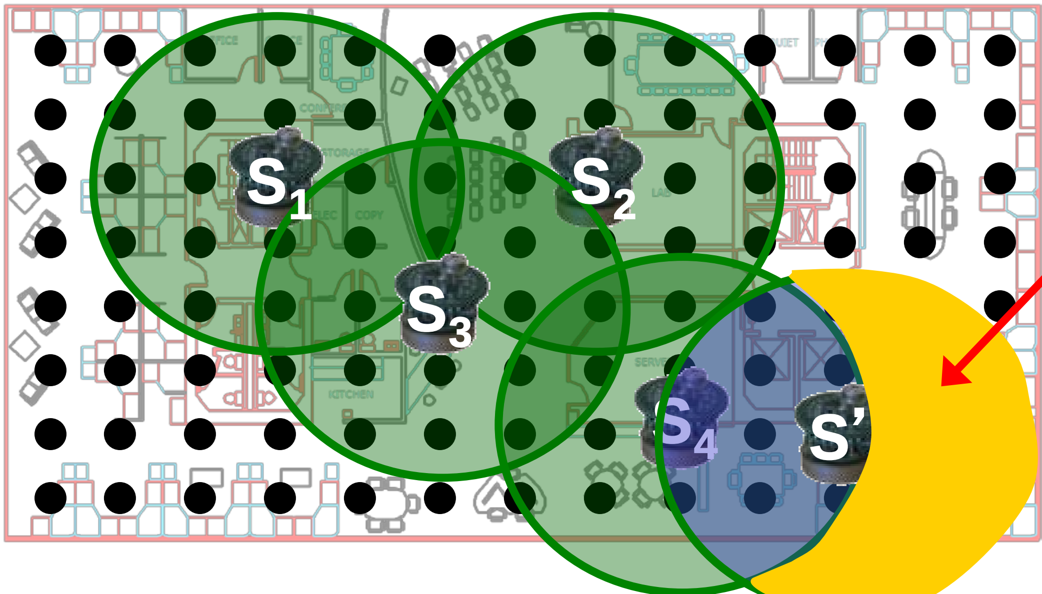


$A=\{s_1,s_2\}$

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

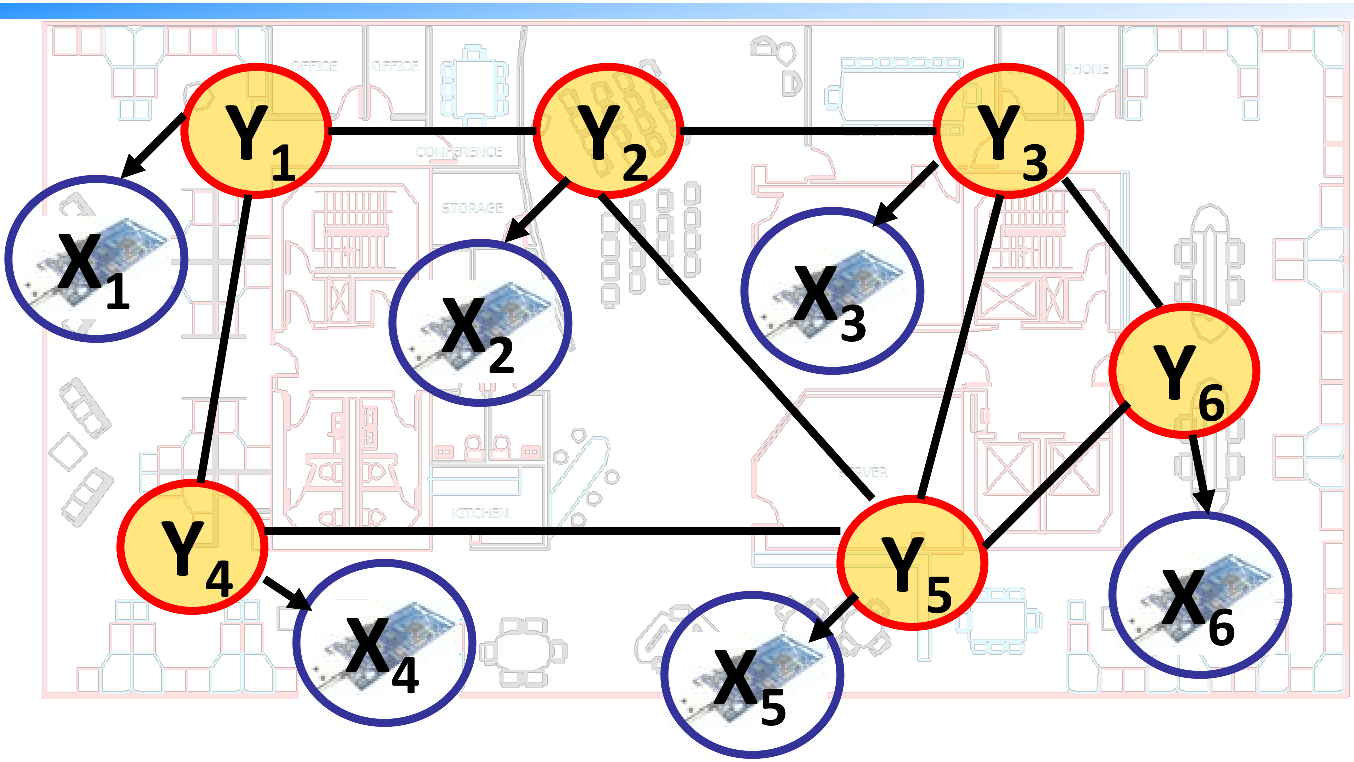
$\geq$

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



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

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

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

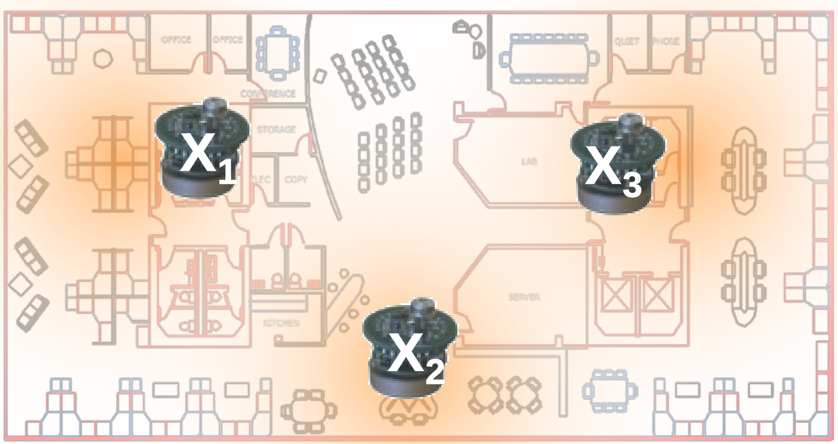
Example: Sensor placement

Utility of having sensors at subset A of all locations

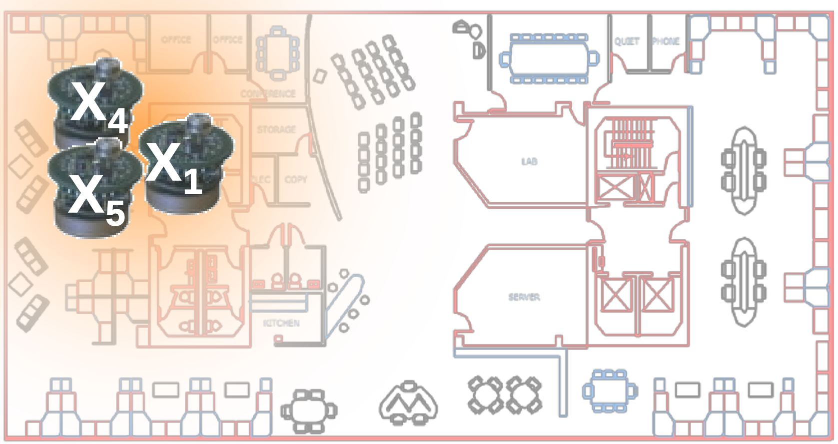
$$F(A) = H(\mathbf{Y}) - H(\mathbf{Y} \mid \mathbf{X}_A)$$

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)

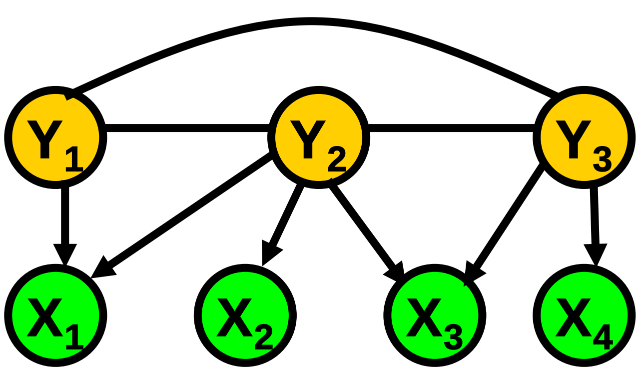
Submodularity of Information Gain

$Y_1,\ldots,Y_m$, X_1, …, X_n random variables

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

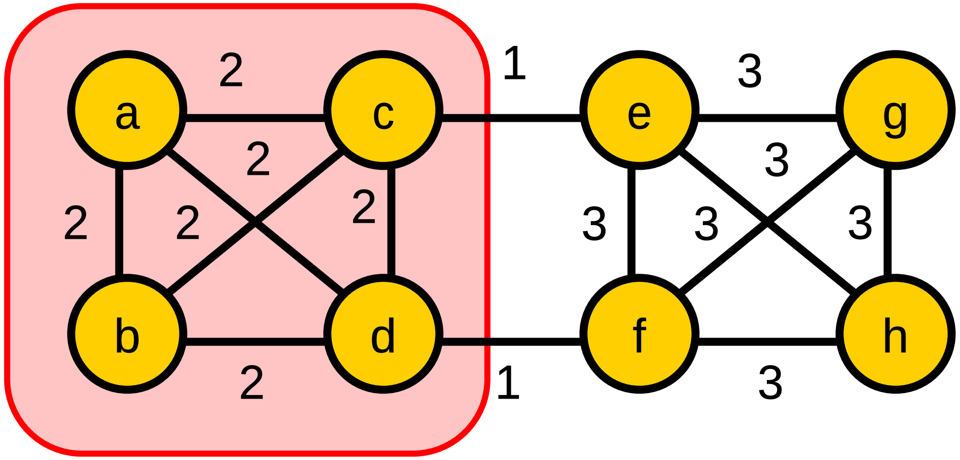
- F(A) is NOT always submodular

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



Proof:
"information never hurts"

Another example: Cut functions

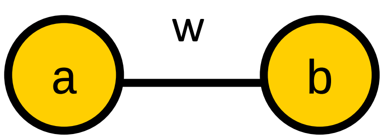


V={a,b,c,d,e,f,g,h}

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

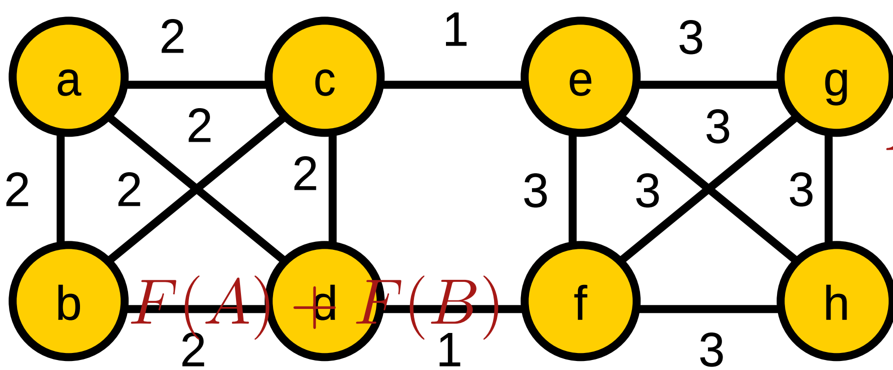
Cut function is submodular!

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 under linear combinations

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

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

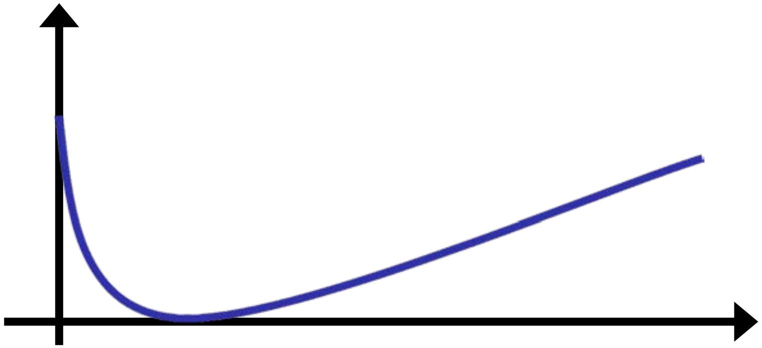
Submodularity closed under nonnegative linear combinations!

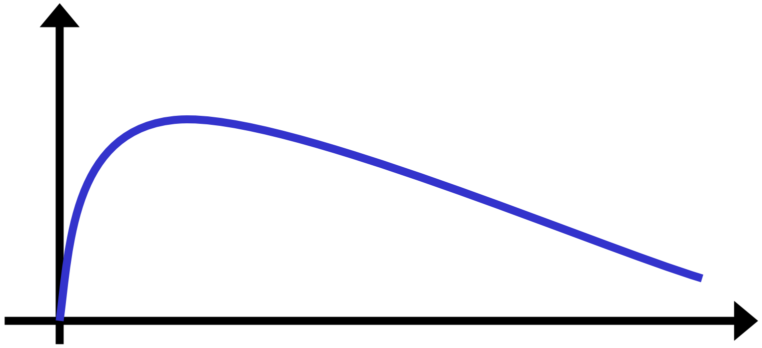
Extremely useful fact:

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

Submodularity …

discrete convexity ….

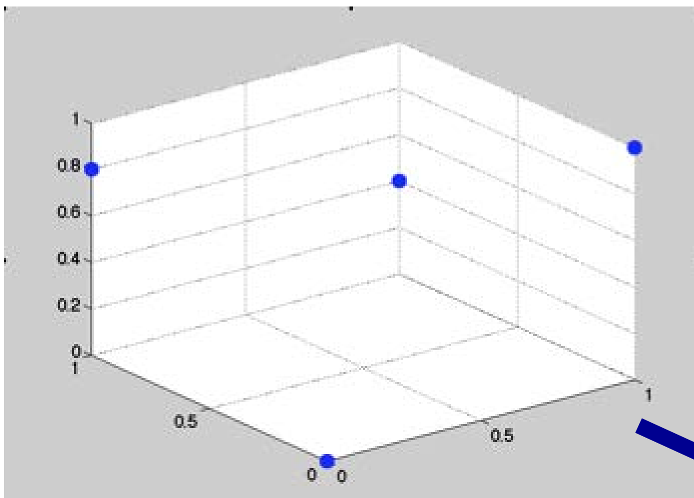


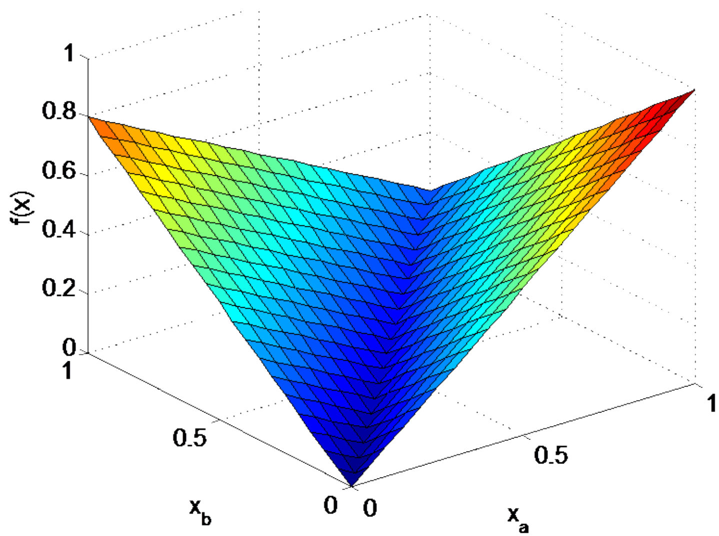


… or concavity?

Convex aspects

- convex extension
  - duality
  - efficient minimization





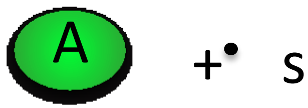
But this is only half of the story…

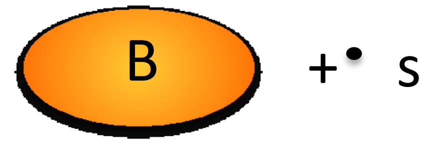
Concave aspects

- submodularity:

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

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

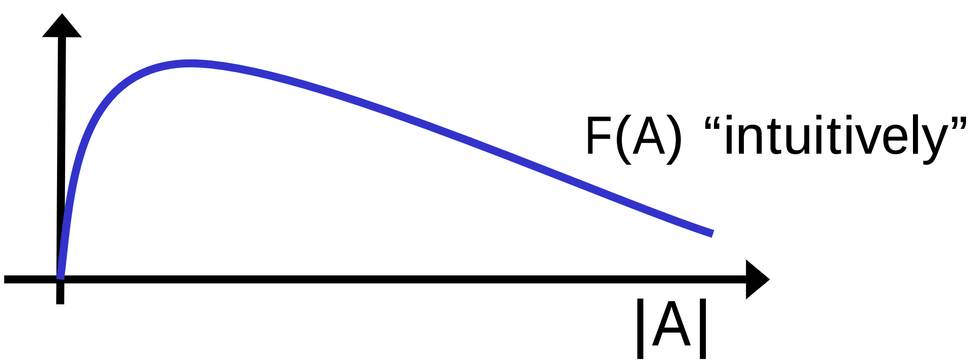




- concavity:

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

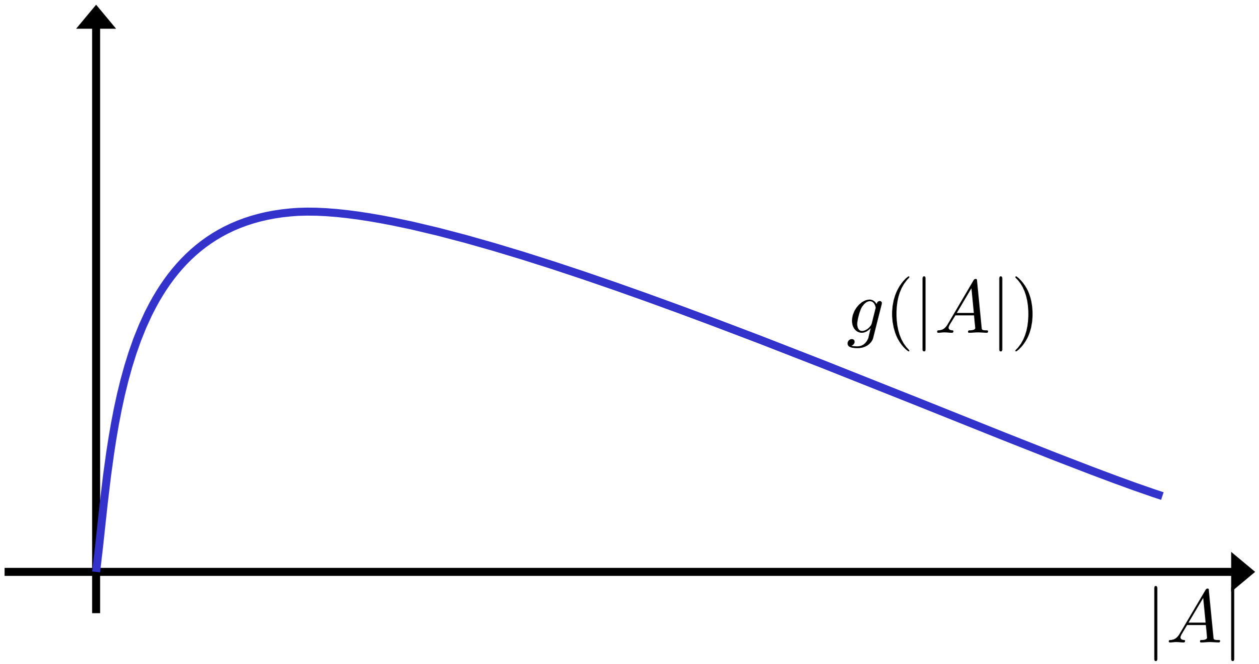
$$f(a+s) - f(a) \quad \geq \quad f(b+s) - f(b)$$



Submodularity and concavity

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

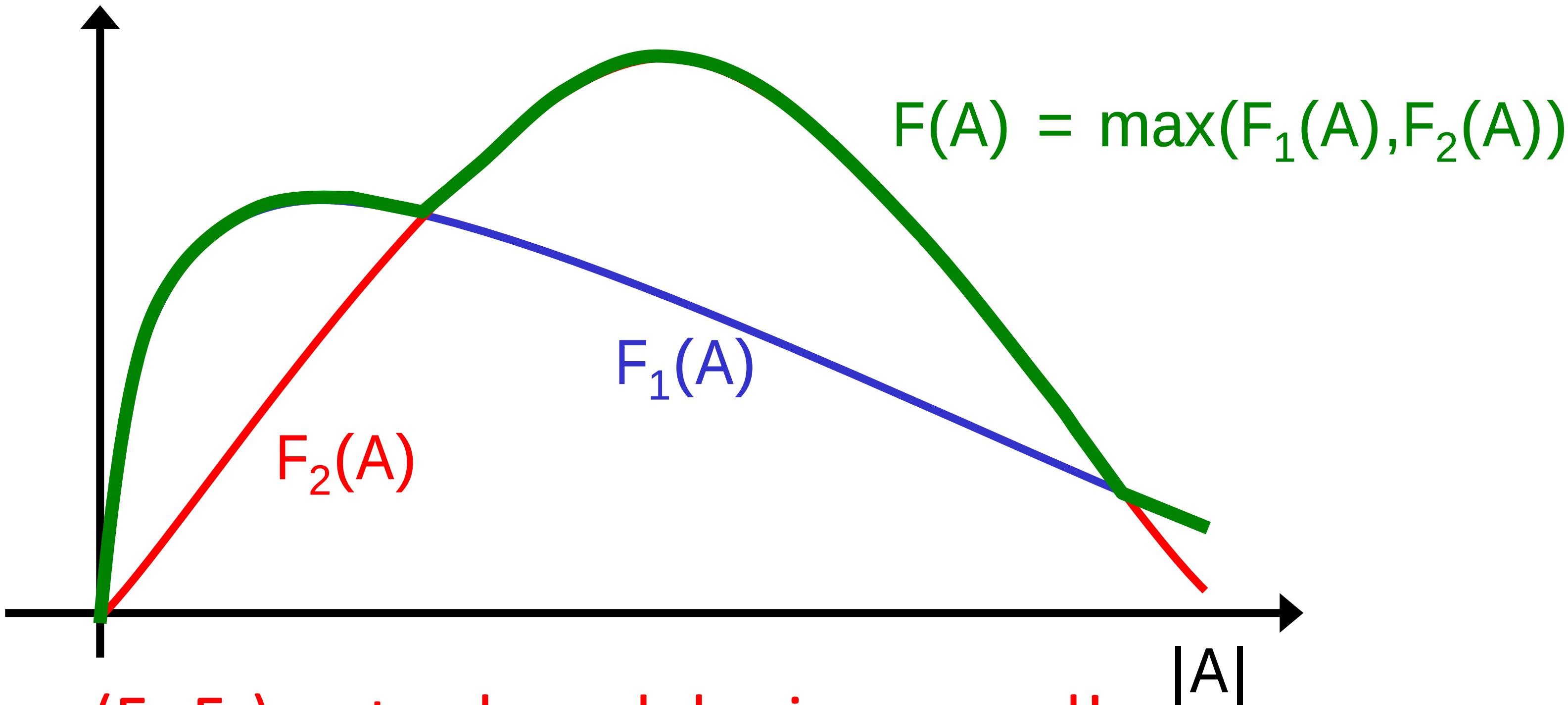
$F(A)$ submodular *if and only if* ... g is concave



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

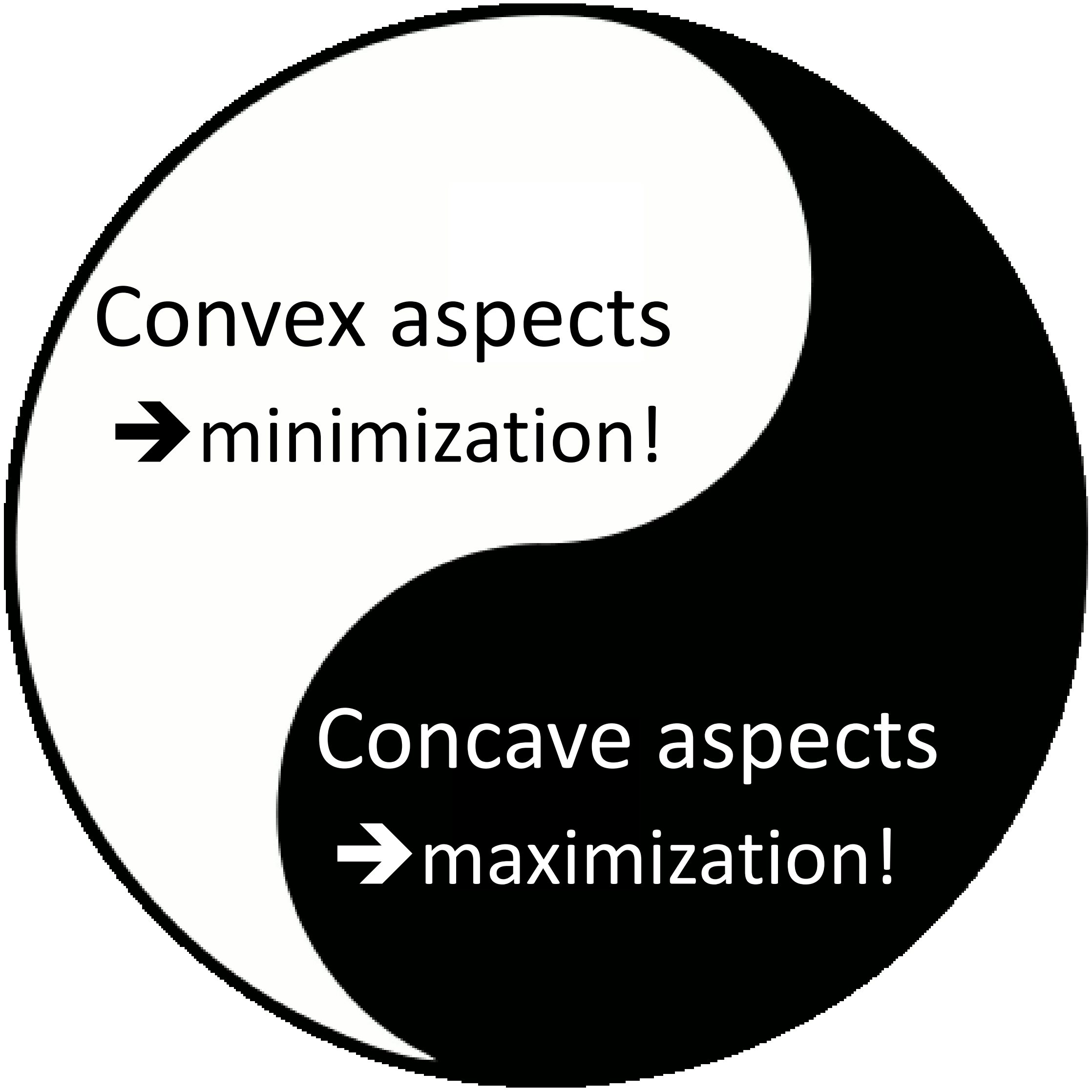
Well, maybe $F(A) = \min(F_1(A),F_2(A))$ instead?

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

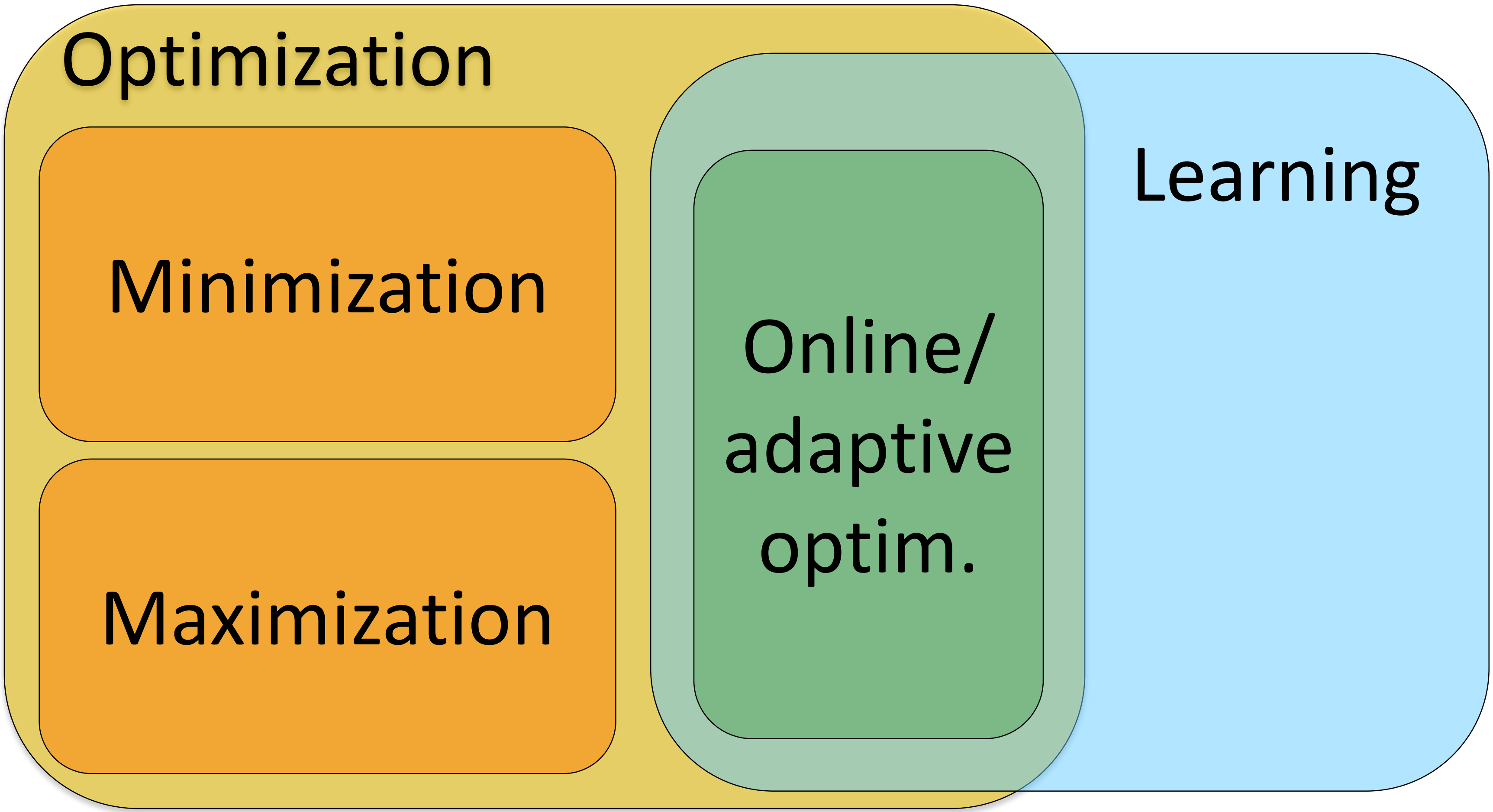
$$F(\{b\}) - F(\{\}) = 0$$
$$<$$
$$F(\{a,b\}) - F(\{a\}) = 1$$

$\min(F_1,F_2)$ not submodular in general!

Two faces of submodular functions



What to do with submodular functions



Here we focus on optimization & applications

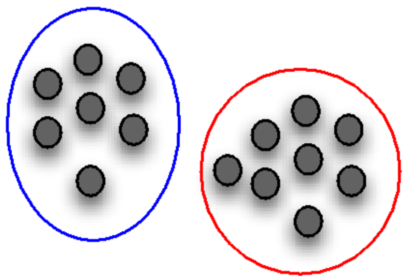
Optimization

Minimization

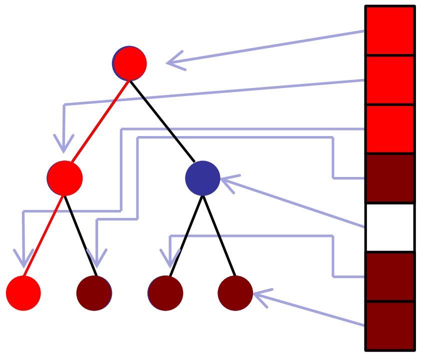
Maximization

Minimization and maximization not the same??

Submodular minimization

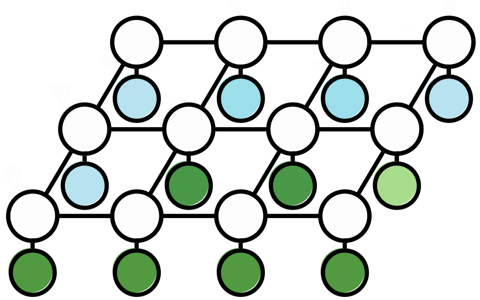


clustering

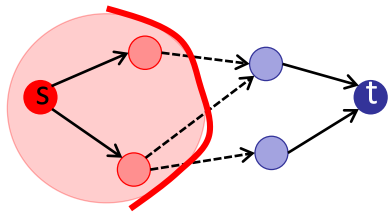


structured sparsity regularization

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



MAP inference



minimum cut

Submodular minimization

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

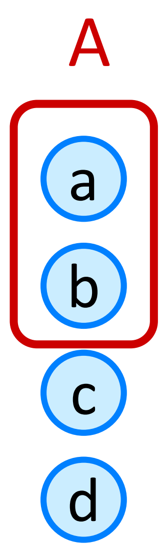
➔ submodularity and convexity

Set functions and energy functions

any set function

with $|V| = n$

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

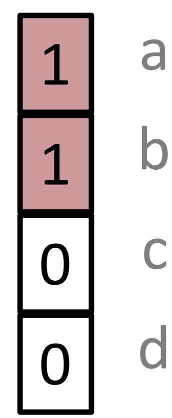


$$\hat{=}$$

... is a function on binary vectors!

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

$$x = e_A$$

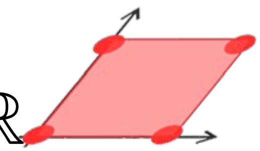


pseudo-boolean function

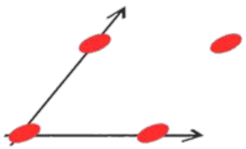
Submodularity and convexity

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!

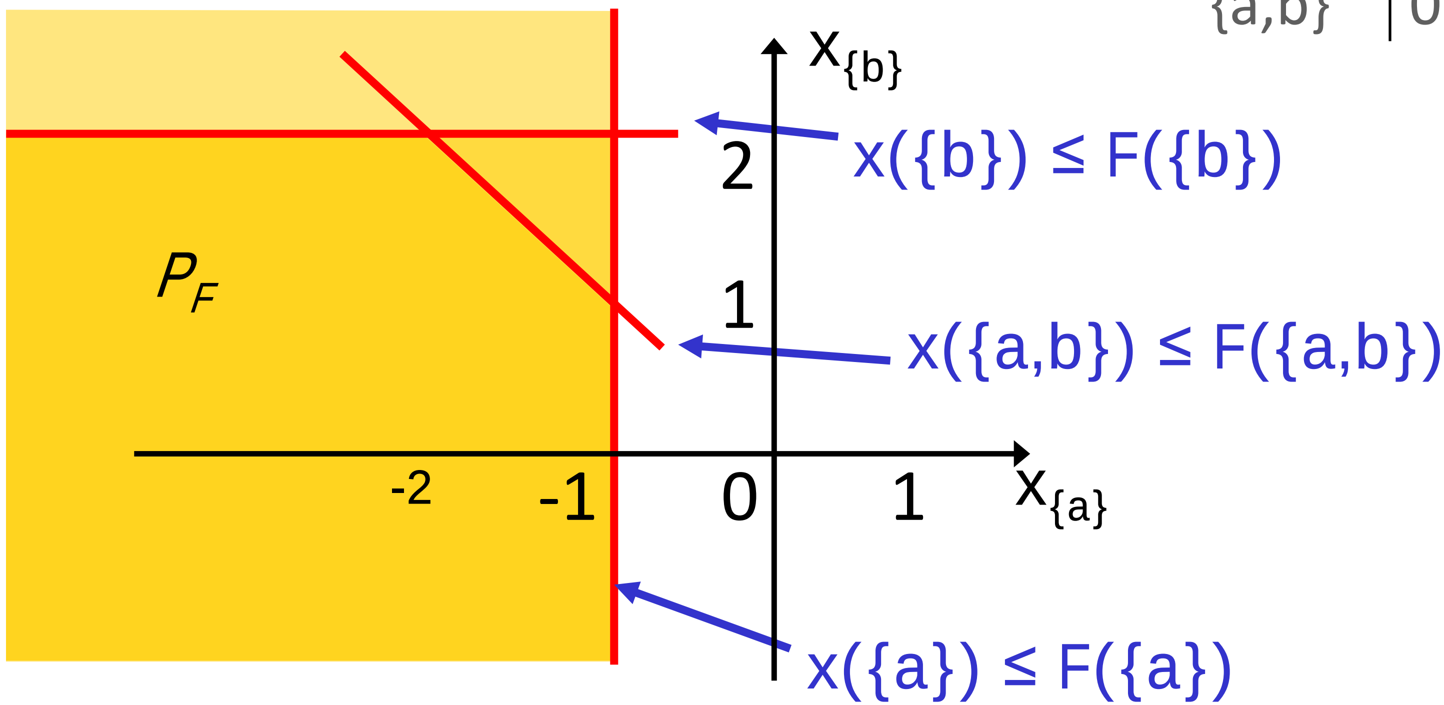
The submodular polyhedron P_F

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

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

Example: V = {a,b}

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



Evaluating the Lovász extension

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

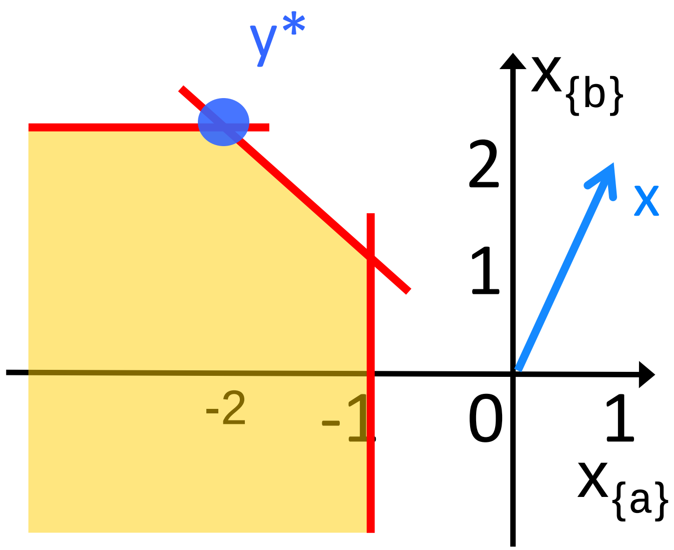
Linear maximization over P_F

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

Exponentially many constraints!!! ☹

Computable in O(n log n) time ☺

[Edmonds '70]

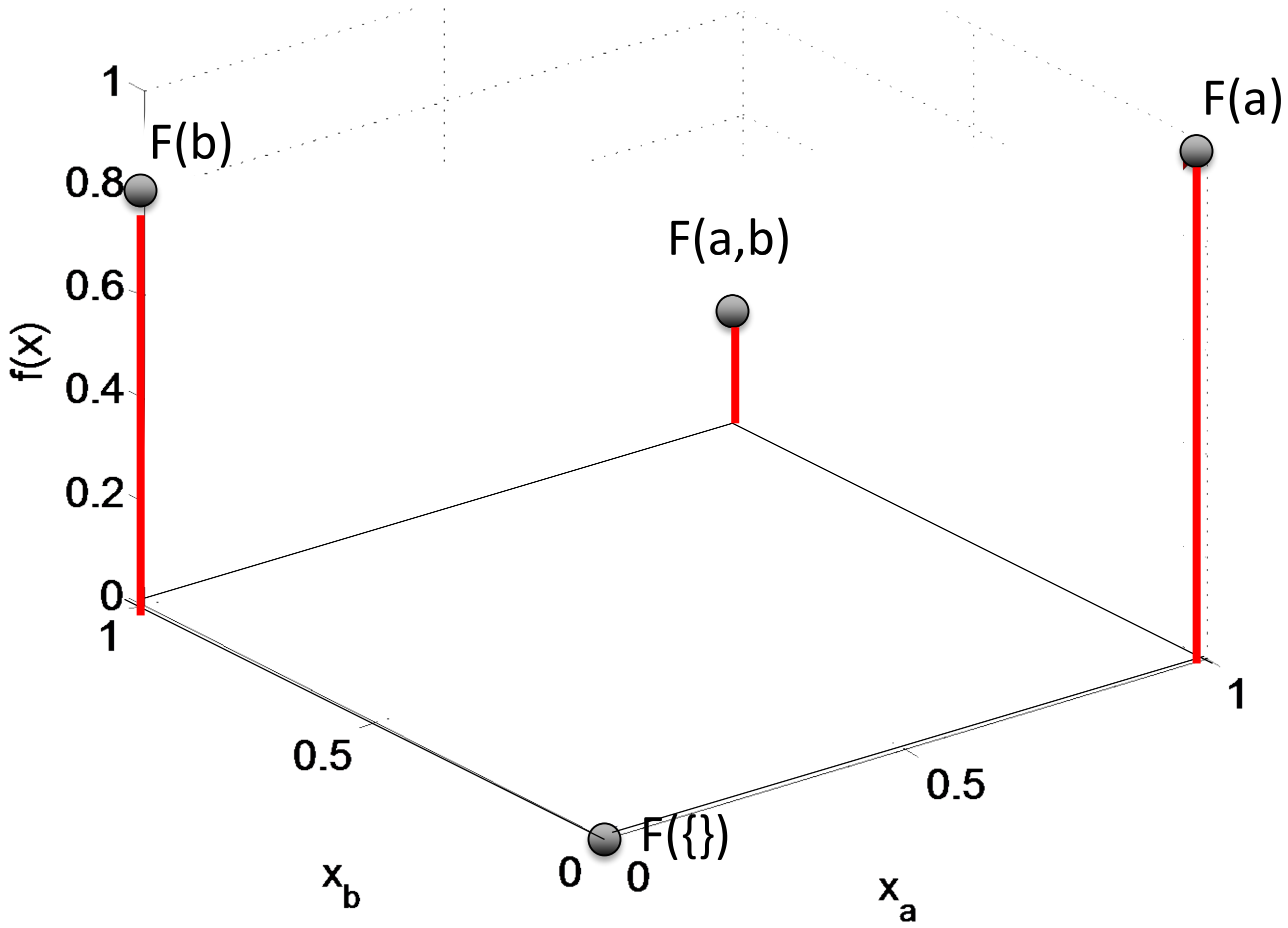


greedy algorithm:

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

- 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}

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

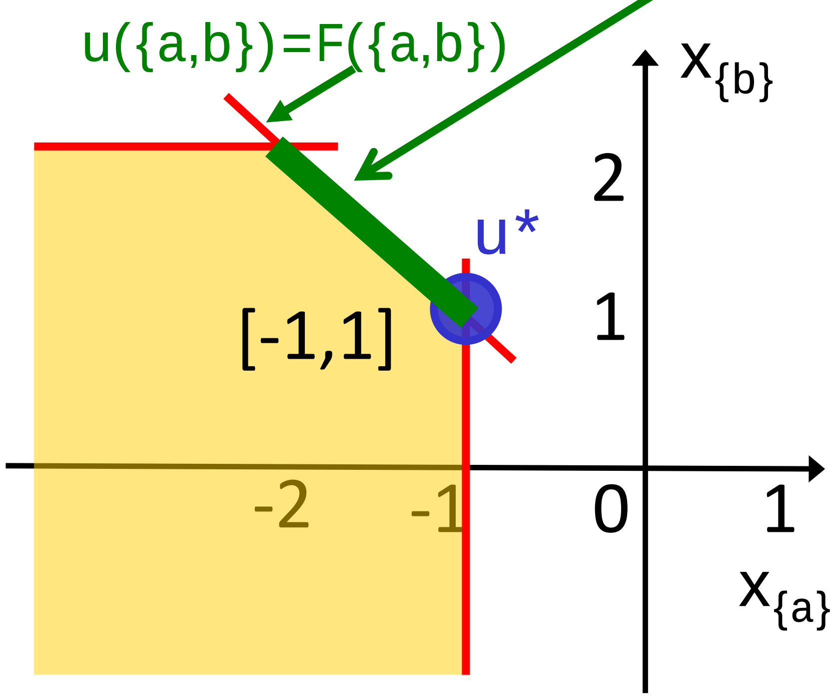
Regularized problem

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

dual: minimum norm problem

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

Base polytope B_F



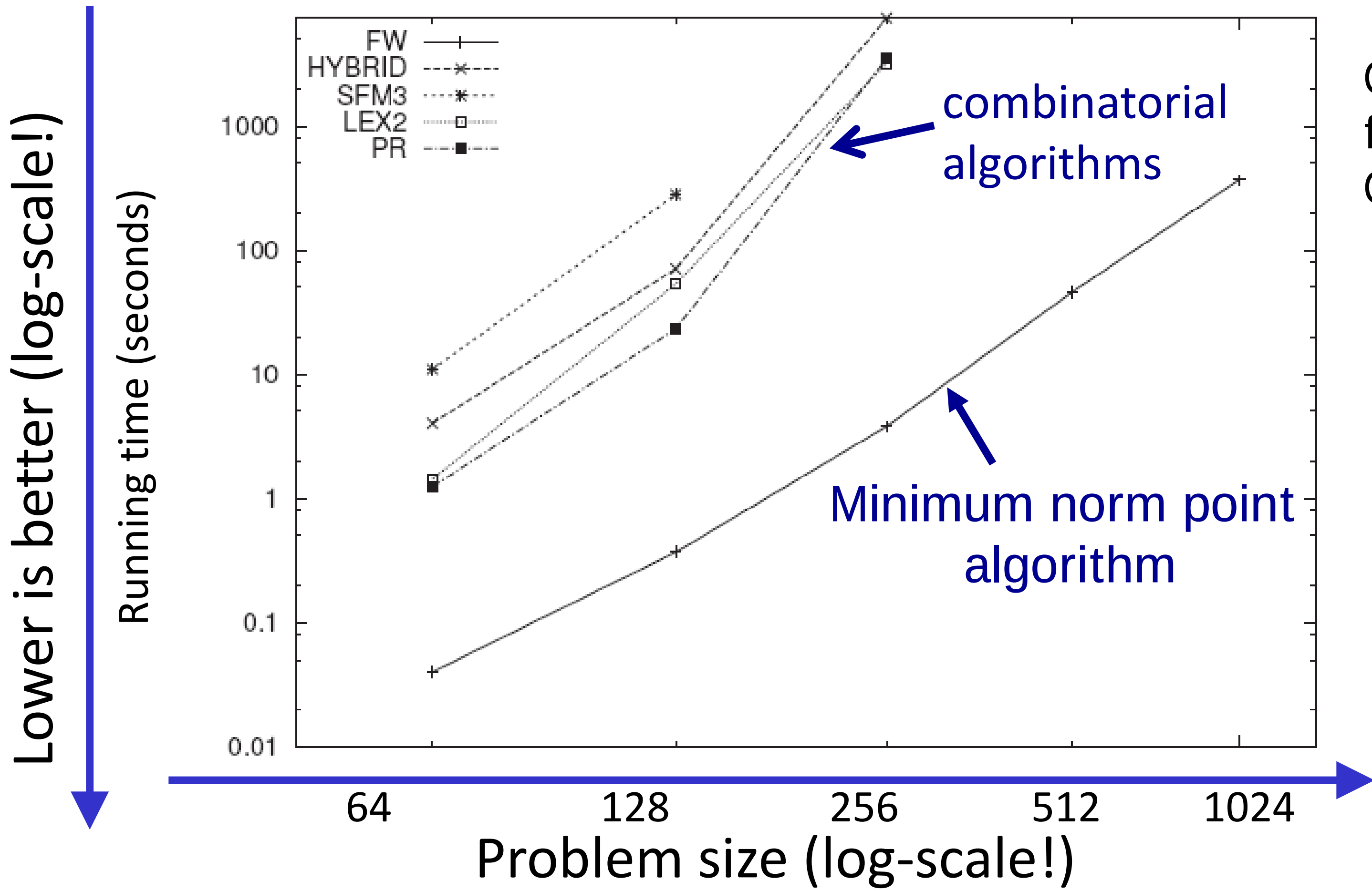
$$A^* = \{i \mid u^*(i) \leq 0\}$$

minimizes *F*:

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

Fujishige '91, Fujishige & Isotani '11

Empirical comparison

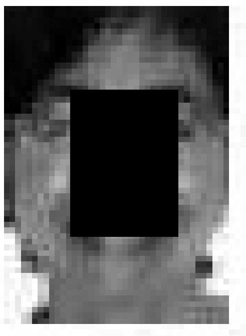


Cut functions from DIMACS Challenge

Minimum norm point algorithm: usually orders of magnitude faster

[Fujishige & Isotani '11]

Example I: Sparsity



$\approx .1$

$+ .4$

$+ .2$

$$y \approx Mx$$

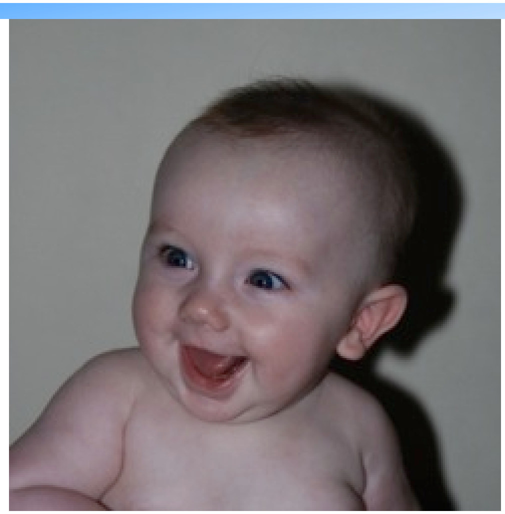
Want „representative“ dictionaries

Want „simple“ explanations (e.g., use few columns of M)

Example I: Sparsity

d

pixels

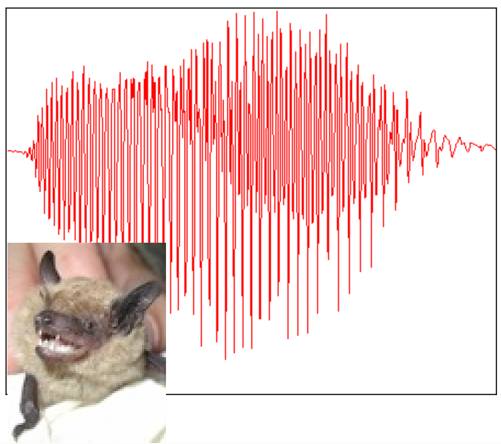


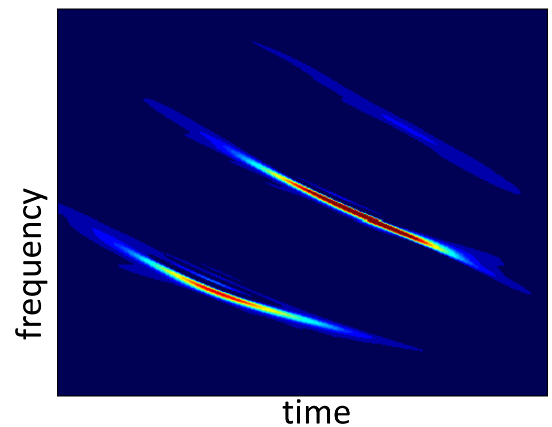
$k \ll d$

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

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

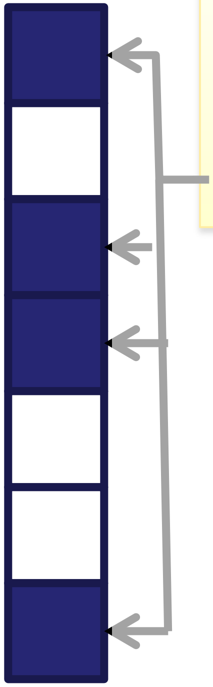
explain *y* with few columns of *M*: few x_i

discrete regularization on support *S* of *x*

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

relax to convex envelope

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



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

in nature: sparsity pattern often not random…

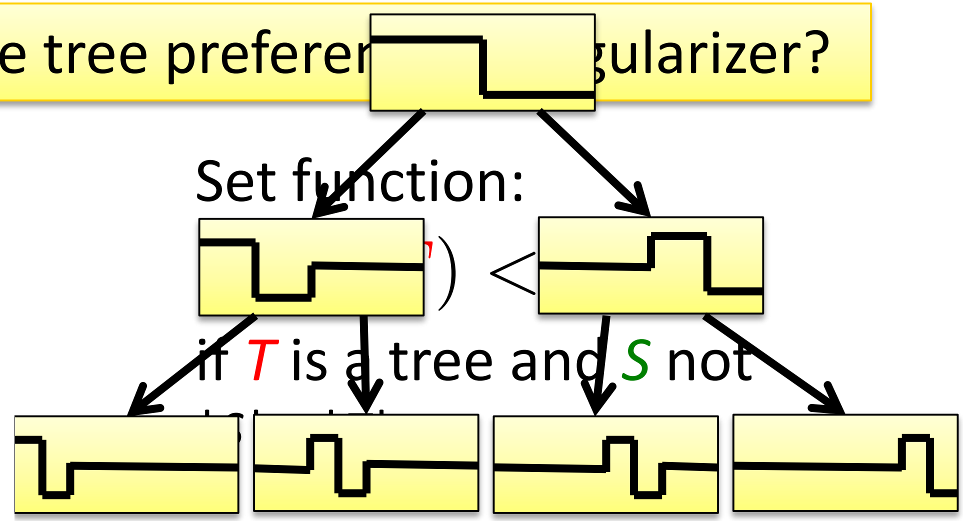
Structured sparsity

Incorporate tree preferen… …gularizer?

Set function:

) <

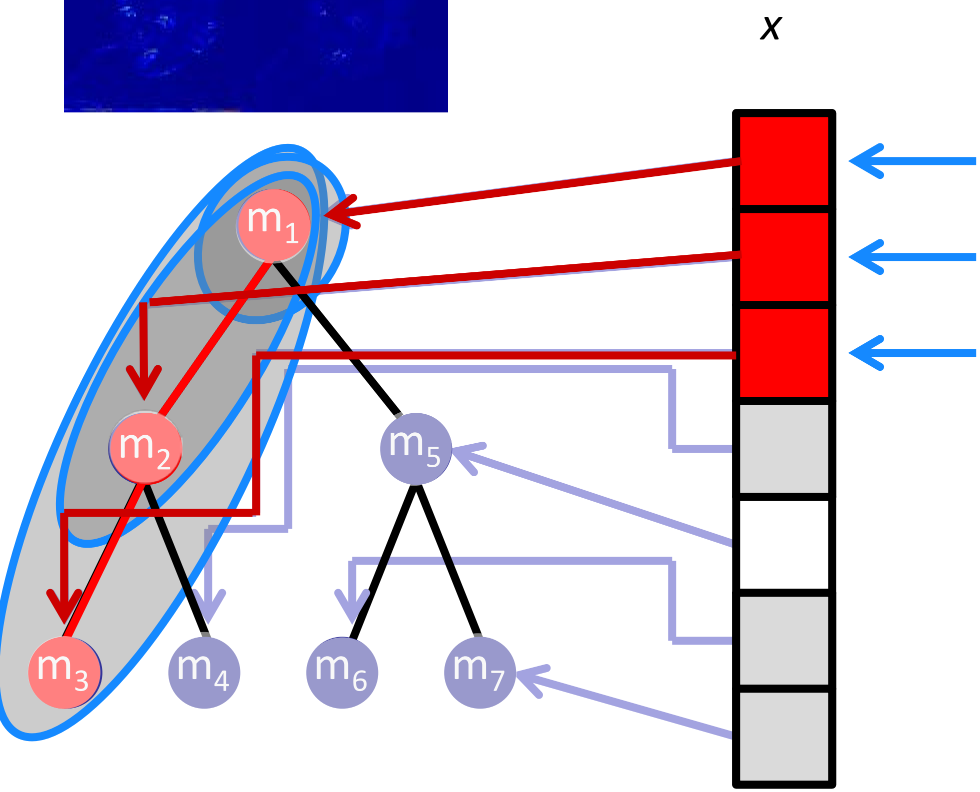
if T is a tree and S not



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

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

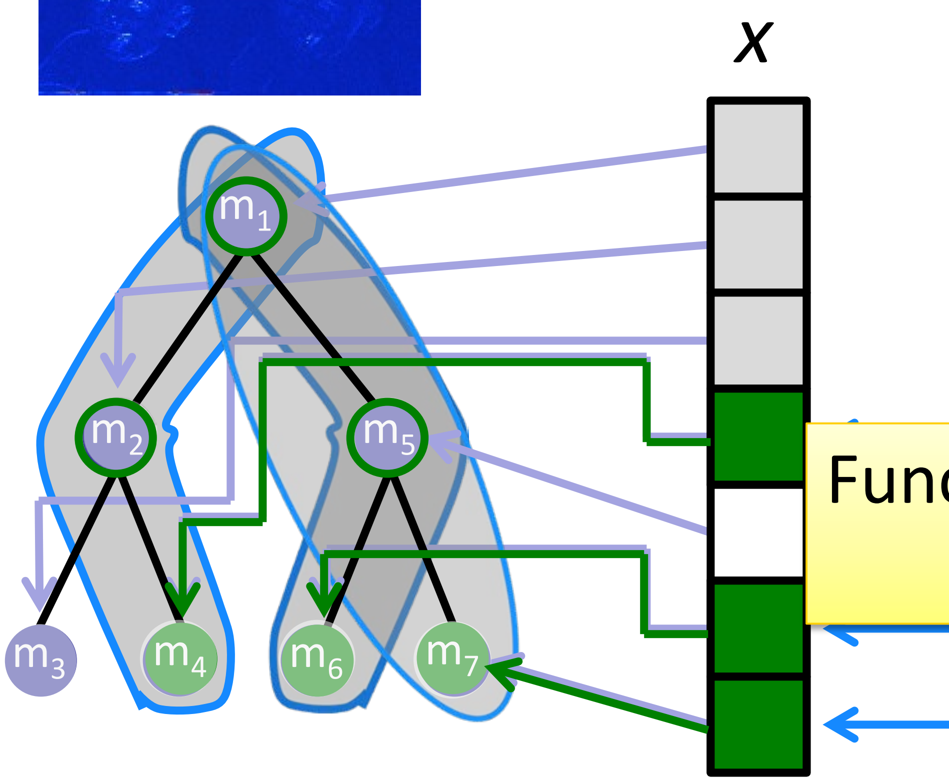
Incorporate tree preference in regularizer?

Set function:

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

If T is a tree and S not,

$|S| = |T|$



Function F is …

submodular! ☺

$$F(T) = 3$$

Sparsity-inducing norms through submodular functions [Bach NIPS 2010]

$$\min_{x} \; \|y - Mx\|^2 \quad + \lambda\Omega(x)$$

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

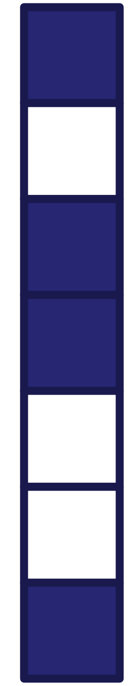
$$\Omega(x) = f(|x|)$$

Optimization: submodular minimization

Further connections: Dictionary Selection

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

Where does the dictionary M come from?

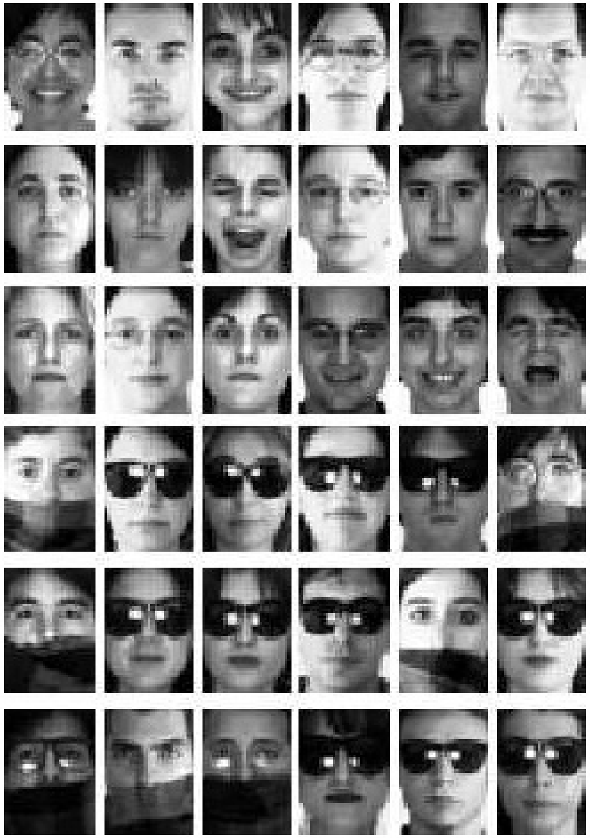


Want to learn it from data: $\{y_1, \ldots, y_n\} \subseteq \mathbb{R}^d$

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

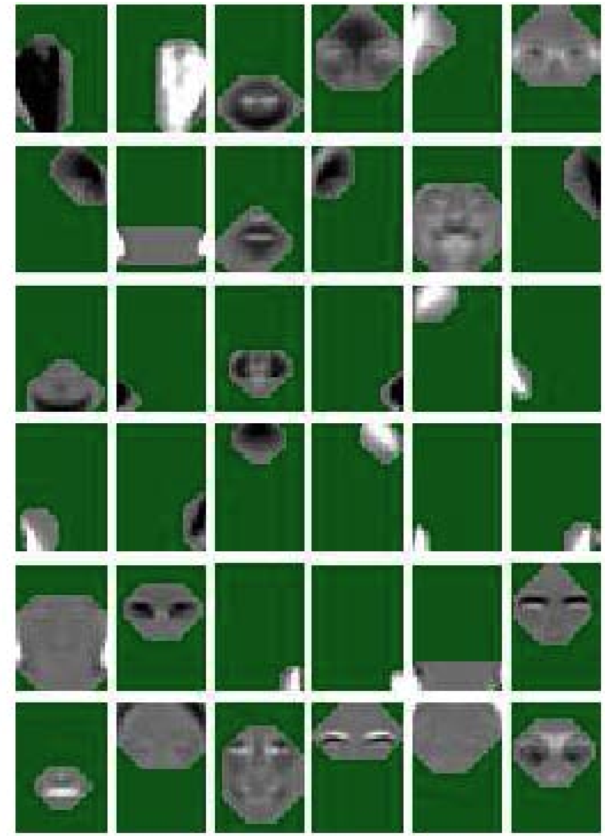
[Krause & Cevher '10; Das & Kempe '11]

Structured sparse dictionary learning [Bach et al, 2011]



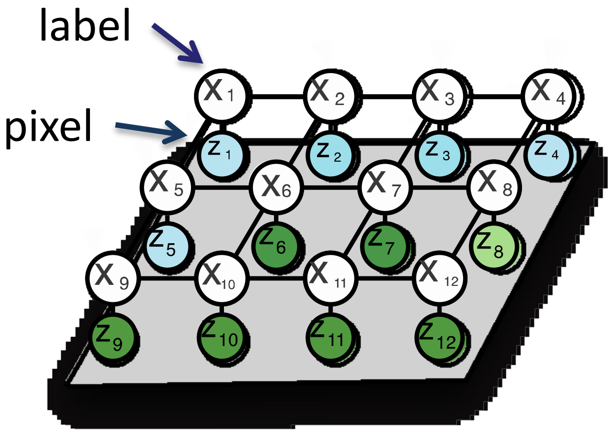
Original images

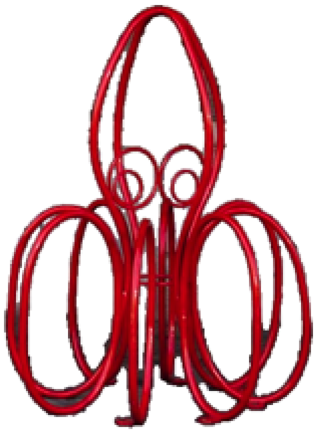
Dictionary from NMF



Structured-sparse Dictionary (SSPCA)

Example II: MAP inference





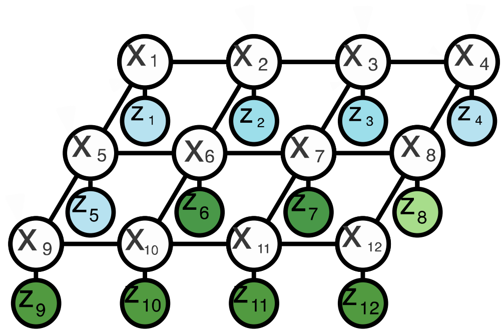
$$\max_{\mathbf{x}\in\{0,1\}^n} \; P(\mathbf{x} \mid \mathbf{z}) \propto \exp(-E(\mathbf{x};\mathbf{z}))$$

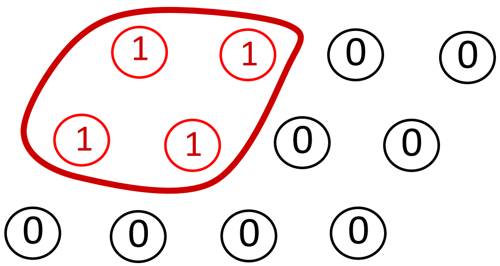
labels

pixel values

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

Example II: MAP inference



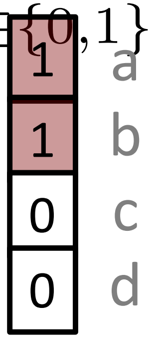


Recall: equivalence

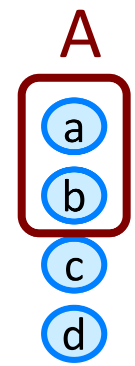
function on binary vectors | set function

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

$x = e_A$



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



if F is submodular, then

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

MAP inference = submodular minimization!

polynomial-time

Special cases

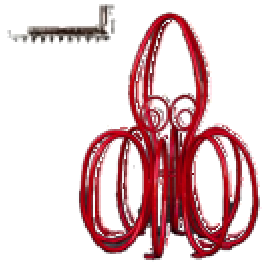
Minimizing general submodular functions:

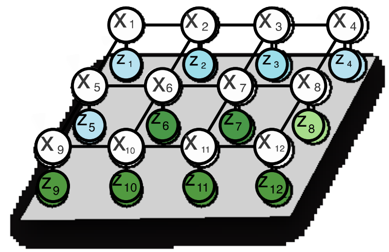
poly-time, but not very scalable

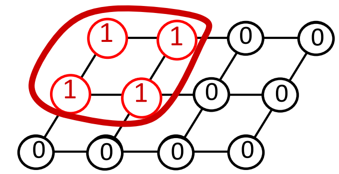
Special structure $\rightarrow$ faster algorithms

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

MAP inference







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

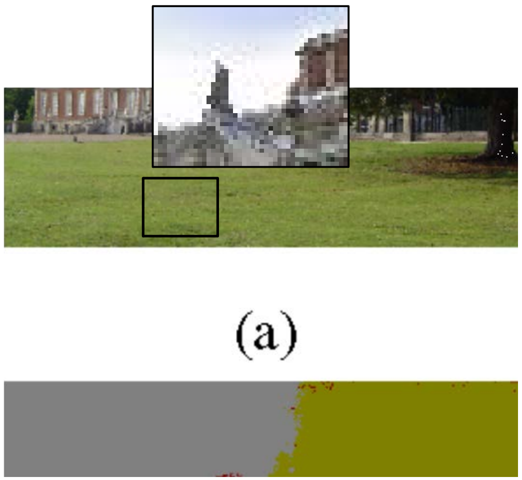
if each E_{ij} is submodular (“attractive”):

$$\underset{a}{E_{ij}(1,0)} + \underset{b}{E_{ij}(0,1)} \;\geq\; E_{ij}(0,0) + \underset{ab}{E_{ij}(1,1)}$$

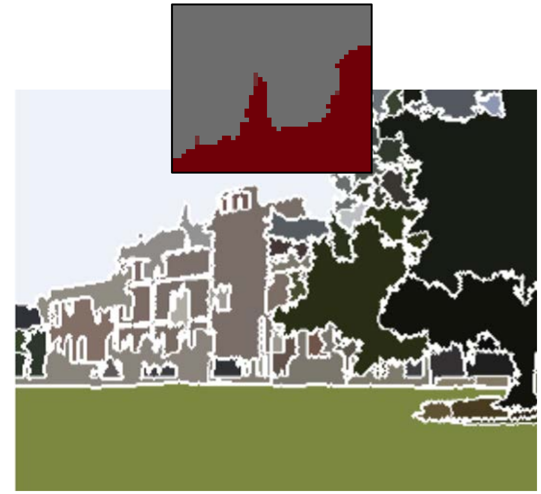
then F is a graph cut function.

MAP inference = Minimum cut: fast ☺

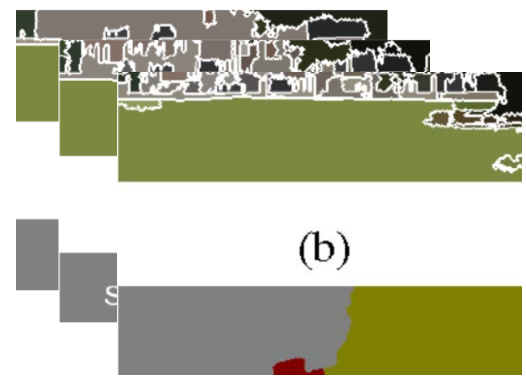
Pairwise is not enough…



color + pairwise



color + pairwise +



Pixels in one tile should have the same label

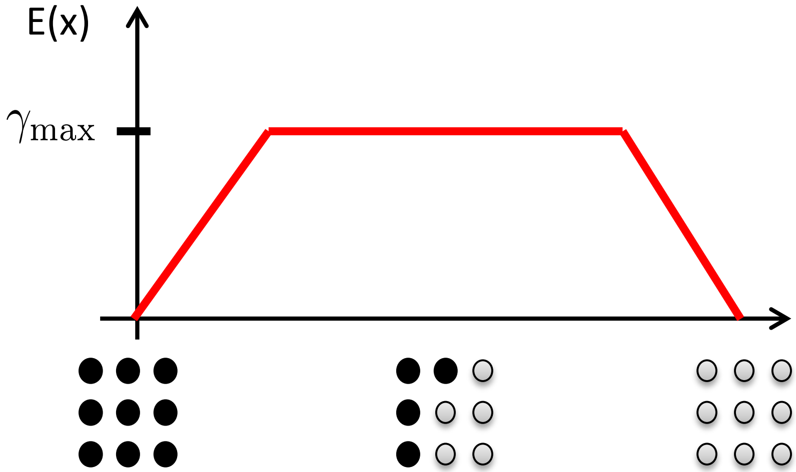
$$E(x) = \sum_i E_i(x_i) + \sum_{ij} E_{ij}(x_i, x_j)$$

[Kohli et al.`09]

Enforcing label consistency

Pixels in a superpixel should have the same label

(b)



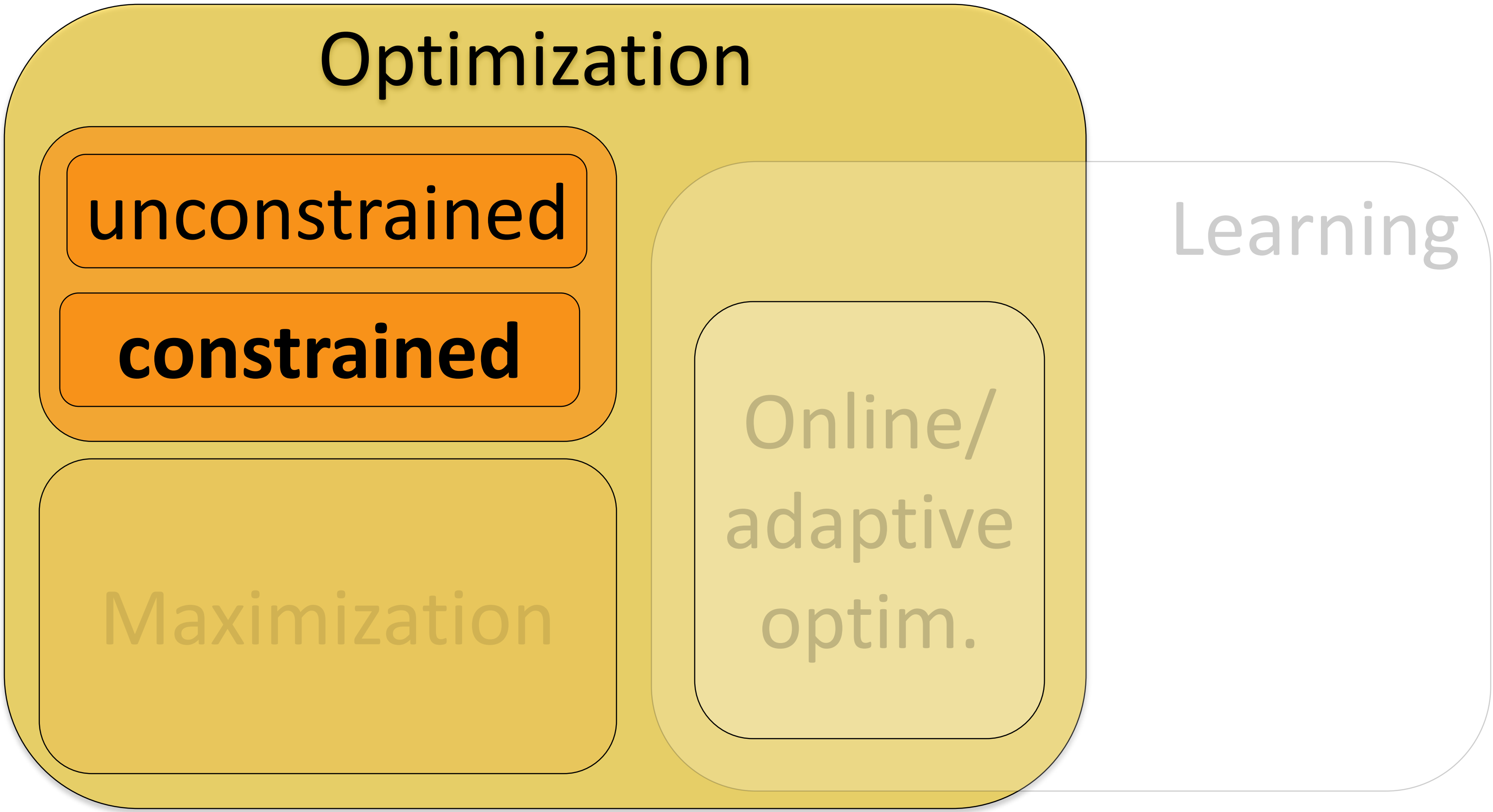
concave function of cardinality ➔ submodular ☺

Can still be transformed into a graph cut instance!

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: odd/even cardinality, inclusion/exclusion of a set $\ldots$

General case: NP hard

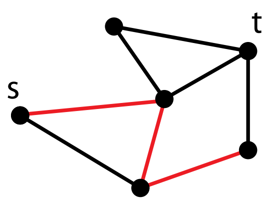
- hard to approximate within polynomial factors!
- But: special cases often still work well

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

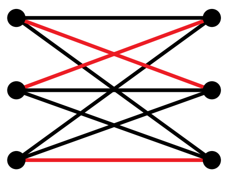
Constraints

minimum…

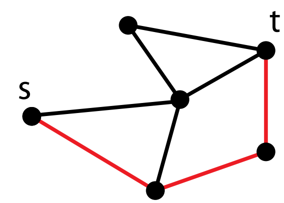
cut



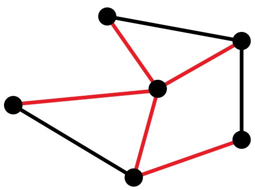
matching



path



spanning tree



ground set: edges in a graph

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

Constrained modular minimization

Constrained submodular minimization

Submodular (“cooperative”) cut
[Jegelka & Bilmes CVPR ‘11]

Graph cut

Efficient constrained optimization

Idea: minimize a series of modular surrogate functions

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

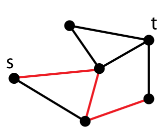
$$\hat{F}^i(S) = \sum_{e \in S} w^i(S)$$

2. Solve easy sum-of-weights (modular) problem:

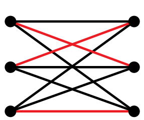
$$S^i = \arg\min_{S \in \mathcal{C}} \hat{F}^i(S)$$

and repeat.

cut

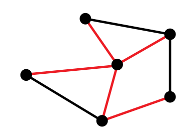


matching

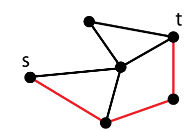


- efficient
- only need to solve sum-of-weights problems
- Provides certain approximation guarantees

spanning tree



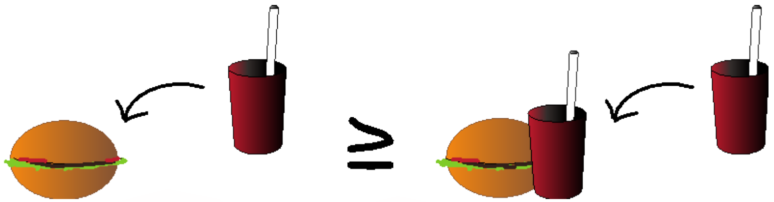
path

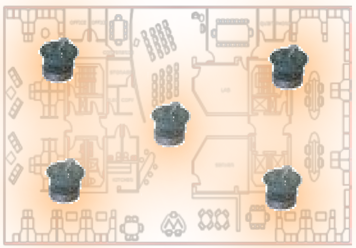


[Jegelka & Bilmes `11, Iyer et al. ICML `13, see also Krause et al ‘06]

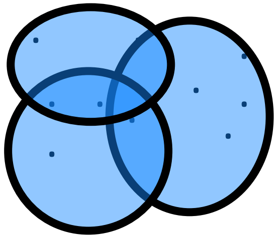
Outline

- What is submodularity?
- Optimization
  - Minimize costs
  - **Maximize utility**
- Applications
- Outlook and pointers

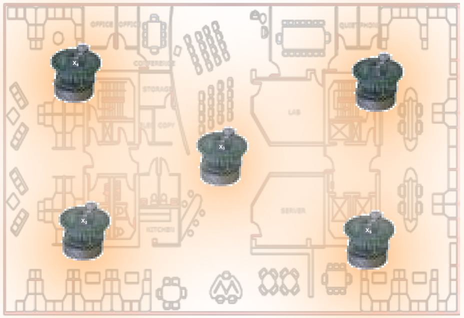




Submodular maximization

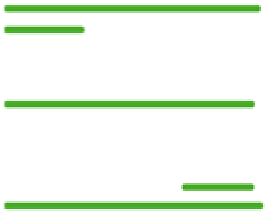


covering

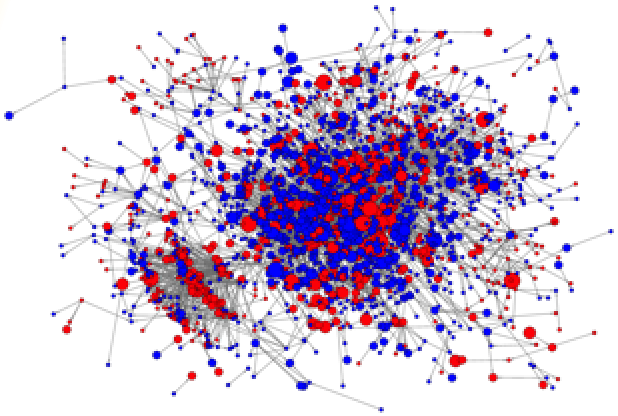


sensing

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

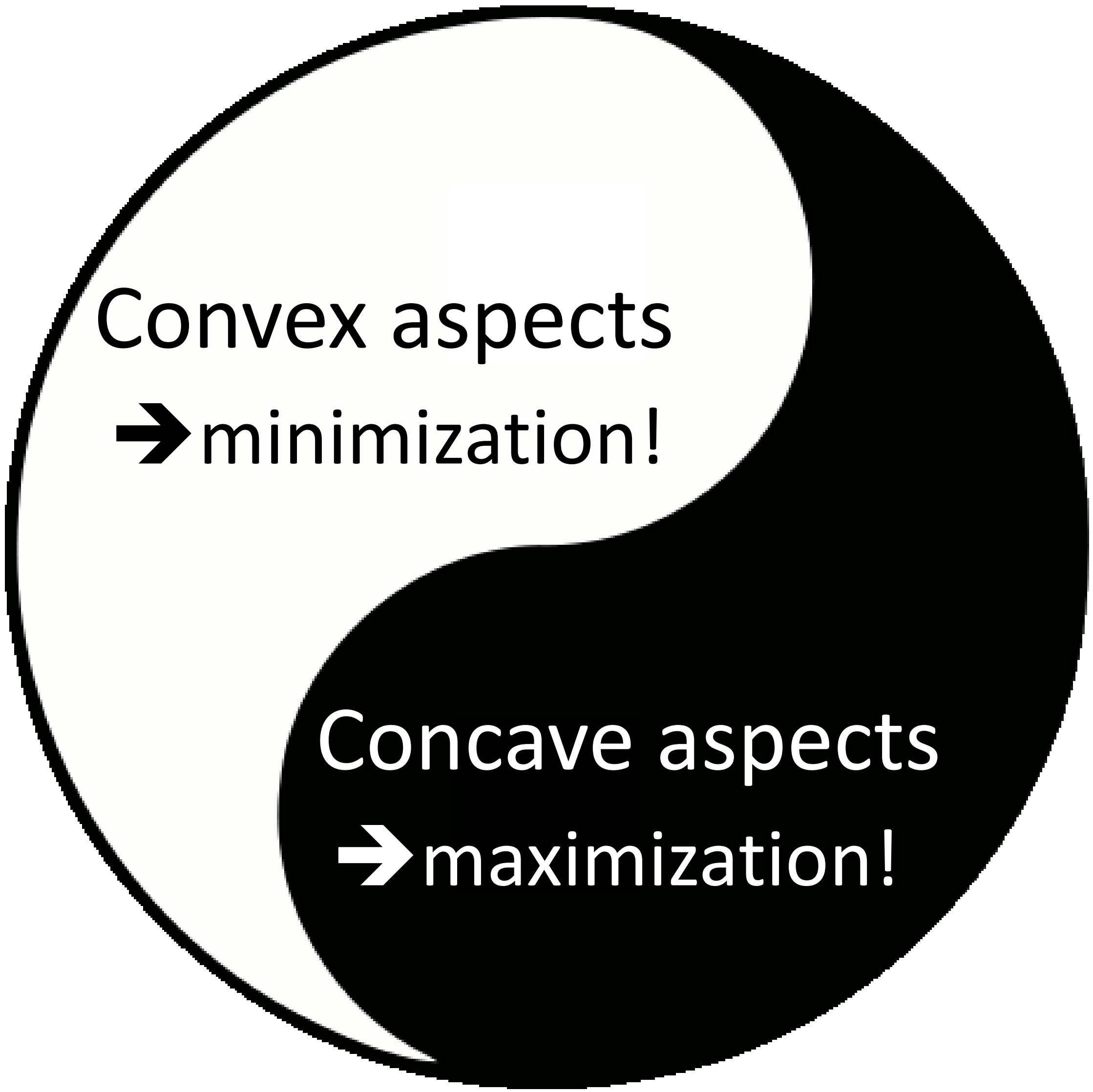


summarization



network inference

Two faces of submodular functions



Submodular maximization

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

➔ submodularity and concavity

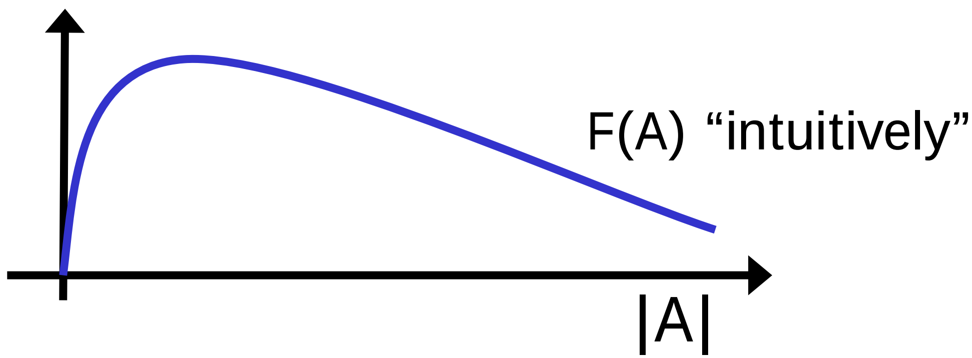
Concave aspects

- submodularity:

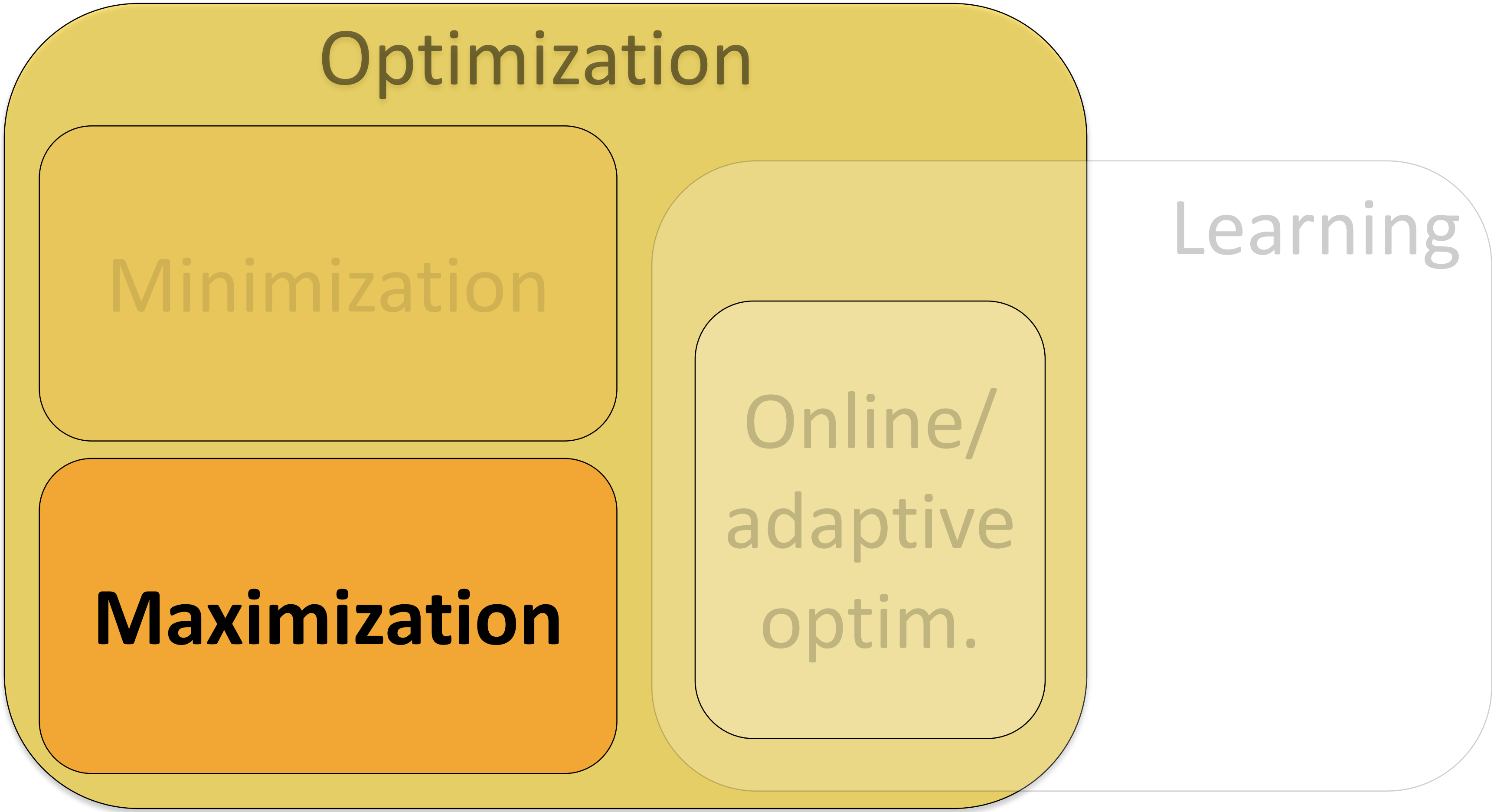
$$A \subseteq B, \;\; s \notin B :$$
$$F(A \cup s) - F(A) \quad \geq \quad F(B \cup s) - F(B)$$

- concavity:

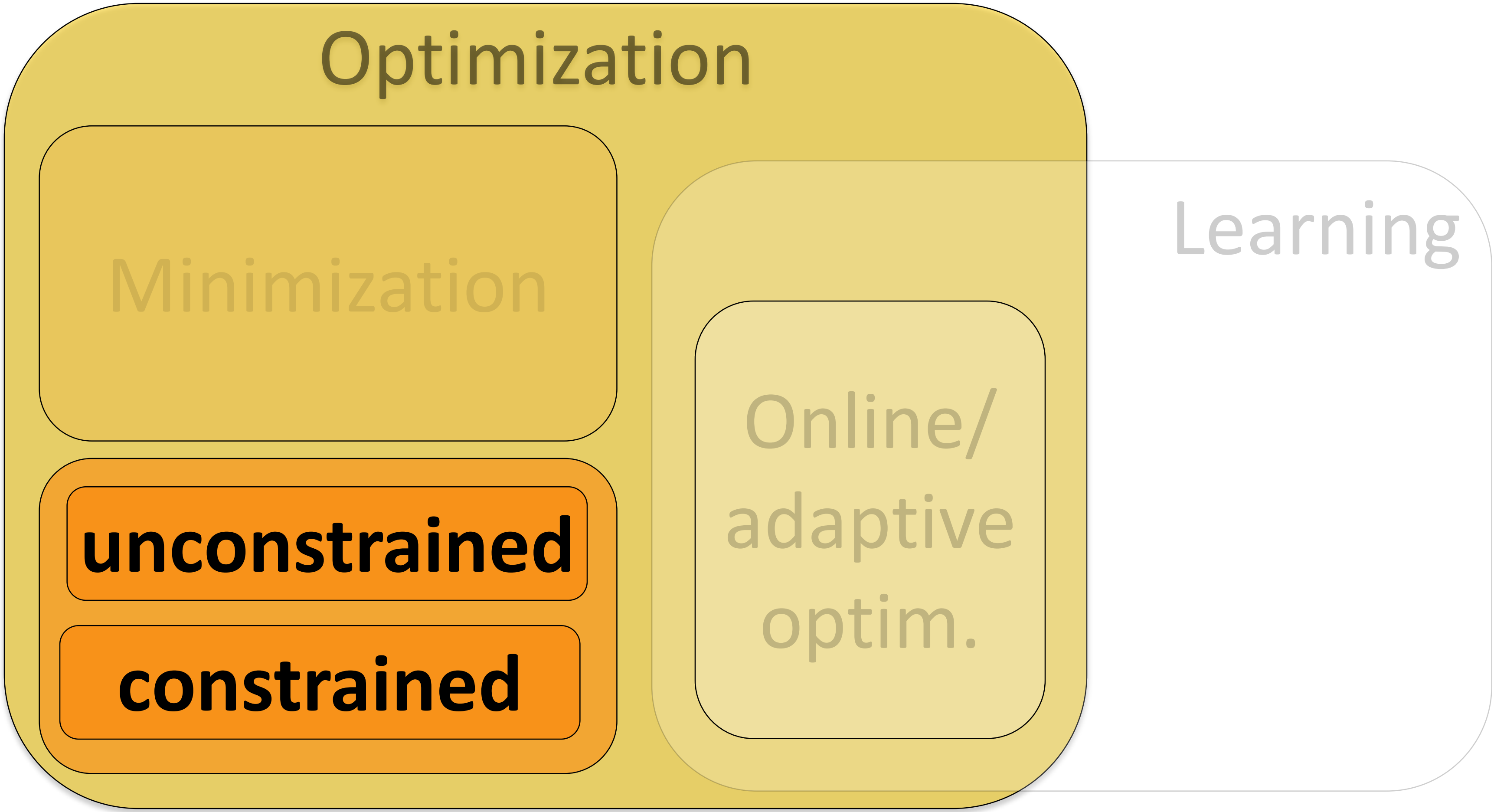
$$a \leq b, \;\; s > 0 :$$
$$f(a+s) - f(a) \quad \geq \quad f(b+s) - f(b)$$



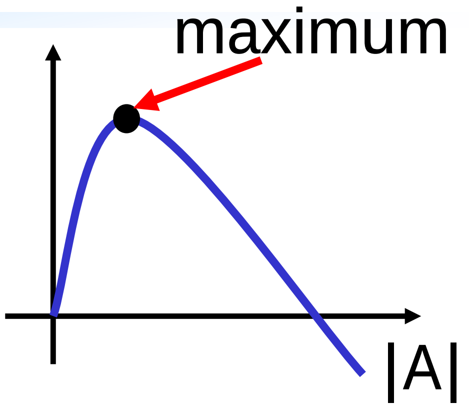
Optimization



Optimization



Maximizing submodular functions



- Suppose we want for submodular F

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

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

- In general: NP hard. Moreover:
- If F(A) can take negative values:
  As hard to approximate as maximum independent set (i.e., NP hard to get $O(n^{1-\varepsilon})$ approximation)

Exact maximization of SFs

- Mixed integer programming
  - Series of mixed integer programs [Nemhauser et al ‘81]
  - Constraint generation [Kawahara et al ‘09]
- Branch-and-bound
  - „Data-Correcting Algorithm“ [Goldengorin et al ’99]

Useful for small/moderate problems

All algorithms worst-case exponential!

Maximizing positive submodular functions

[Feige, Mirrokni, Vondrak '09; Buchbinder, Feldman, Naor, Schwartz '12]

Theorem

Given a nonnegative submodular function F,
RandomizedUSM returns set A_R such that

$$F(A_R) \geq 1/2 \ \max_A F(A)$$

- Cannot do better in general than ½ unless P = NP

Unconstrained vs. constraint maximization

Given monotone utility F(A) and cost C(A), optimize:

Option 1:

$$\boxed{\begin{aligned} &\max_A F(A) - C(A) \\ &\text{s.t.}\quad A \subseteq V \end{aligned}}$$

"Scalarization"

Option 2:

$$\boxed{\begin{aligned} &\max_A F(A) \\ &\text{s.t.}\quad C(A) \leq B \end{aligned}}$$

"Constrained maximization"

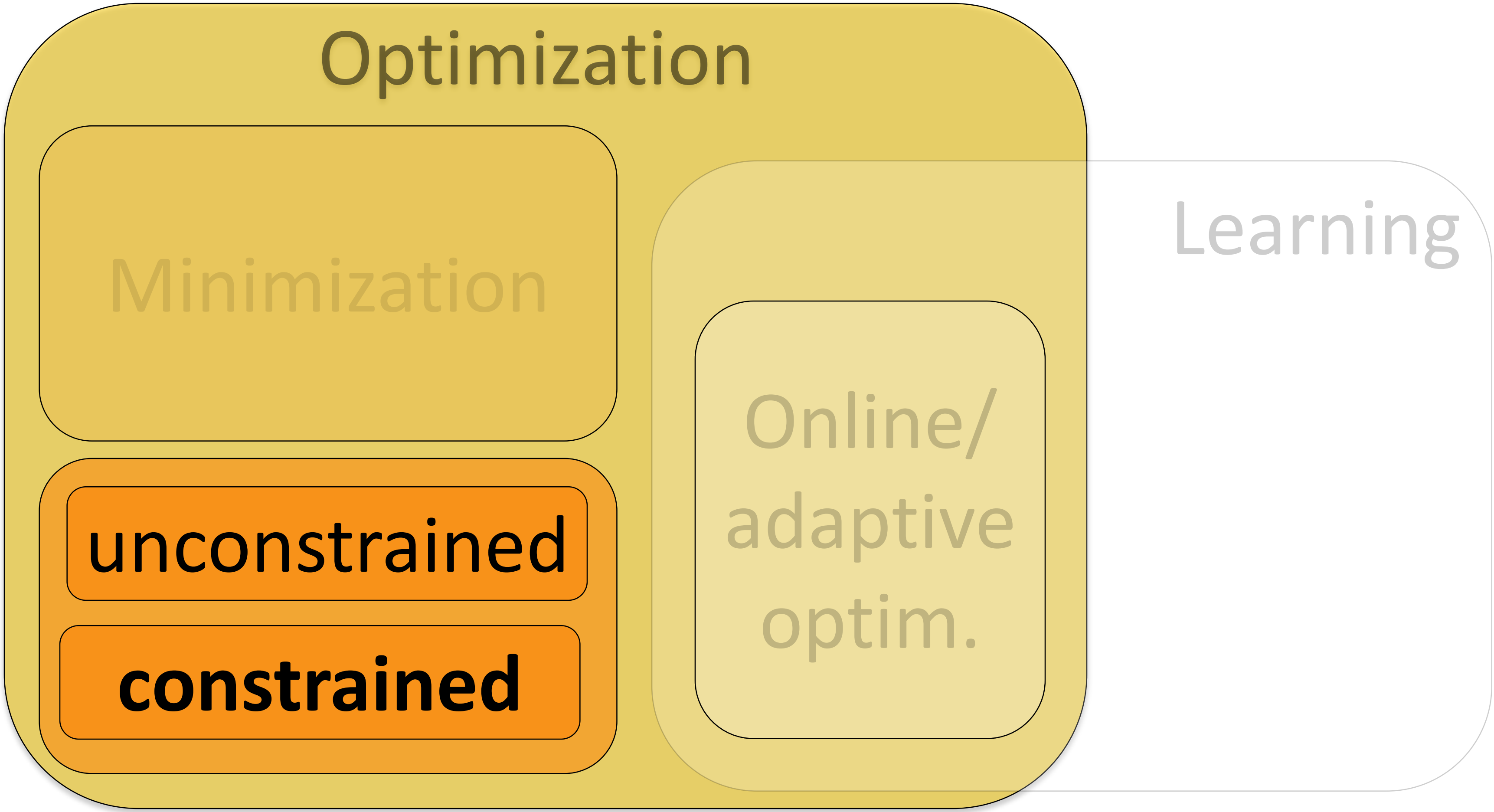
Can get 1/2 approx…

if F(A)-C(A) ≥ 0 for all sets A

Positiveness is a strong requirement ☹

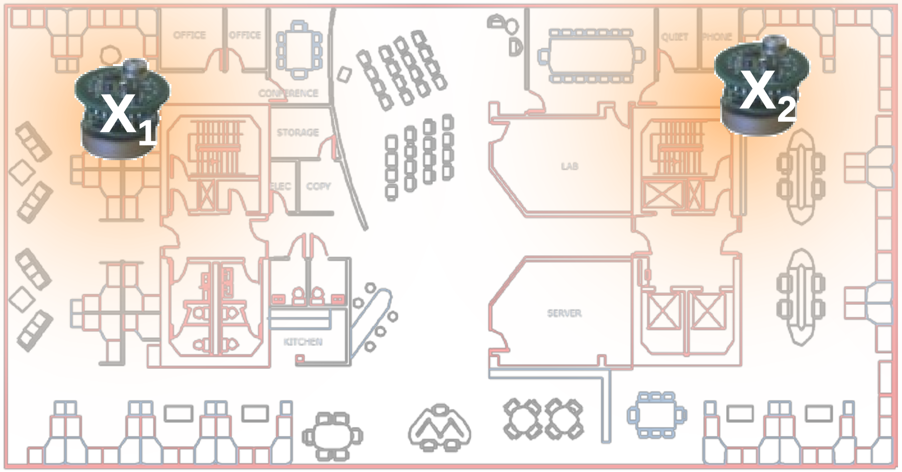
What is possible?

Optimization

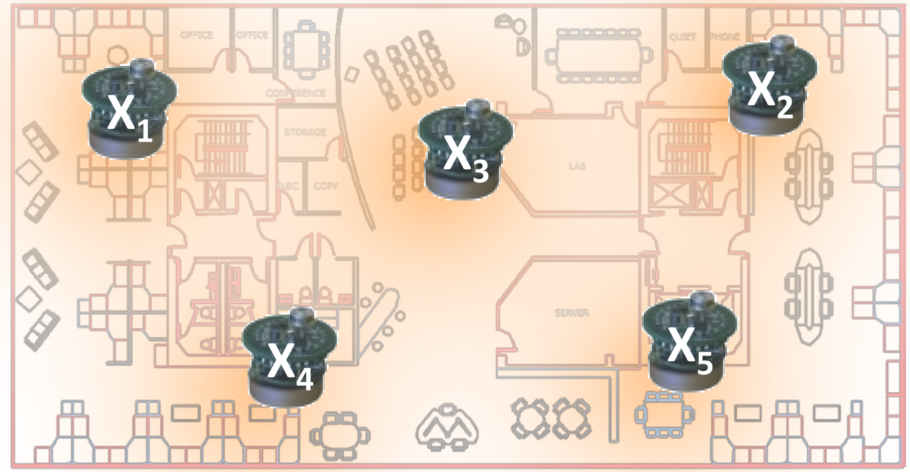


Monotonicity

Placement A = {1,2}



Placement B = {1,…,5}



F is monotonic: $$\forall A, s : \underbrace{F(A \cup \{s\}) - F(A)}_{\Delta(s \mid A)} \geq 0$$

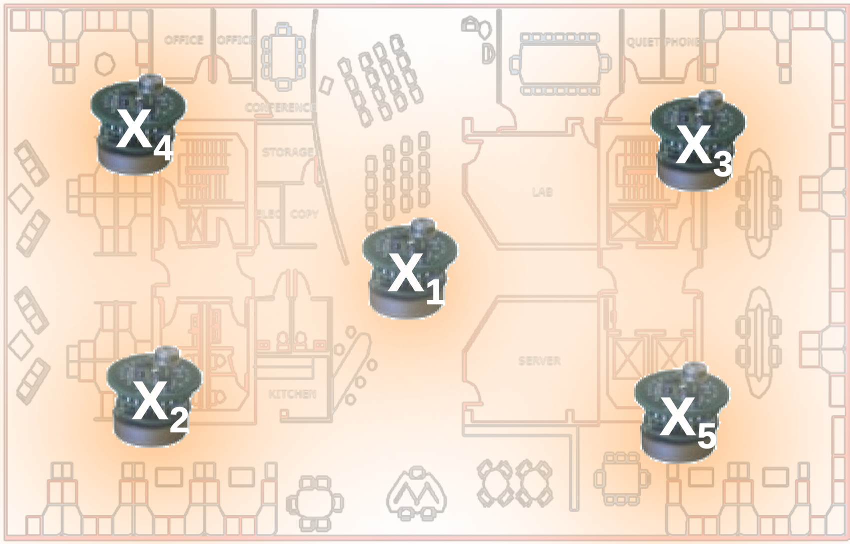
$$\Delta(s \mid A) \geq 0$$

Adding sensors can only help

Cardinality constrained maximization

- **Given**: finite set V, monotone submodular F
- **Want**: $\mathcal{A}^* \subseteq \mathcal{V}$ such that
$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}|\leq k} F(\mathcal{A})$$

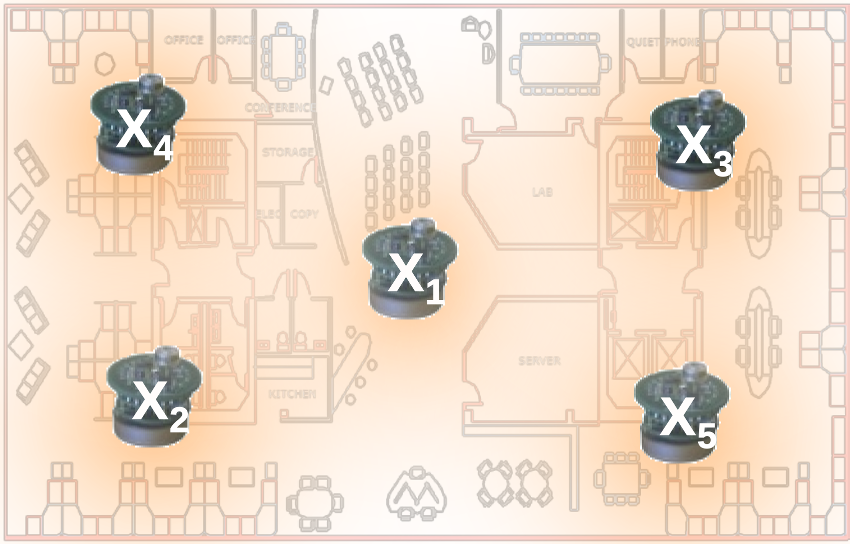
NP-hard!



Greedy algorithm

- **Given**: finite set V, monotone submodular F
- **Want**: $\mathcal{A}^* \subseteq \mathcal{V}$ such that $\mathcal{A}^* = \operatorname{argmax}_{|\mathcal{A}| \leq k} F(\mathcal{A})$

NP-hard!



Greedy algorithm:

Start with $\mathcal{A} = \emptyset$

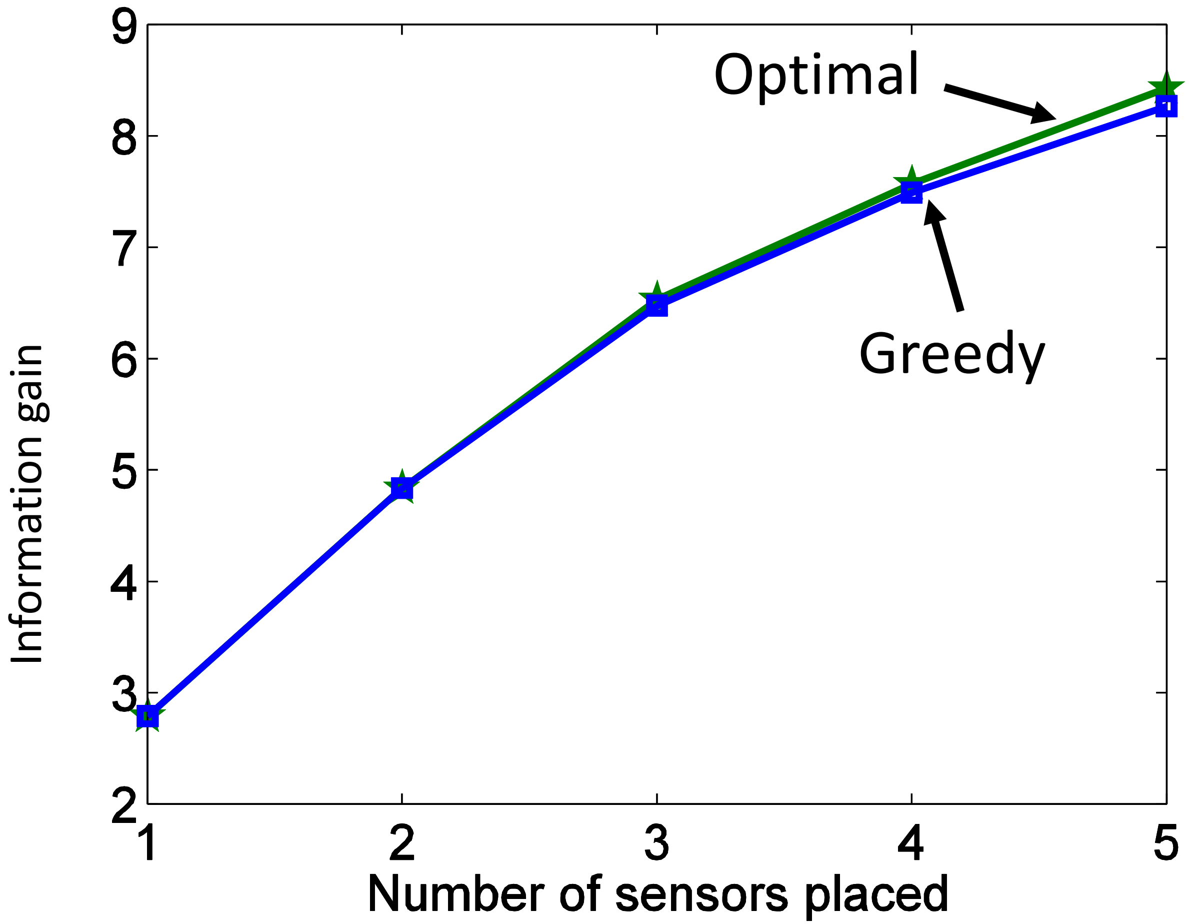
For i = 1 to k

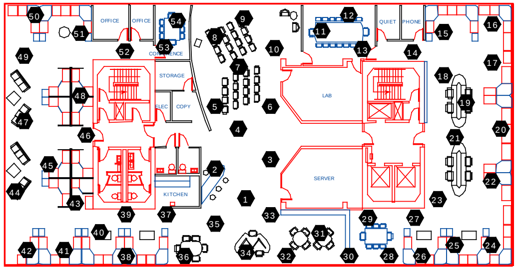
$s^* \leftarrow \arg\max_s F(\mathcal{A} \cup \{s\})$

$\mathcal{A} \leftarrow \mathcal{A} \cup \{s^*\}$

How well can this simple heuristic do?

Performance of greedy





Temperature data from sensor network

Greedy empirically close to optimal. Why?

One reason submodularity is useful

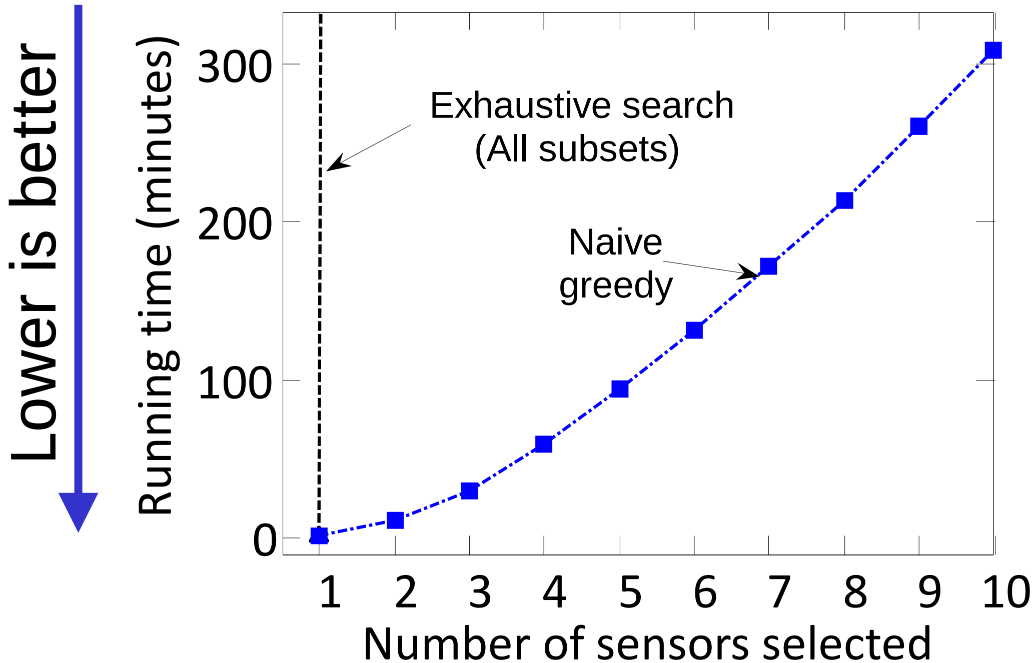
Theorem [Nemhauser, Fisher & Wolsey '78]

For monotonic submodular functions,
Greedy algorithm gives constant factor approximation

$$F(A_{greedy}) \geq \underbrace{(1-1/e)}_{\sim 63\%} F(A_{opt})$$

- Greedy algorithm gives near-optimal solution!
- In general, need to evaluate exponentially many sets to do better!
  [Nemhauser & Wolsey '78]
- Also many special cases are hard (set cover, mutual information, …)

Even greedy can be slow…



Sensor placement

Placing 10 sensors takes 5 hours on highly optimized implementation

Scaling up the greedy algorithm [Minoux ’78]

In round i+1,

- have picked $A_i = \{s_1,\ldots,s_i\}$
- pick $s_{i+1} = \text{argmax}_s\, F(A_i \cup \{s\})\text{-}F(A_i)$

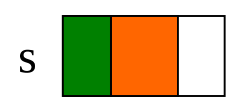
I.e., maximize “marginal benefit” $\otimes(s \mid A_i)$

$$\otimes(s \mid A_i) = F(A_i \cup \{s\})\text{-}F(A_i)$$

Key observation: Submodularity implies

$$i \leq j \;\; \Rightarrow \;\; \otimes(s \mid A_i) \;\geq\; \otimes(s \mid A_j)$$

$\otimes(s \mid A_i) \geq \otimes(s \mid A_{i+1})$

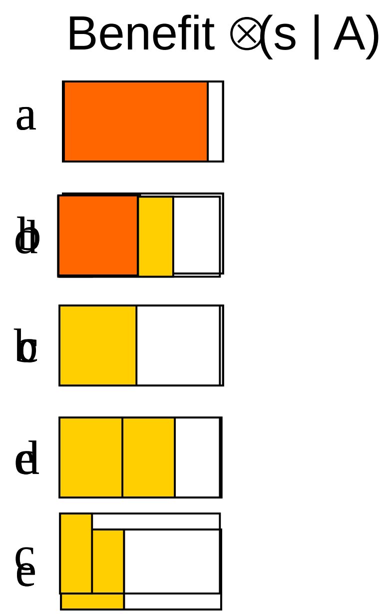


Marginal benefits can never increase!

"Lazy" greedy algorithm [Minoux '78]

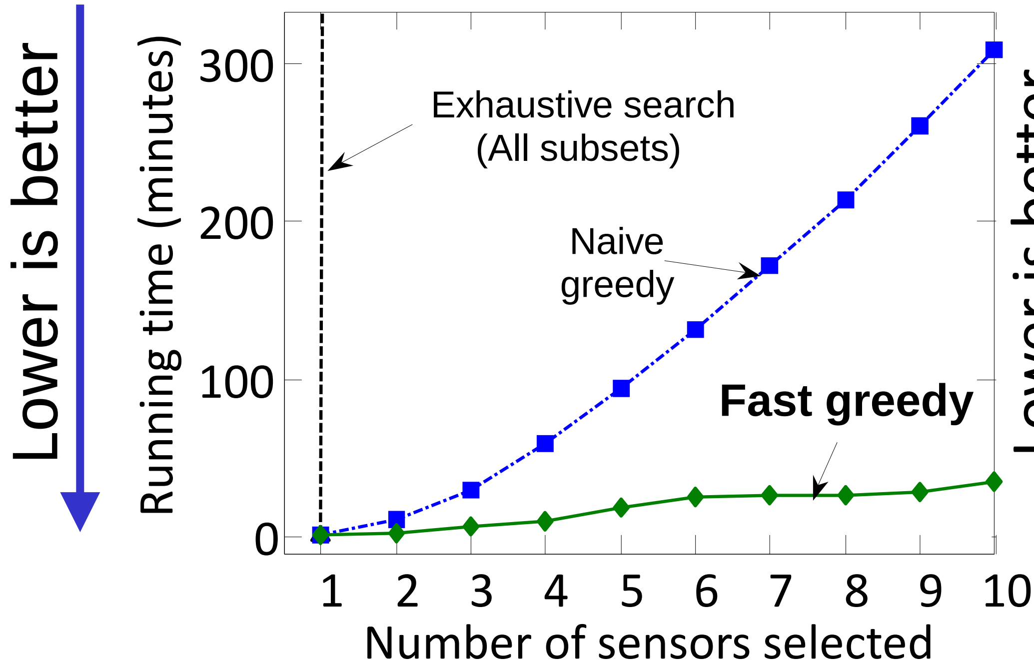
Lazy greedy algorithm:

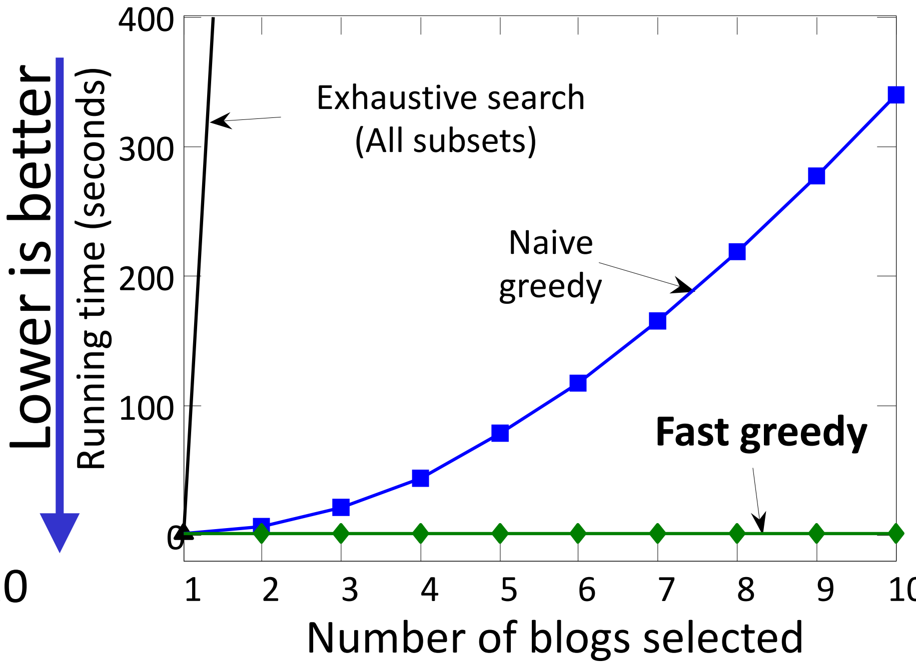
- First iteration as usual
- Keep an ordered list of marginal benefits $\otimes_i$ from previous iteration
- Re-evaluate $\otimes_i$ only for top element
- If $\otimes_i$ stays on top, use it, otherwise re-sort



Note: Very easy to compute online bounds, lazy evaluations, etc. [Leskovec, Krause et al. '07]

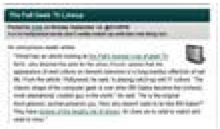
Empirical improvements [Leskovec, Krause et al'06]





Sensor placement

Blog selection

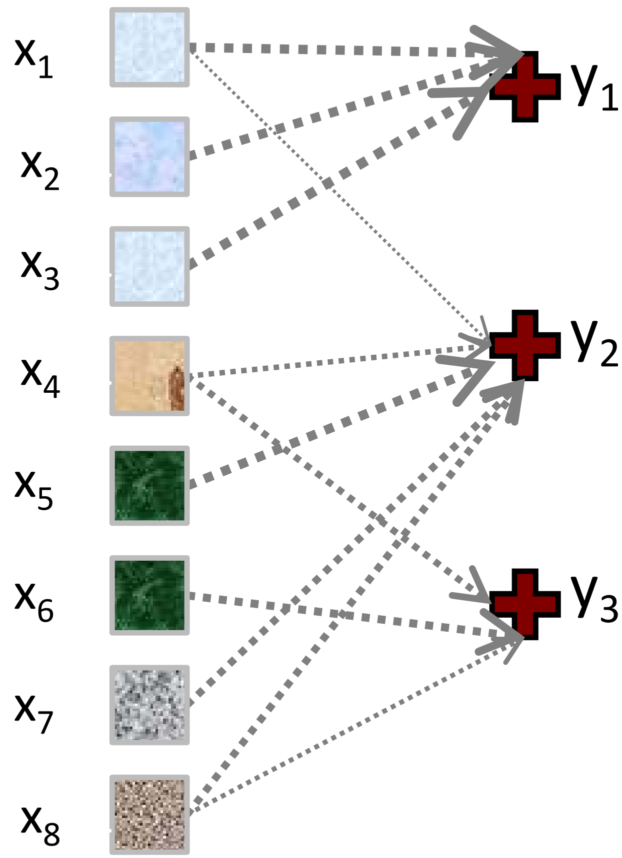


30x speedup

700x speedup

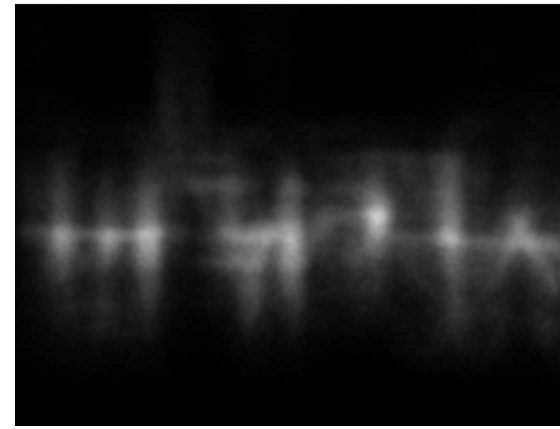
Multiple object detection [Barinova et al.'12]

x_j = index of hypothesis explaining x_j



Voting elements

Hypotheses

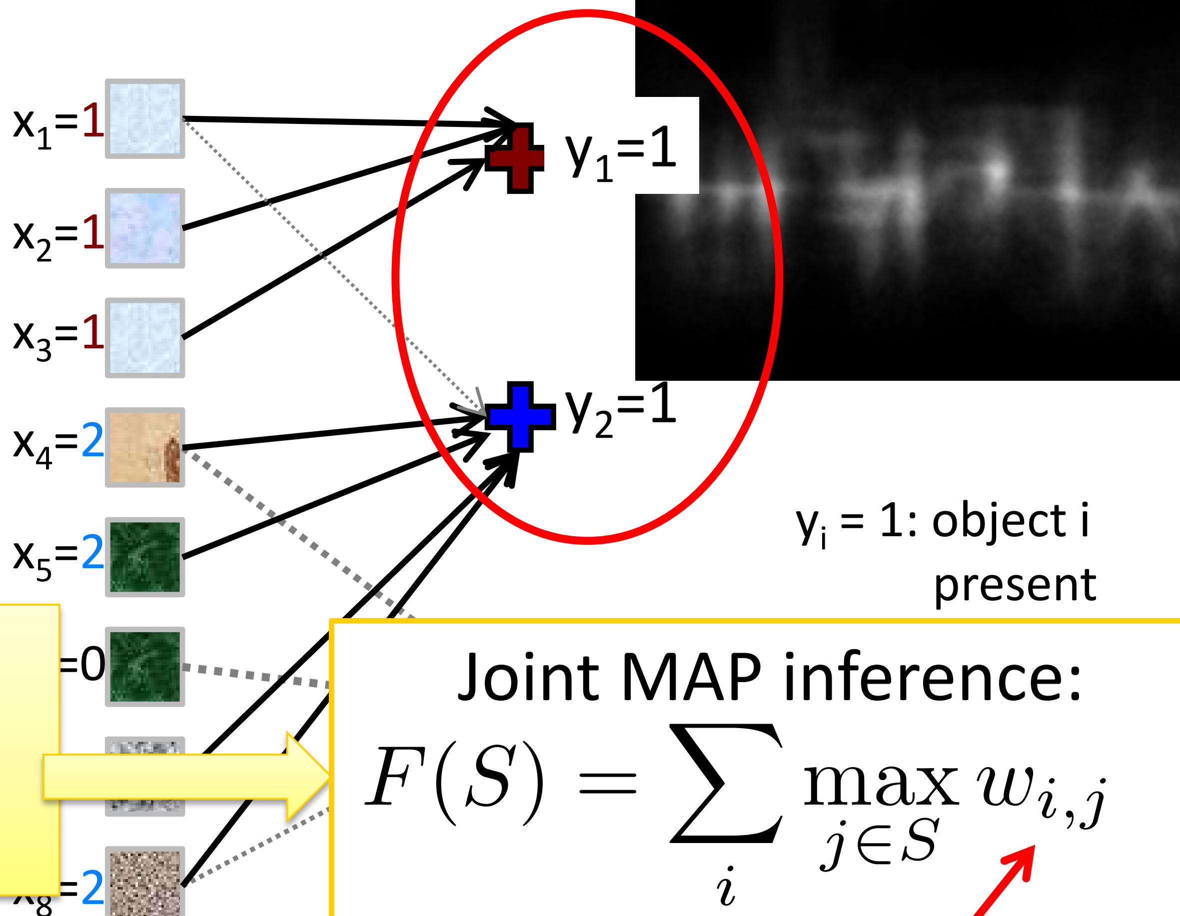


y_i = 1: object i present

y_i = 0: object i not present

Illustrations courtesy of Pushmeet Kohli

Multiple object detection [Barinova et al.'12]



x_j = index of hypothesis explaining x_j

y_i = 1: object i present

submodular maximization ☺

Voting elements

Joint MAP inference:

$$F(S) = \sum_i \max_{j \in S} w_{i,j}$$

Weight element i wrt hyp. j

Illustrations courtesy of Pushmeet Kohli

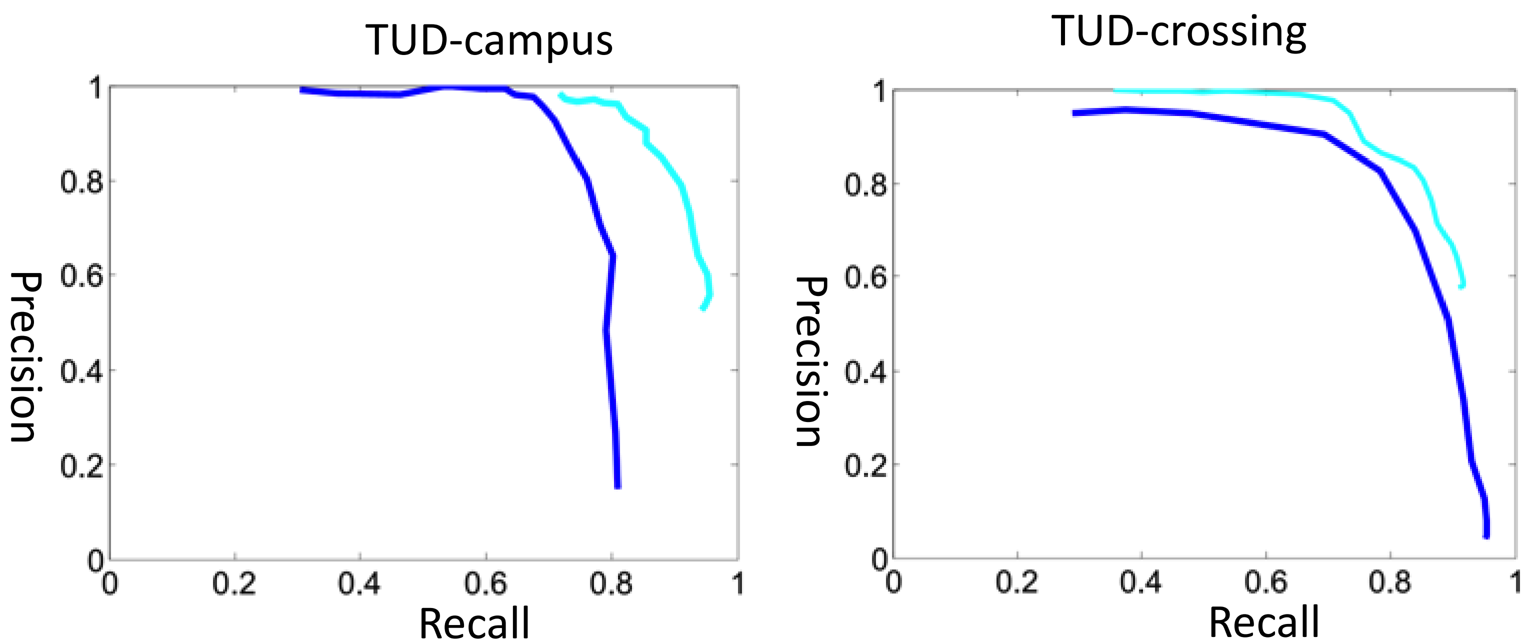
Inference

Datasets from [Andriluka et al. CVPR 2008]
(with strongly occluded pedestrians added)

Using the Hough forest trained in [Gall&Lempitsky CVPR09]

Illustrations courtesy of Pushmeet Kohli

Results for pedestrians detection

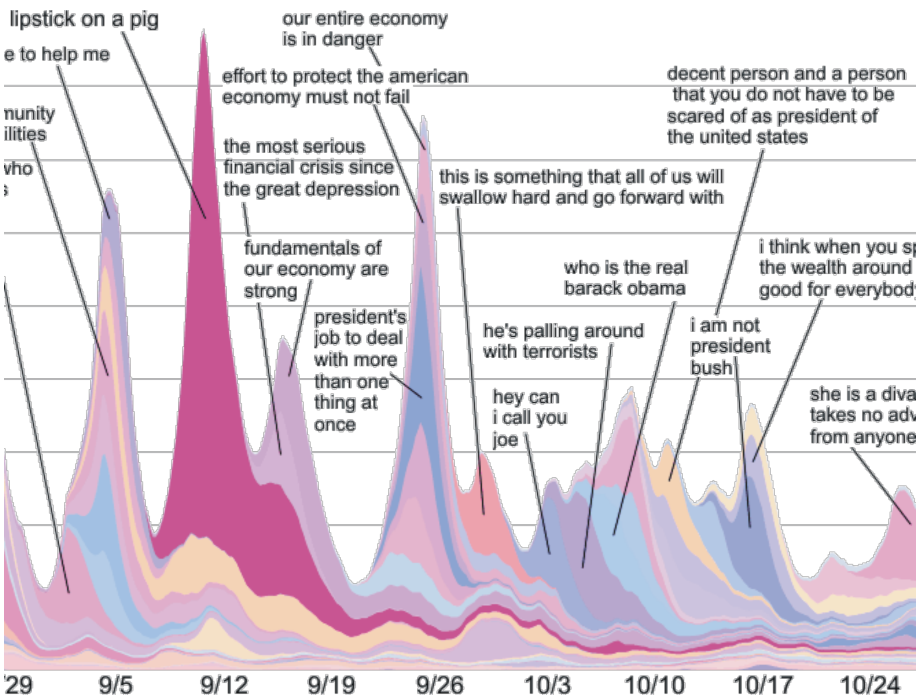


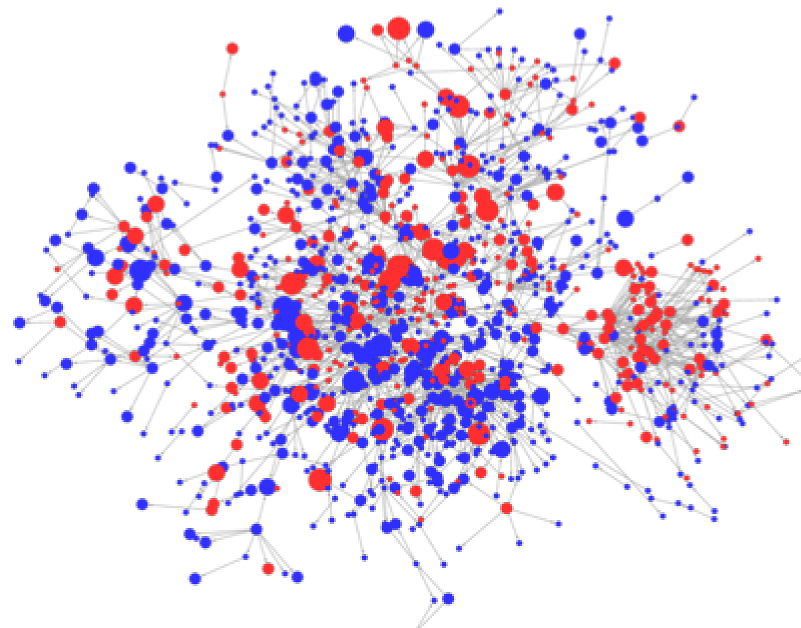
Blue = Hough transform + non-maximum suppression

Light-blue = greedy detection

submodularity for detection also in [Blaschko'11]

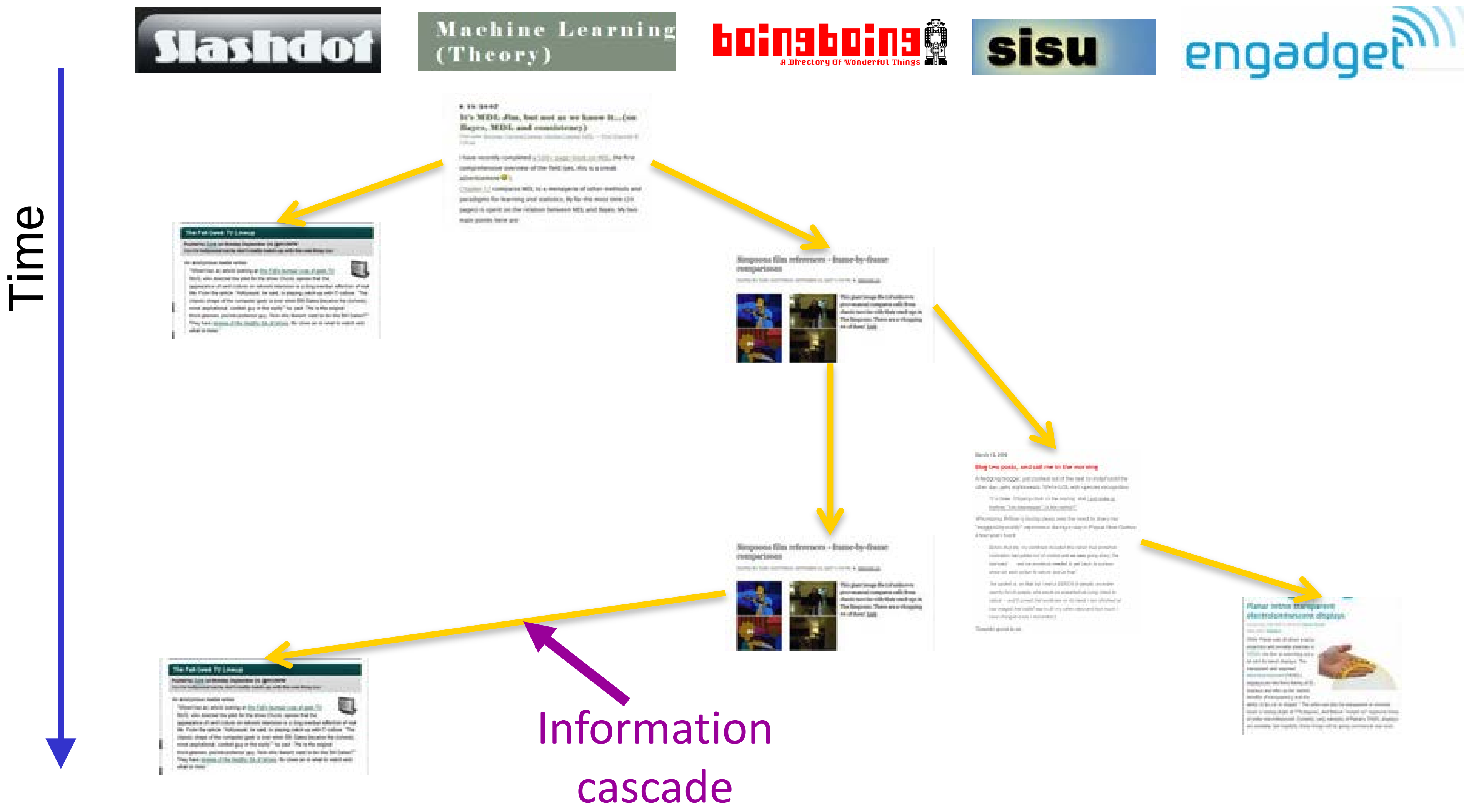
Network inference





How can we learn who influences whom?

Cascades in the Blogosphere



Inferring diffusion networks

[Gomez Rodriguez, Leskovec, Krause ACM TKDE 2012]

Given:

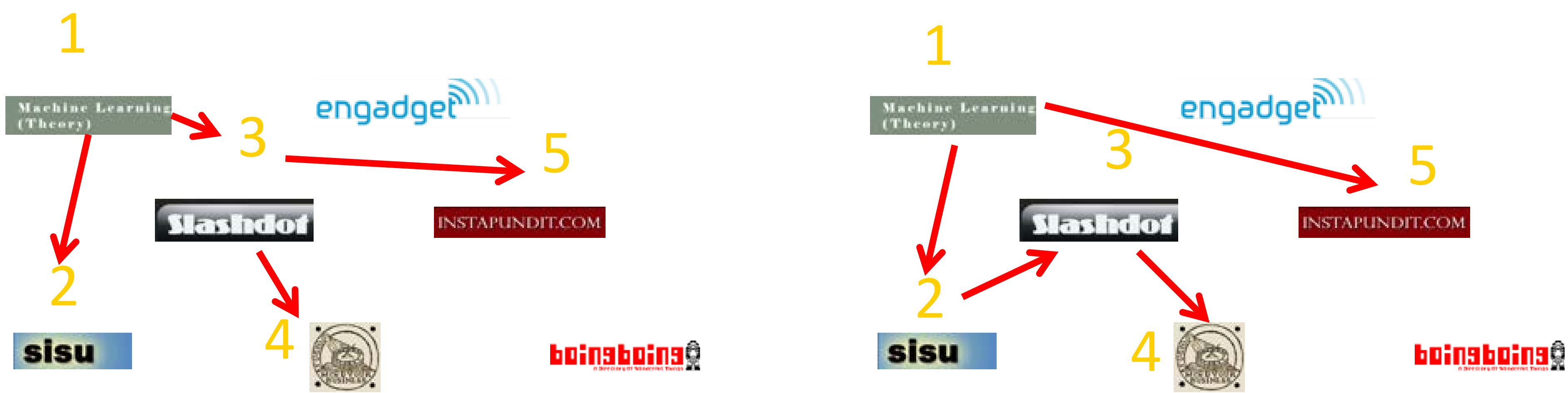
Want:

Given traces of influence, wish to infer sparse directed network G=(V,E)

➔ Formulate as optimization problem

$$E^* = \arg \max_{|E| \leq k} F(E)$$

Estimation problem

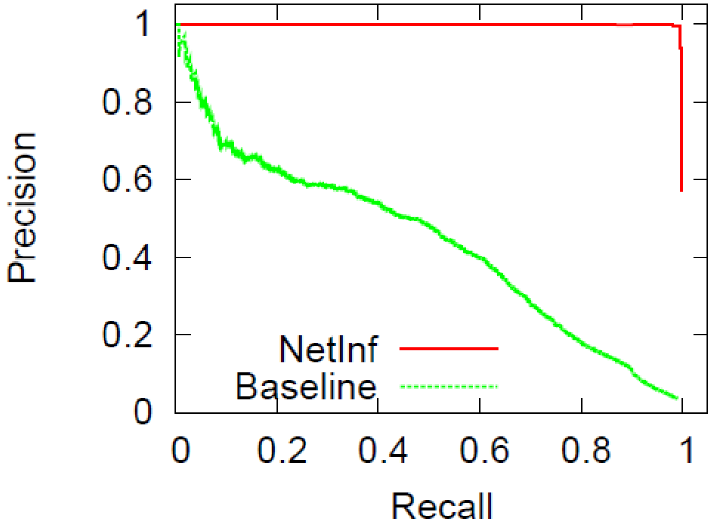


- Many influence trees T consistent with data
- For cascade C_i, model $P(C_i \mid T)$
- Find sparse graph that maximizes likelihood for all observed cascades

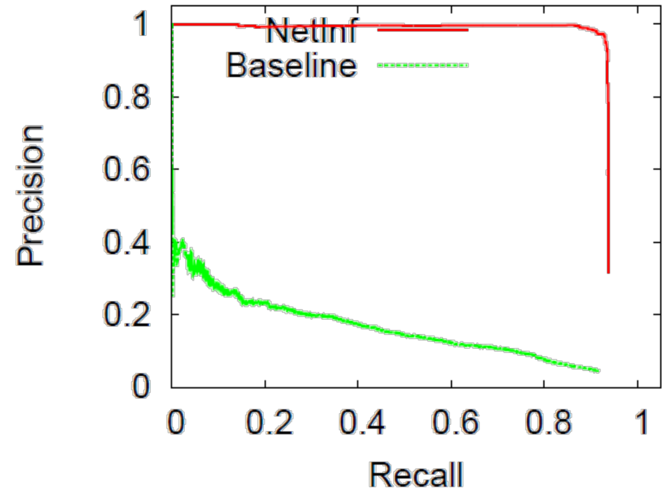
➔ Log likelihood monotonic submodular in selected edges

$$F(E) = \sum_i \log \max_{\text{tree } T \subseteq E} P(C_i \mid T)$$

Evaluation: Synthetic networks



1024 node hierarchical Kronecker exponential transmission model

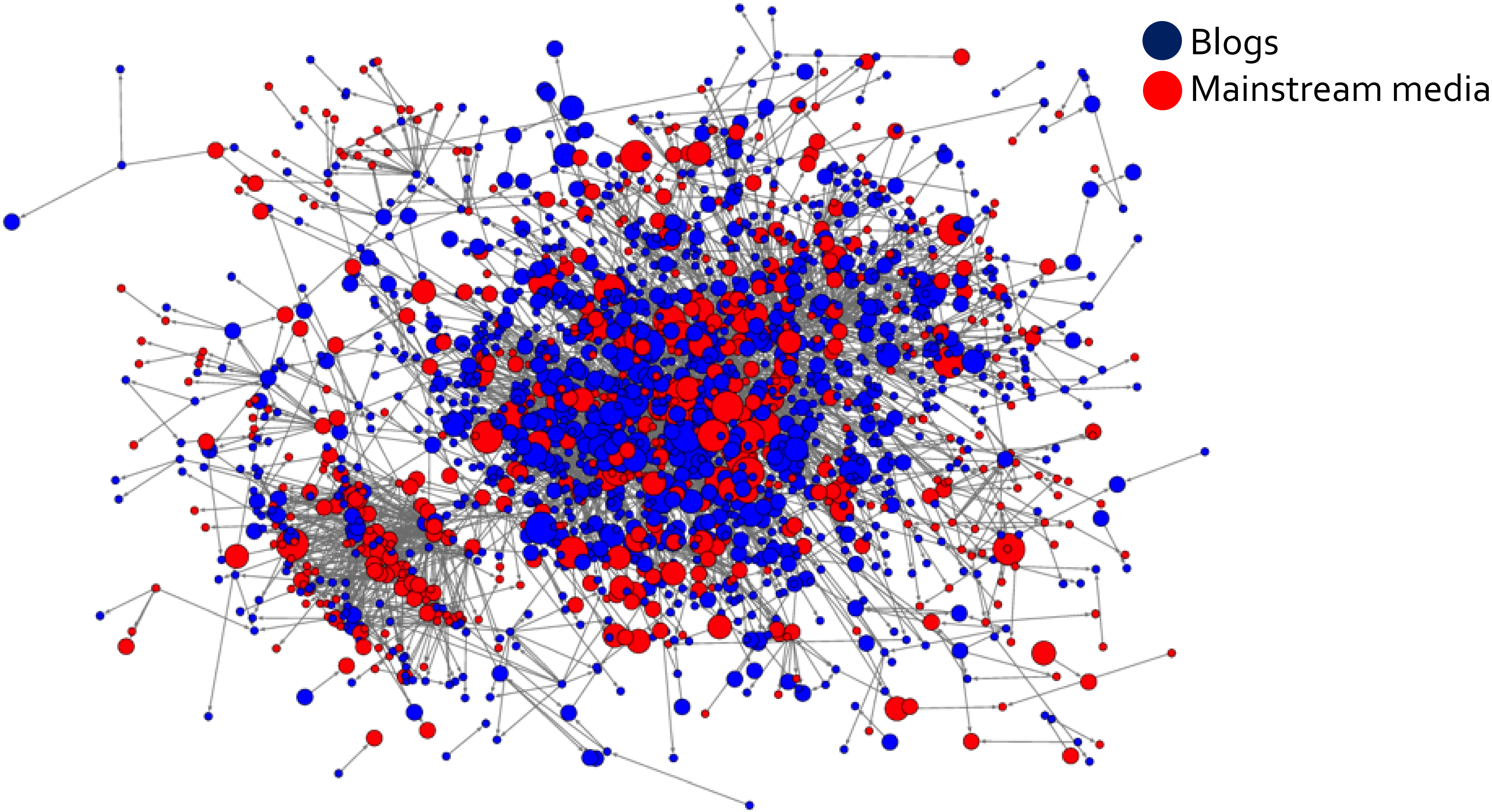


1000 node Forest Fire (α = 1.1) power law transmission model

- Performance does not depend on the network structure:
  - Synthetic Networks: Forest Fire, Kronecker, etc.
  - Transmission time distribution: Exponential, Power Law
- Break-even point of > 90%

Diffusion Network

[Gomez Rodriguez, Leskovec, Krause ACM TKDE 2012]

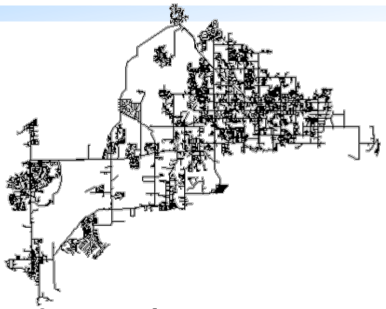


Actual network inferred from 172 million articles from 1 million news sources

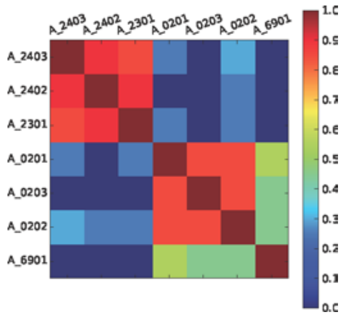
Submodular Sensing Problems

[with Guestrin, Leskovec, Singh, Sukhatme, …]

Environmental monitoring
[UAI'05, JAIR '08, ICRA '10]



Water distribution networks
[J WRPM '08]



Experiment design
[NIPS '10, '11, PNAS'13]

Machine Learning (Theory) Slashdot boingboing sisu engadget

Recommending blogs & news
[KDD '07, '10]

Can all be reduced to monotonic submodular maximization

Maximization: More complex constraints

- Approximate submodular maximization possible under a variety of constraints:
  - (Multiple) matroid constraints
  - Knapsack (non-constant cost functions)
  - Multiple matroid and knapsack constraints
  - Path constraints (Submodular orienteering)
  - Connectedness (Submodular Steiner)
  - Robustness (minimax)
  - ...

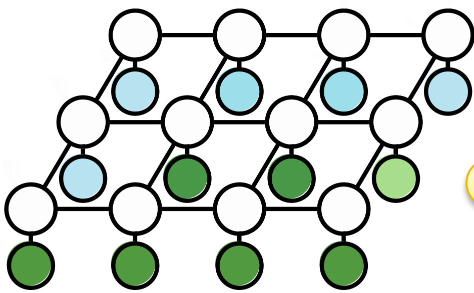
Greedy works well

Need **non-greedy** algorithms

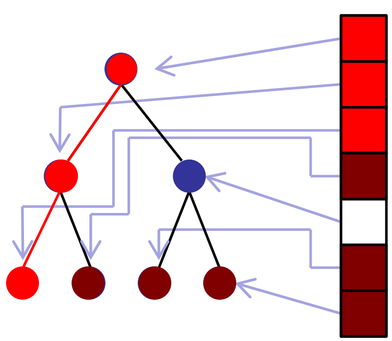
- Survey on „Submodular Function Maximization“ [Krause & Golovin ‘12] on submodularity.org

Two-faces of submodular functions

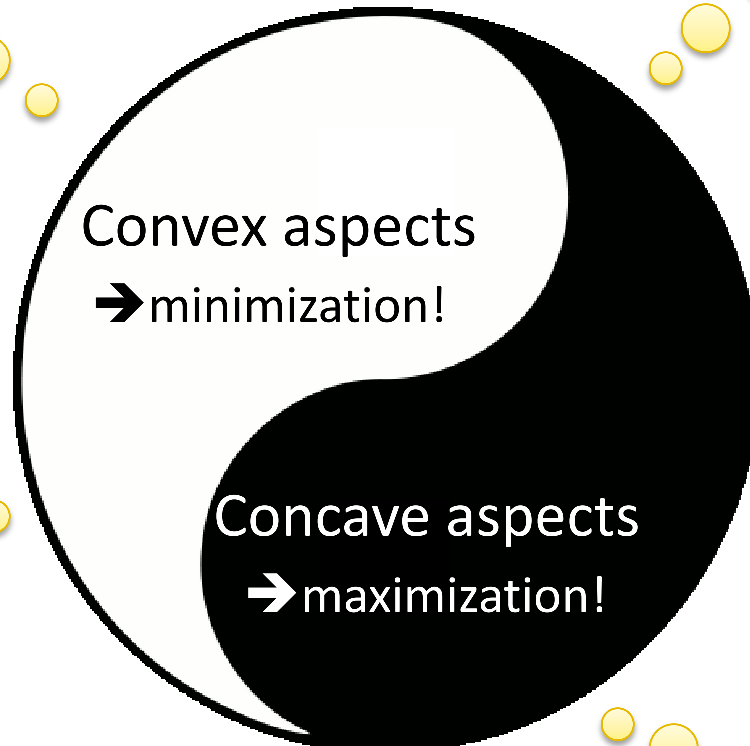
Cuts,
clustering,
similarity



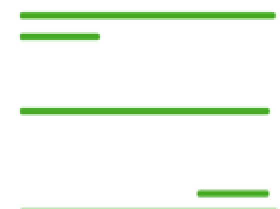
MAP inference



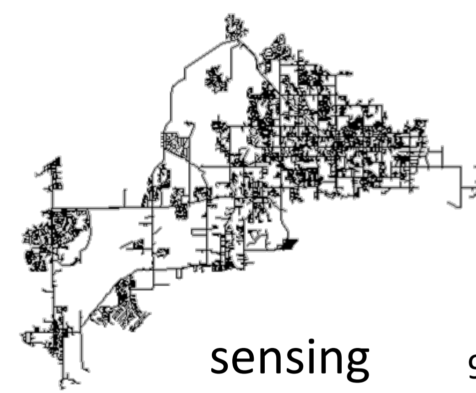
structured sparsity
regularization



Coverage,
diversity



summarization



sensing

	Maximization	**Minimization**
Unconstrained	NP-hard, but well-approximable (if nonnegative)	Polynomial time! Generally ineffcent (n^6), but can exploit special cases (cuts; symmetry; decomposable; ...)
Constrained	NP-hard but well-approximable „Greedy-(like)“ for cardinality, matroid constraints; Non-greedy for more complex (e.g., connectivity) constraints	NP-hard; hard to approximate in general, still useful algorithms

Further topics in submodularity & ML

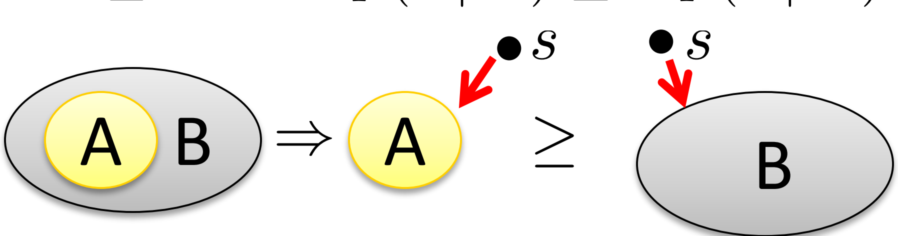
- Learning submodular functions
  - **Goal**: learn a submodular function from few samples
  - **Applications**: Preference elicitation, graph sketching, ...
  - Generally very hard
  - Possible under special structure (e.g., sparsity)
- Online submodular optimization
  - **Goal**: Repeatedly solve submodular optimization problems
  - **Applications**: Recommender systems
  - No regret algorithms for online submodular min & max
- Active learning with submodular functions
  - **Goal**: Adaptive select elements given feedback
  - **Applications**: Active learning, experimental design
  - *Adaptive submodularity* generalizes SFs to policies

From sets to policies

[Golovin & Krause JAIR 2011, IJCAI-JAIR Best Paper 2013]

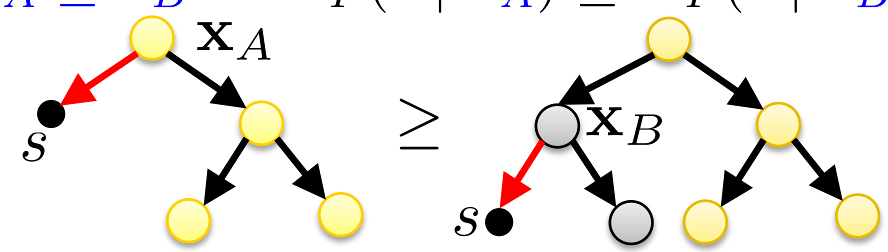
Submodularity	⇨ **Adaptive submodularity**
Applies to: set functions	policies, value functions

$$A \subseteq B \Rightarrow \Delta_F(s \mid A) \geq \Delta_F(s \mid B)$$



$$\max_A F(A)$$

$$\mathbf{x}_A \preceq \mathbf{x}_B \Rightarrow \Delta_F(s \mid \mathbf{x}_A) \geq \Delta_F(s \mid \mathbf{x}_B)$$



$$\max_\pi F(\pi)$$

Greedy algorithm provides

- *(1-1/e)* for max. w card. const.
- *1/(p+1)* for p-indep. systems
- *log Q* for min-cost-cover
- *4* for min-sum-cover

Greedy policy provides

- *(1-1/e)* for max. w card. const.
- *1/(p+1)* for p-indep. systems
- *log Q* for min-cost-cover
- *4* for min-sum-cover

Other directions

- Game theory
  - Equilibria in cooperative (supermodular) games / fair allocations
  - Price of anarchy in non-cooperative games
  - Incentive compatible submodular optimization
- Generalizations of submodular functions
  - L#-convex / discrete convex analysis
  - XOS/Subadditive functions
- More optimization algorithms
  - Robust submodular maximization
  - Maximization and minimization under complex constraints
  - Multilinear extension and applications
  - Submodular-supermodular procedure / semigradient methods

Further resources

- submodularity.org
  - Tutorial Slides
  - Annotated bibliography
  - Matlab Toolbox for Submodular Optimization
  - Links to workshops and related meetings
- discml.cc
  - NIPS Workshops on Discrete Optimization in Machine Learning
  - Videos of invited talks on videolectures.net

…

Conclusions

- Discrete optimization abundant in applications
- Fortunately, some of those have structure: submodularity
- Submodularity can be exploited to develop efficient, scalable algorithms with strong guarantees
- Many exciting research directions! ☺