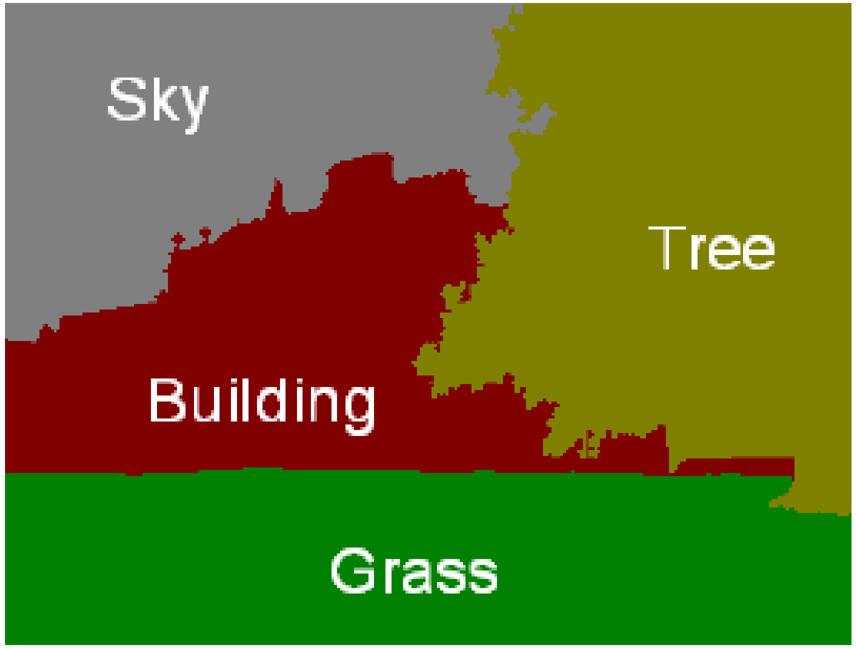
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Tutorial on Submodularity in Machine Learning and Computer Vision

Informatik
Computer Science

Stefanie Jegelka
Andreas Krause

Semantic Segmentation

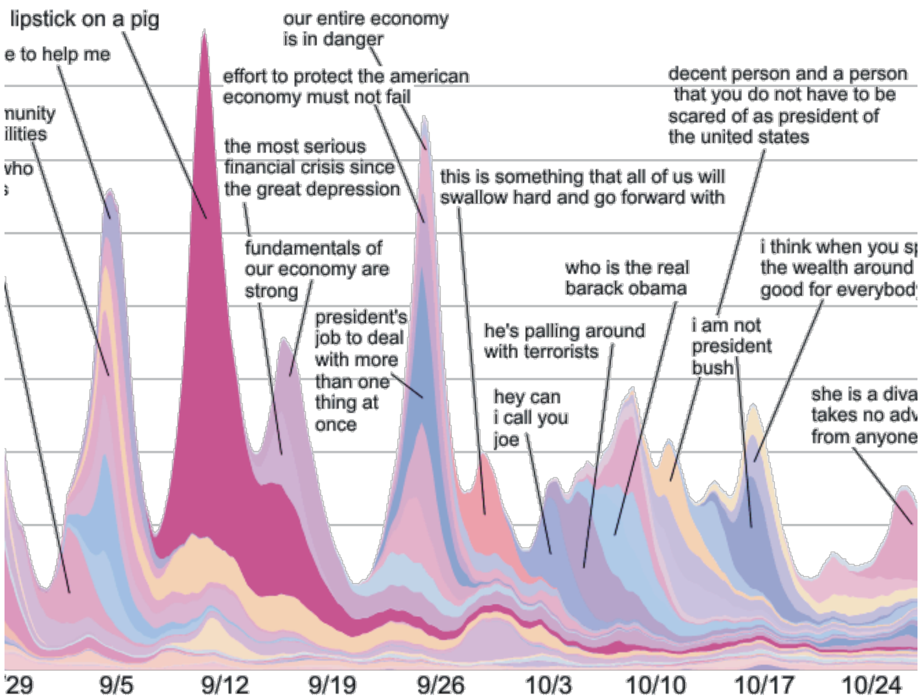


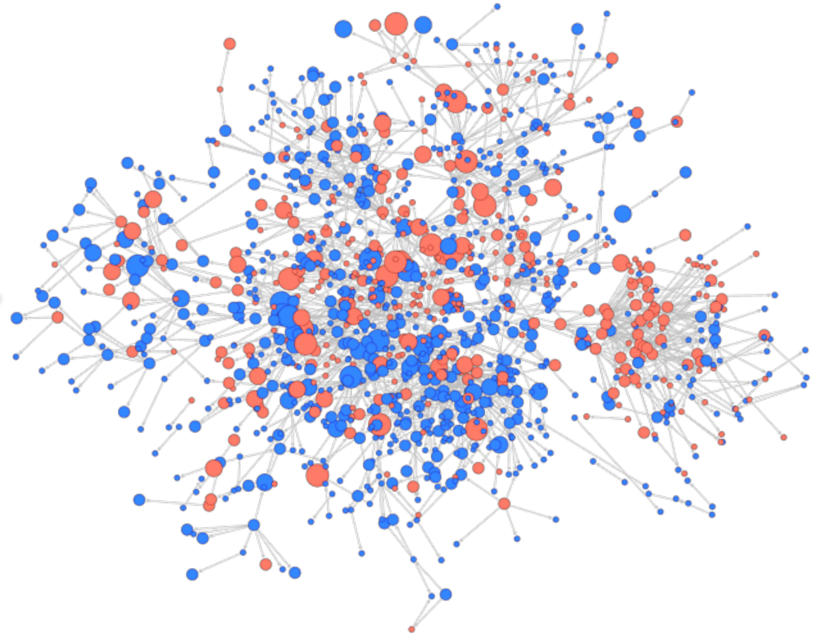
How can we map pixels to objects?

Document Summarization

How can we select representative sentences?

Network inference





How can we learn who influences whom?

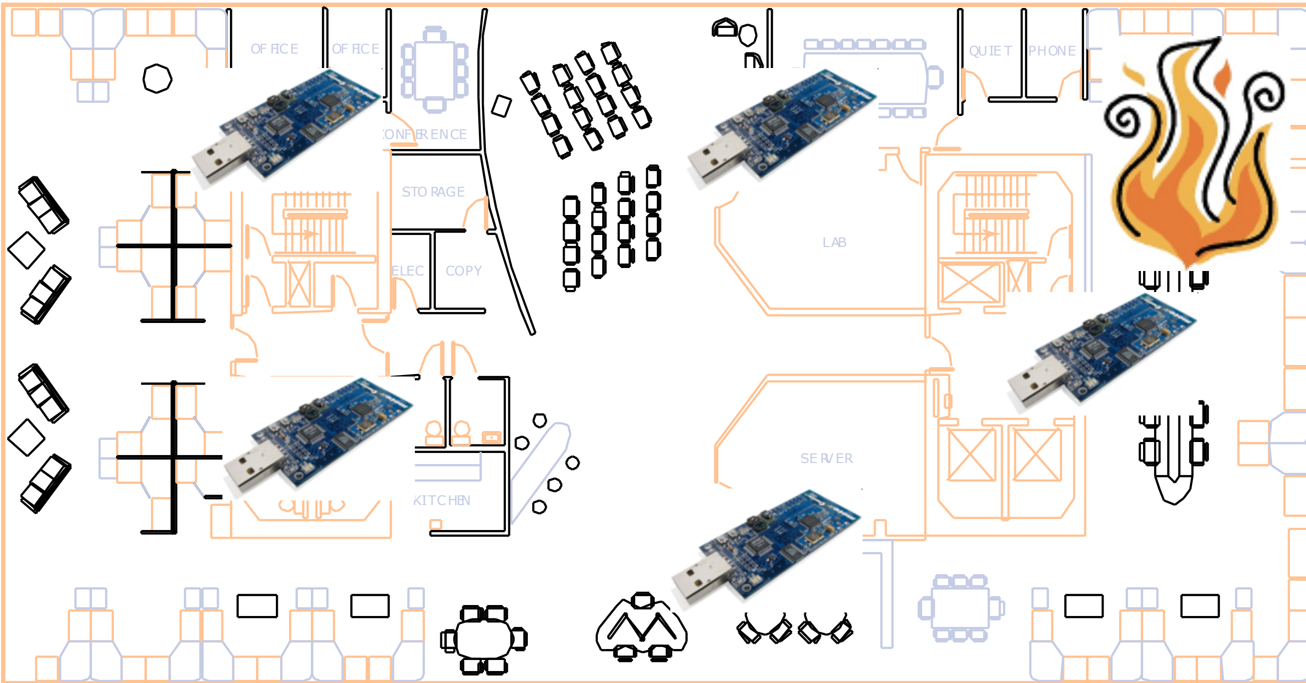
What's common?

- Can be formalized as optimizing a set function F(S) under constraints
- Generally very hard
- Structure helps: We'll see that if F(S) is submodular,
  - can solve maximization and minimization problems with strong guarantees
  - can solve learning problems involving submodular functions
- You'll learn about theory and applications

Outline

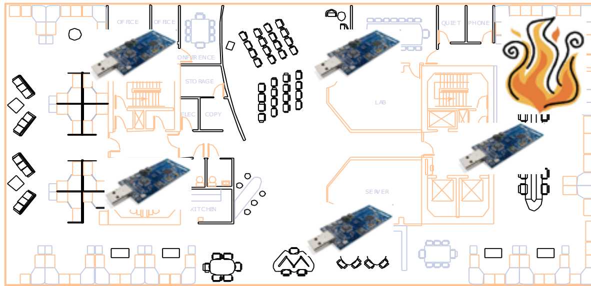
- What is submodularity?
  - Properties of submodular functions
- Optimization
  - Minimization
  - Maximization
- Learning
- Learning for Optimization

Running example: Sensor placement



Want to place sensors to monitor temperature

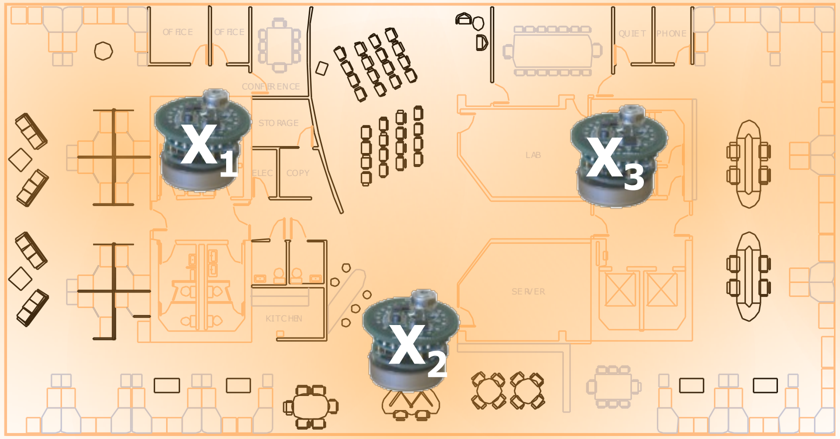
Set functions



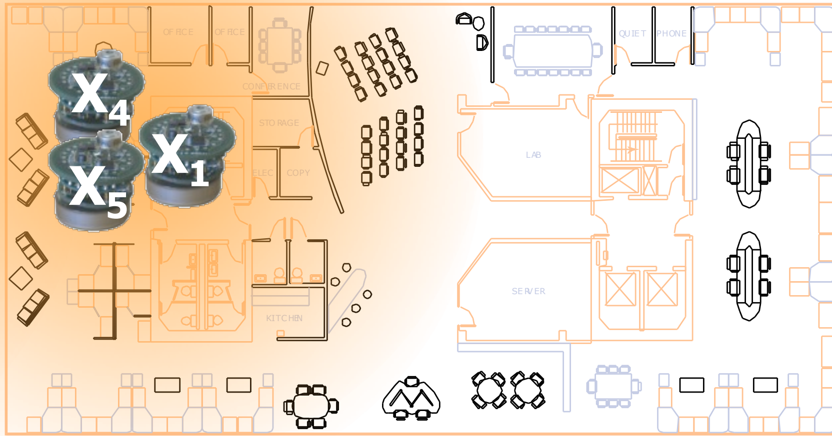
- Finite set V = {1,2,…,n}
- Set function $F : 2^V \rightarrow \mathbb{R}$
- Will always assume F({}) = 0 (w.l.o.g.)
- Assume black-box that can evaluate F for any input A
  - Approximate (noisy) evaluation of F is ok

Example: Sensor placement

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)

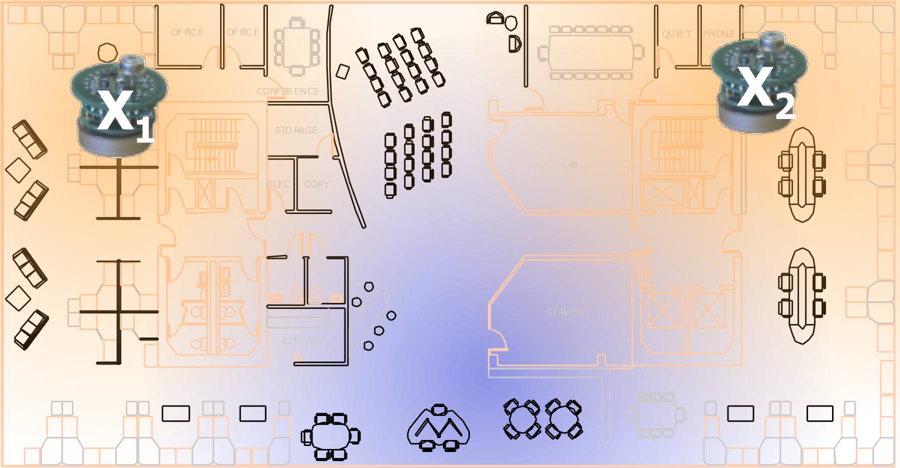
Marginal gains

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

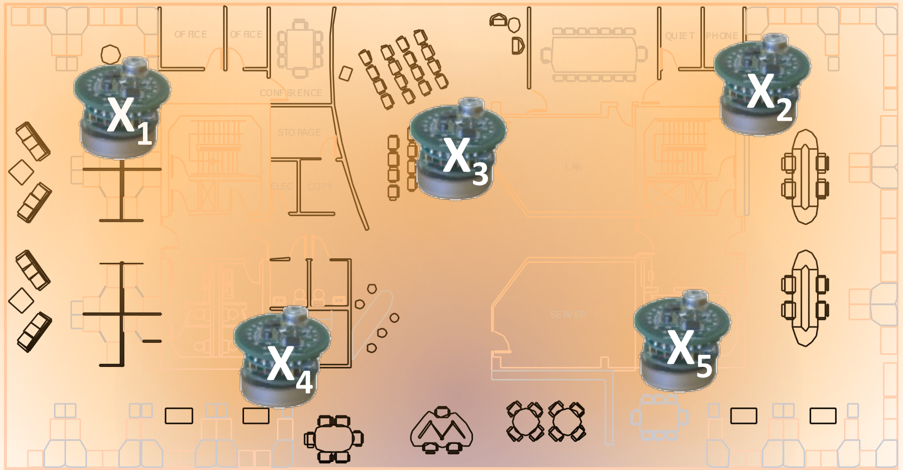
New sensor s

Submodularity: Decreasing marginal gains

Placement A = {1,2}



Placement B = {1,…,5}

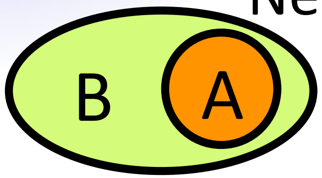


Adding *s* will help a lot!

New sensor s

Adding *s* doesn't help much

Submodularity:



+ • s Large improvement

+ • s Small improvement

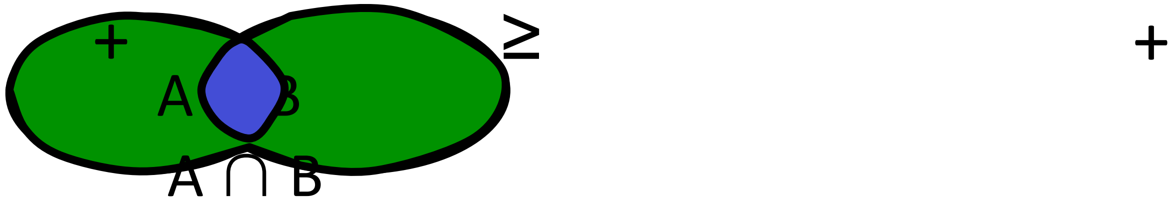
For $A \subseteq B : F(A \cup \{s\}) - F(A) \geq F(B \cup \{s\}) - F(B)$

$$\Delta(s \mid A) \quad \geq \quad \Delta(s \mid B)$$

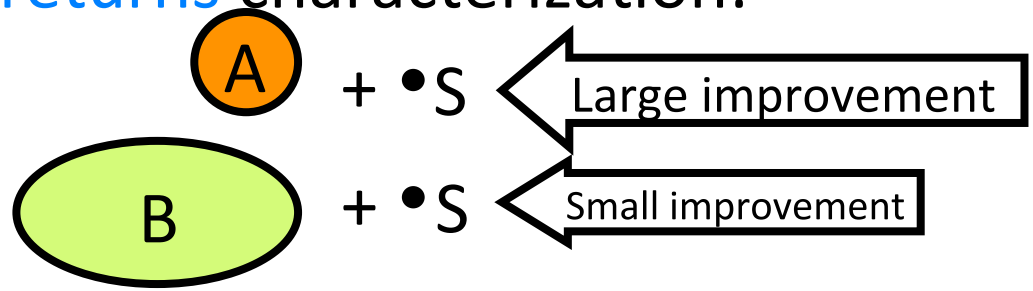
Alternative characterizations

- Set function F on V is called submodular if

$$\forall A, B \subseteq V : F(A) + F(B) \geq F(A \cup B) + F(A \cap B)$$



- Equivalent diminishing returns characterization:



$$\forall A \subseteq B \subseteq V, s \notin B :$$

$$F(A \cup \{s\}) - F(A) \geq F(B \cup \{s\}) - F(B)$$

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

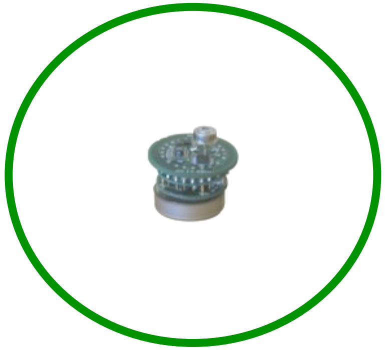
Questions

How do I prove my problem is submodular?

Why is submodularity useful?

Example: Set cover

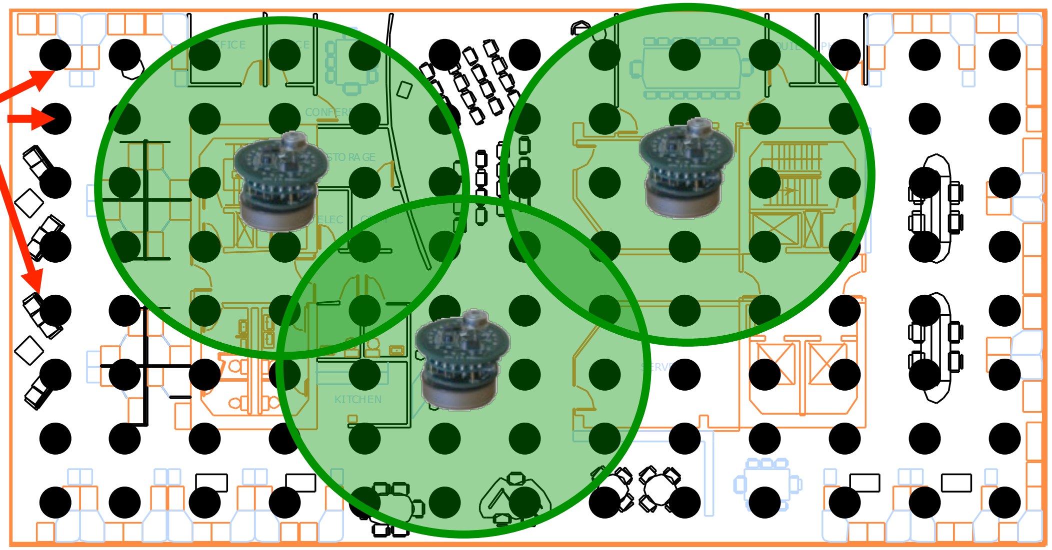
Place sensors in building



Node predicts values of positions with some radius

Want to cover floorplan with discs

Possible locations V



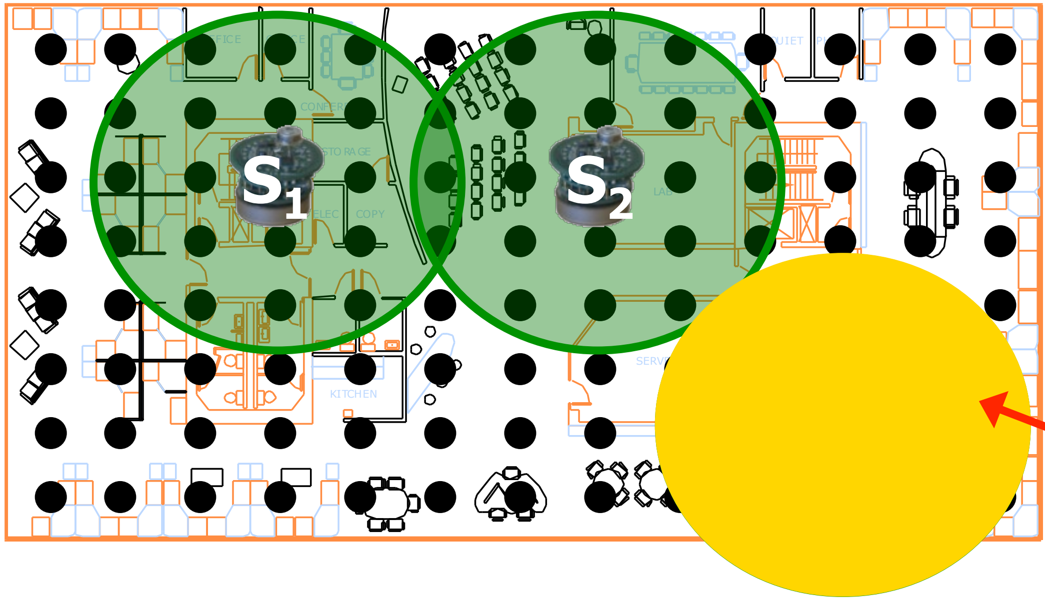
For $A \subseteq V$: F(A) = "area covered by sensors placed at A"

Formally:

W finite set, 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

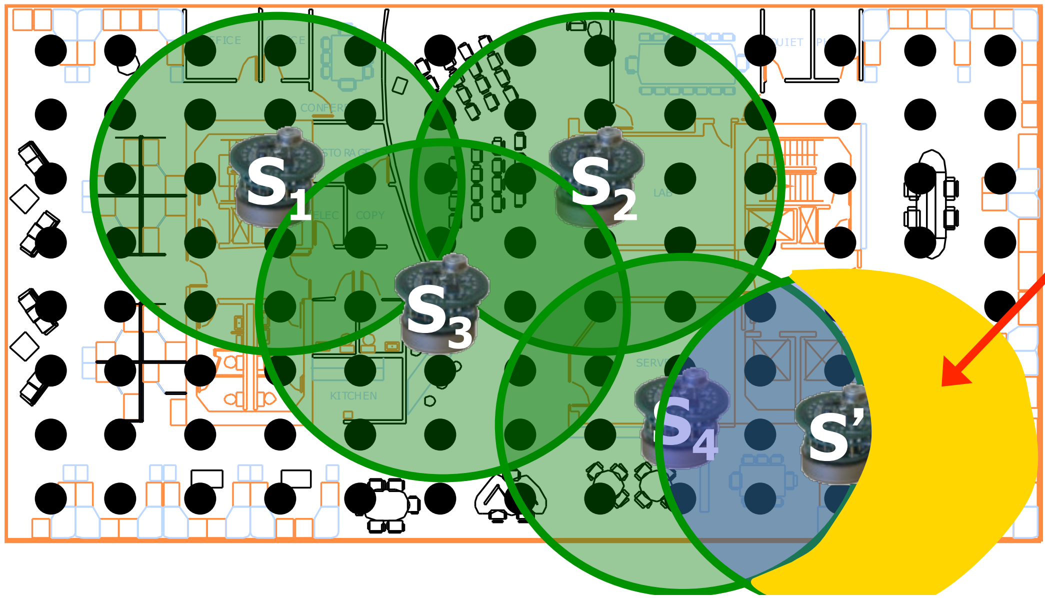


$A=\{S_1,S_2\}$

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

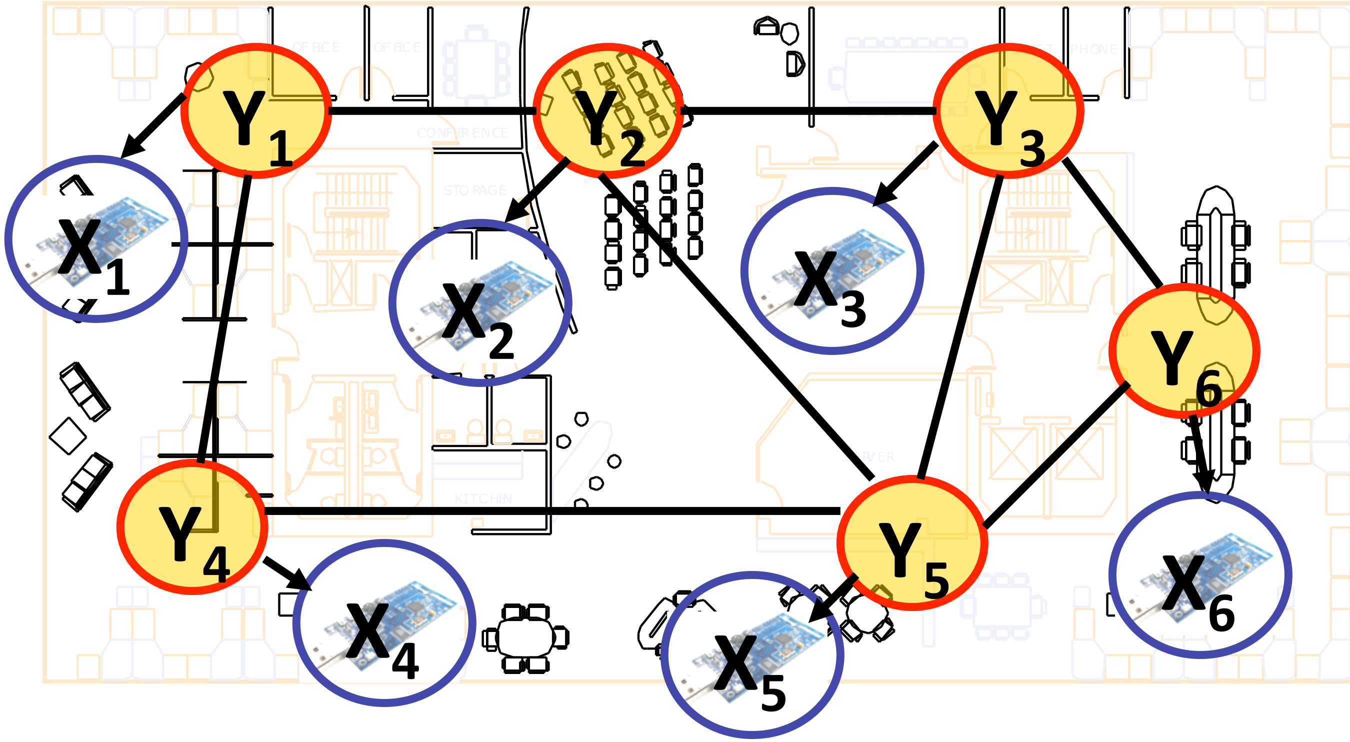
$\geq$

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



$B = \{S_1,S_2,S_3,S_4\}$

More complex model for sensing



Y_s: temperature at location s

X_s: sensor value at location s

$$\mathbf{X_s = Y_s + noise}$$

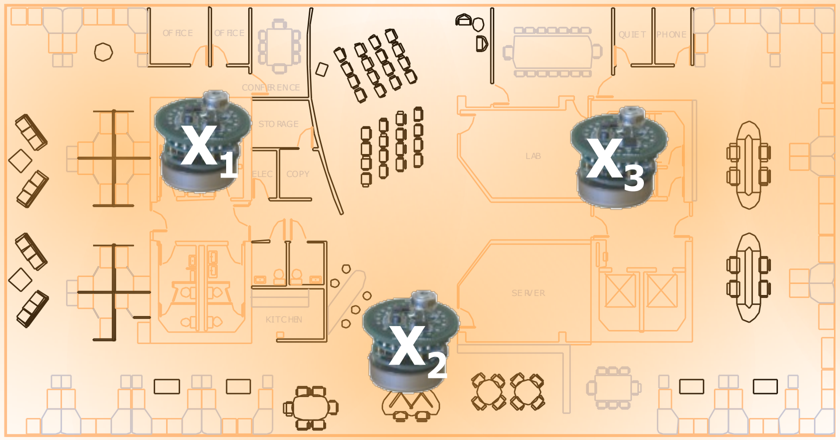
Joint probability distribution

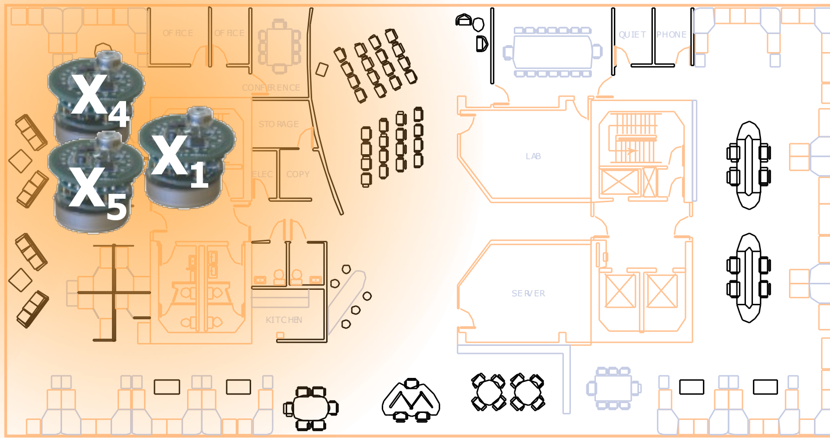
$$P(X_1,\ldots,X_n,Y_1,\ldots,Y_n) = \underbrace{P(Y_1,\ldots,Y_n)}_{\text{Prior}}\ \underbrace{P(X_1,\ldots,X_n \mid Y_1,\ldots,Y_n)}_{\text{Likelihood}}$$

Example: Sensor placement

Utility of having sensors at subset A of all locations

$$F(A) = \underbrace{H(\mathbf{Y})}_{\substack{\textbf{Uncertainty}\\ \textbf{about temperature Y}\\ \textbf{before}\text{ sensing}}} - \underbrace{H(\mathbf{Y} \mid \mathbf{X}_A)}_{\substack{\textbf{Uncertainty}\\ \textbf{about temperature Y}\\ \textbf{after}\text{ sensing}}}$$





A={1,2,3}: High value F(A) A={1,4,5}: Low value F(A)

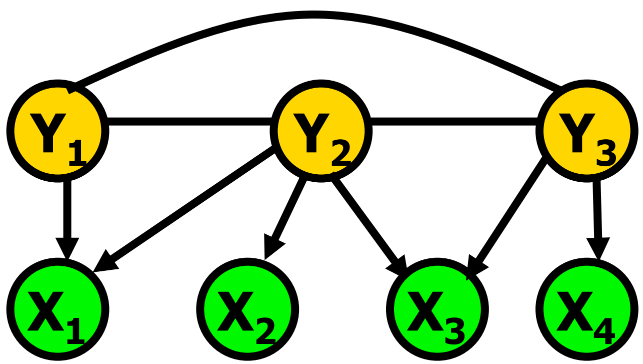
Example: Submodularity of info-gain

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

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

- F(A) is NOT always submodular

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



Proof: Submodularity of information gain

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

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

Variables X independent given Y

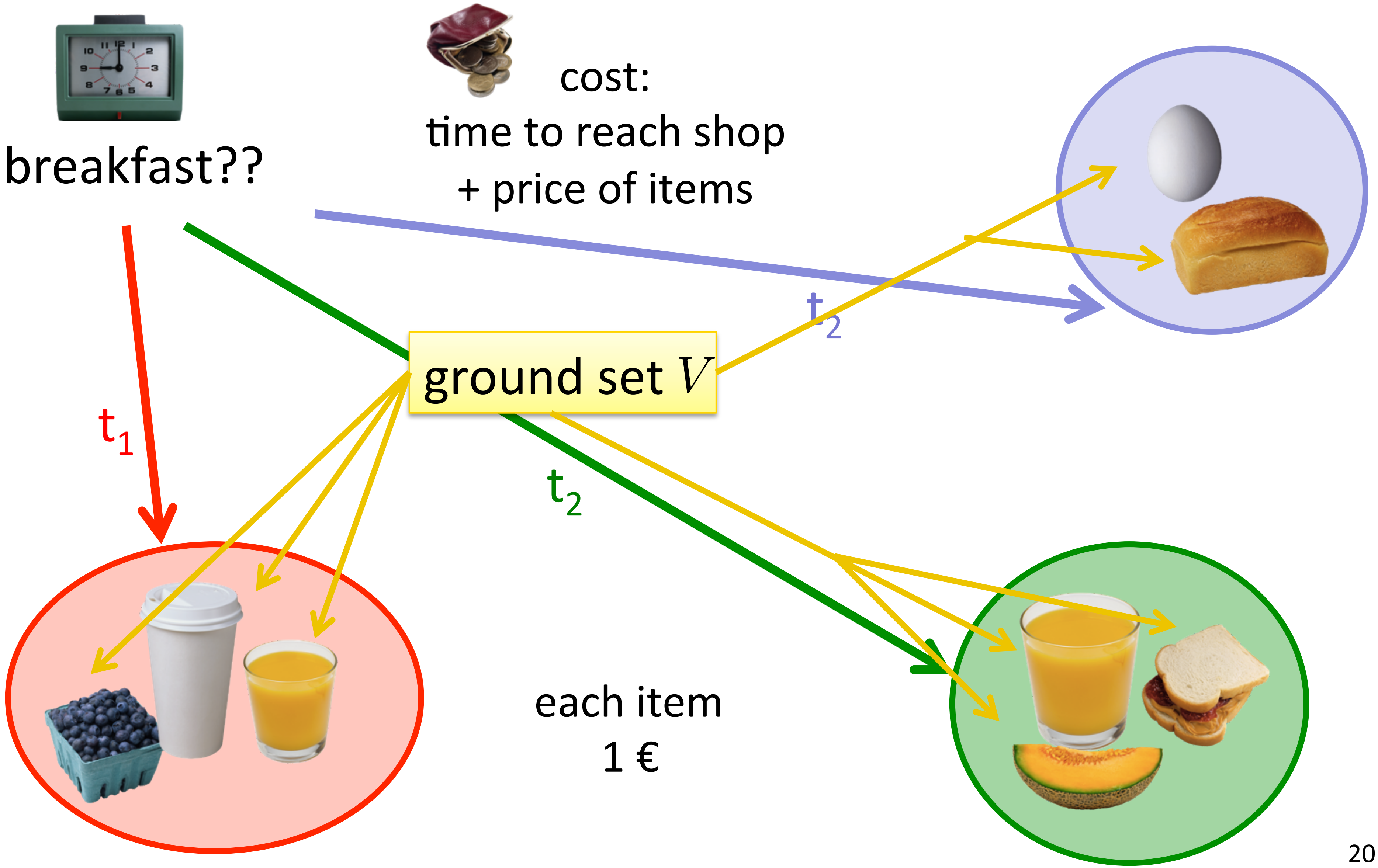
$$F(A \cup \{s\}) - F(A) = \underbrace{H(X_s \mid X_A)}_{\substack{\text{Nonincreasing in A:} \\ A \subseteq B \,:\, H(X_s|X_A) \geq H(X_s|X_B)}} - \underbrace{H(X_s \mid Y)}_{\substack{\text{Constant} \\ \text{(indep. of A)}}}$$

„information never hurts"

$\Delta(s \mid A) = F(A \cup \{s\})-F(A)$ monotonically nonincreasing

$\Leftrightarrow$ F submodular ☺

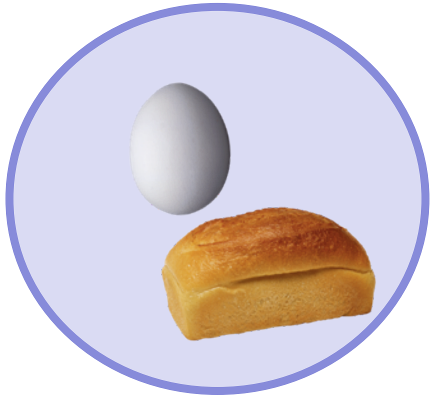
Example: costs



Example: costs

breakfast??

cost:
time to shop
+ price of items

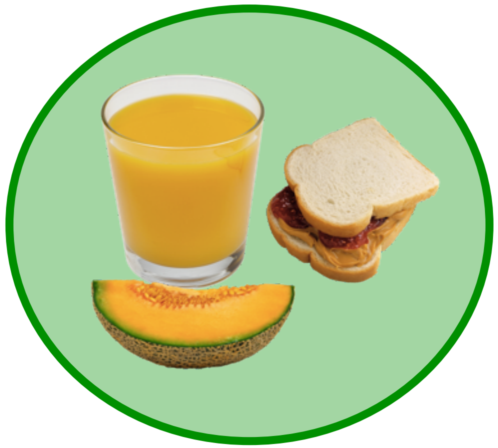


F(  ) = cost() + cost( , )

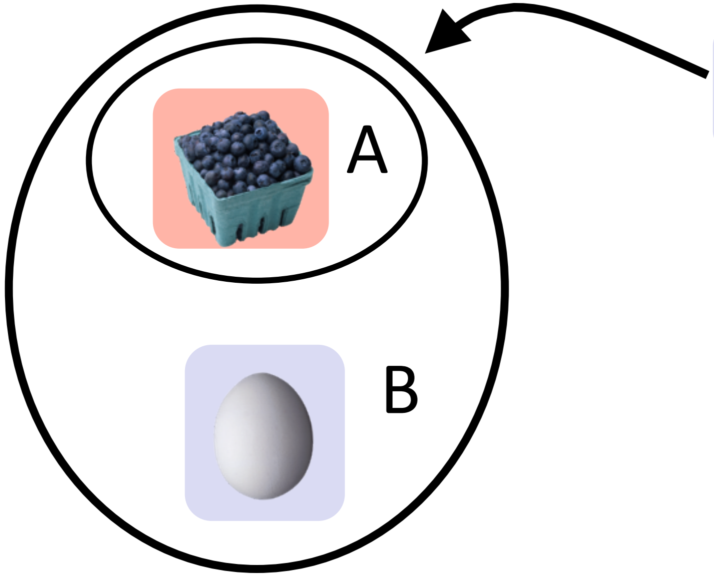
$= t_1 + 1 + t_2 + 2$

$= \#\text{shops} + \#\text{items}$

submodular?

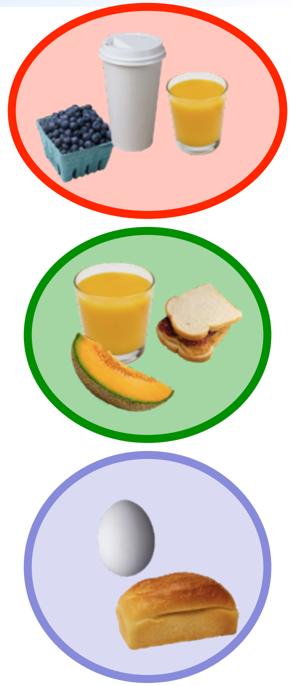


Shared fixed costs



$$\Delta(b|A) = 1 + 1$$

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

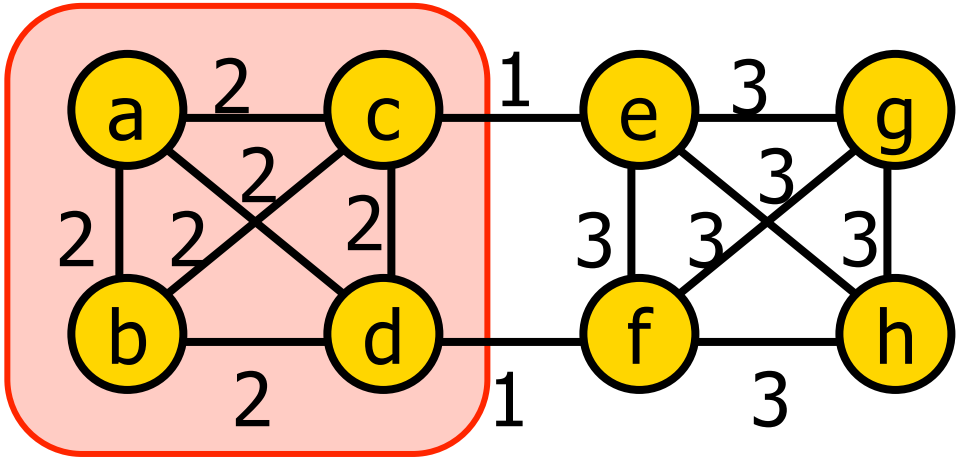


marginal cost: #new shops + #new items

diminishing ➔ cost is submodular!

- shops: shared fixed cost
- economies of scale

Another example: Cut functions



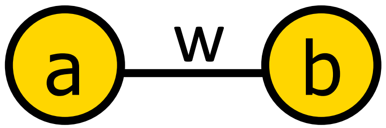
V={a,b,c,d,e,f,g,h}

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

–

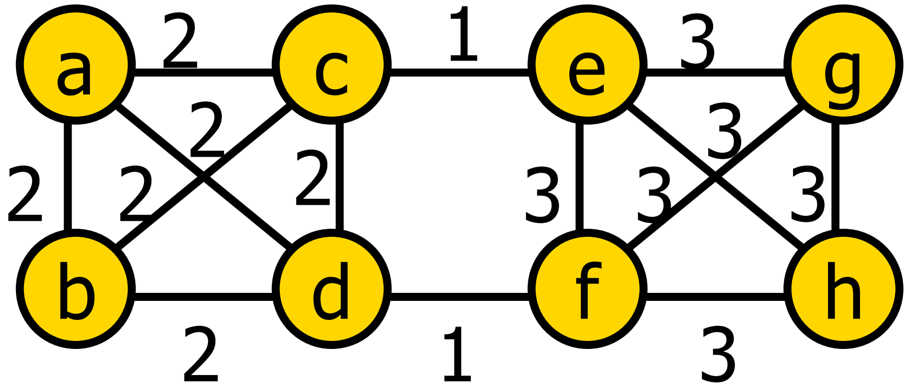
Cut function is submodular!

Why are cut functions submodular?



S	$F_{ab}(S)$
{}	0
{a}	w
{b}	w
{a,b}	0

Submodular if w ≥ 0!



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

➔ 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!

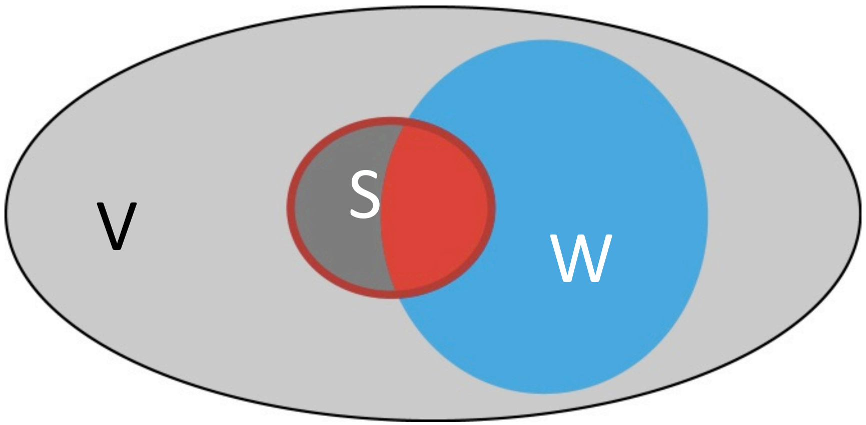
Extremely useful fact:

- $F_\theta(A)$ submodular $\Rightarrow \sum_\theta P(\theta) F_\theta(A)$ submodular!
- Multicriterion optimization:
  $F_1,\ldots,F_m$ submodular, $\lambda_i>0 \Rightarrow \sum_i \lambda_i F_i(A)$ submodular
- 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



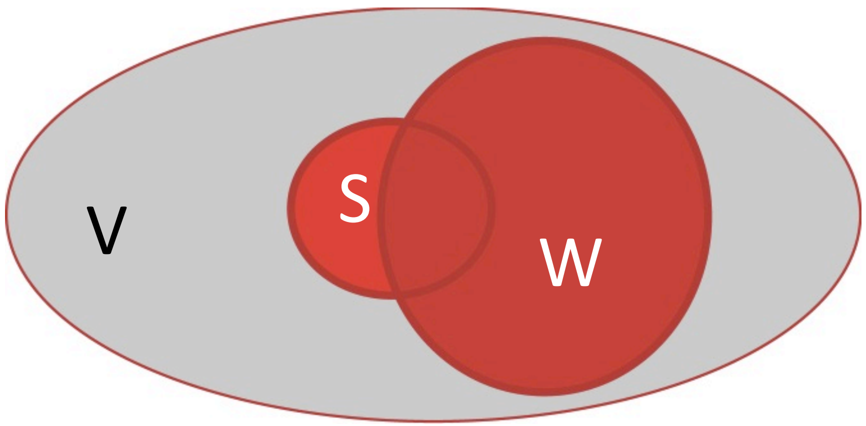
Other closedness properties

- 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

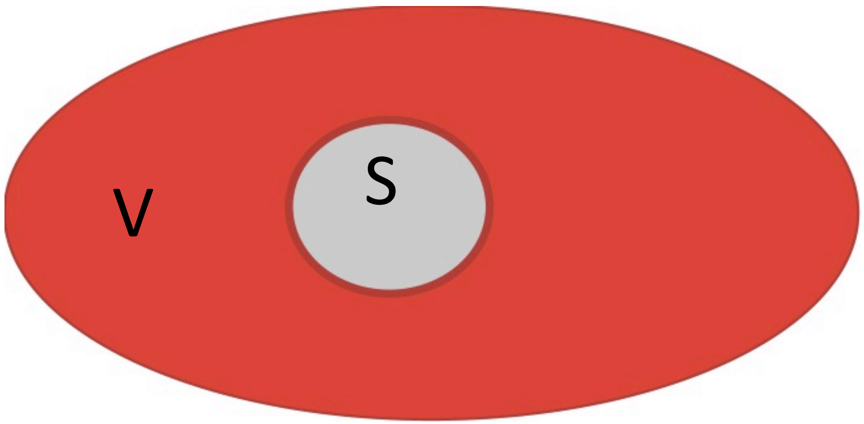
  Then $$F'(S) = F(S \cap W)$$ is submodular

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

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

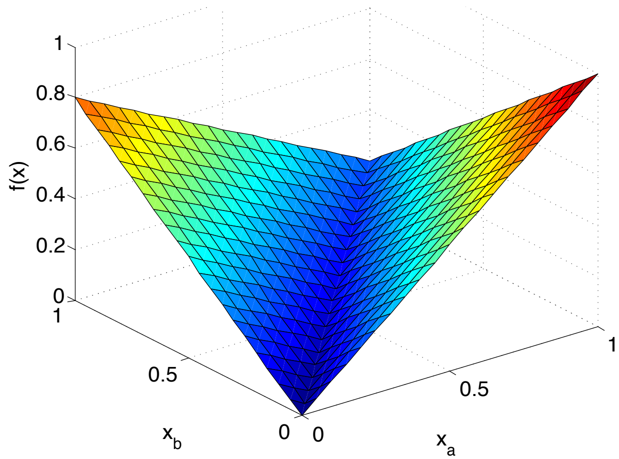
- Reflection: F(S) submodular on

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



Convex aspects

- Submodularity as discrete analogue of convexity
  - Convex extension
  - Efficient minimization
  - Duality



- However, this is only one half of the story...

Concave aspects

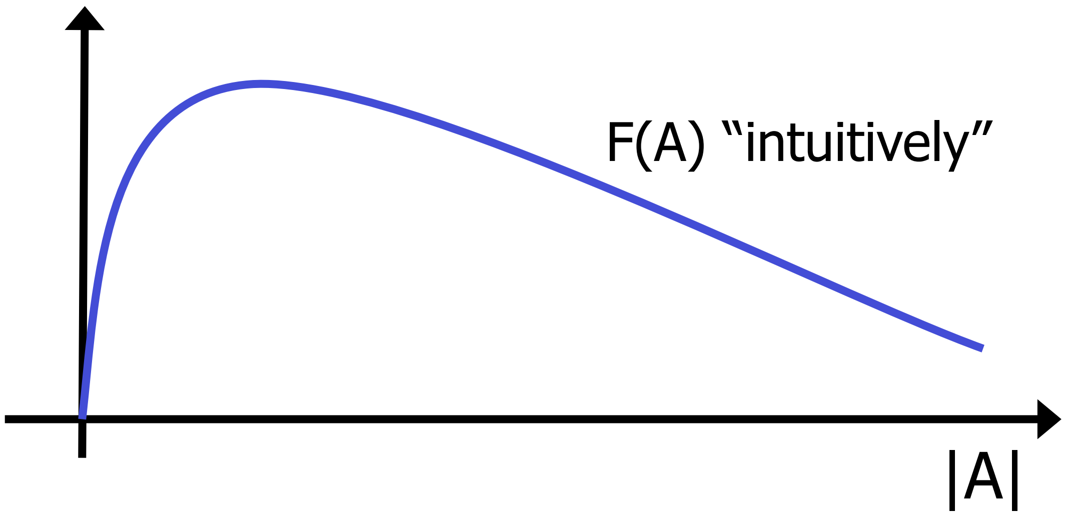
- Marginal gain $\Delta_F(s \mid A) = F(\{s\} \cup A) - F(A)$

- Submodular:

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

- Concave:

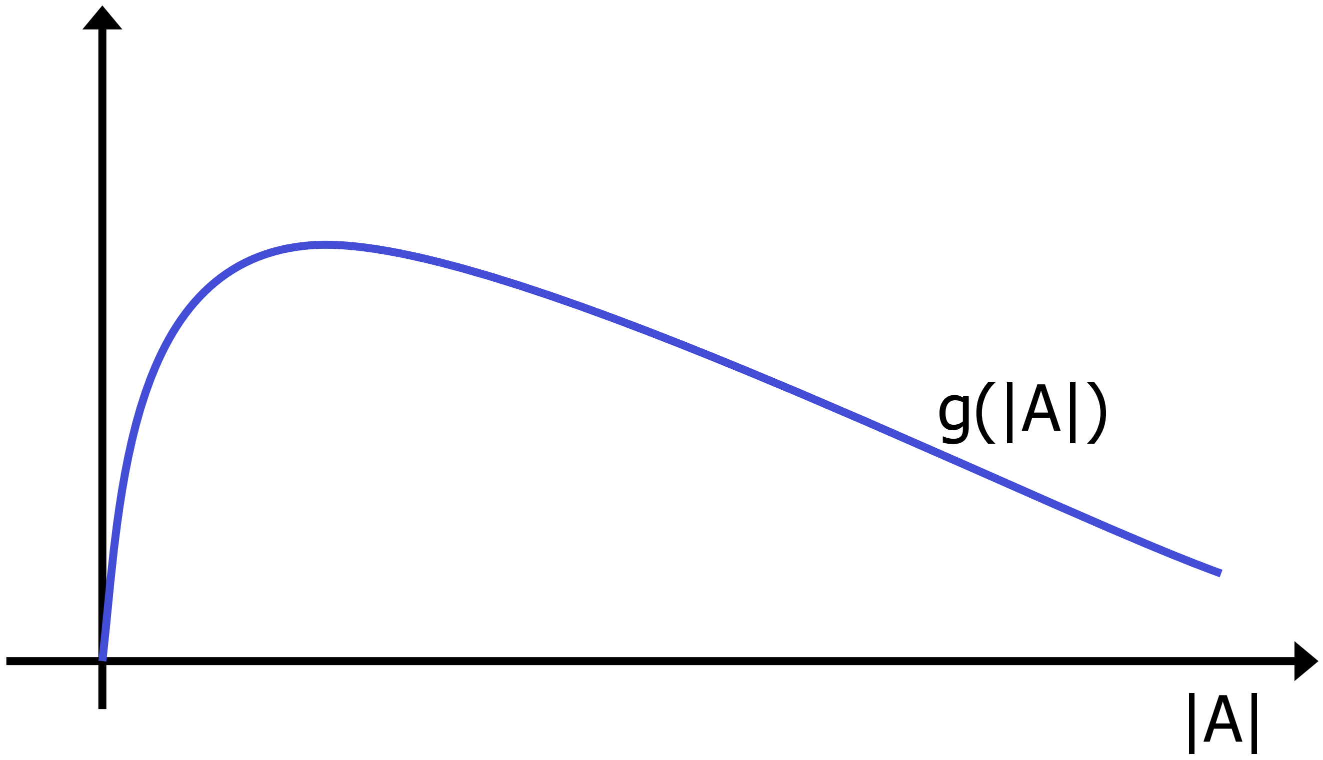
$$\forall\, a < b,\ s > 0 \qquad f(a+s) - f(a) \geq f(b+s) - f(b)$$



Submodularity and Concavity

Suppose $g: N \rightarrow R$ and $F(A) = g(|A|)$

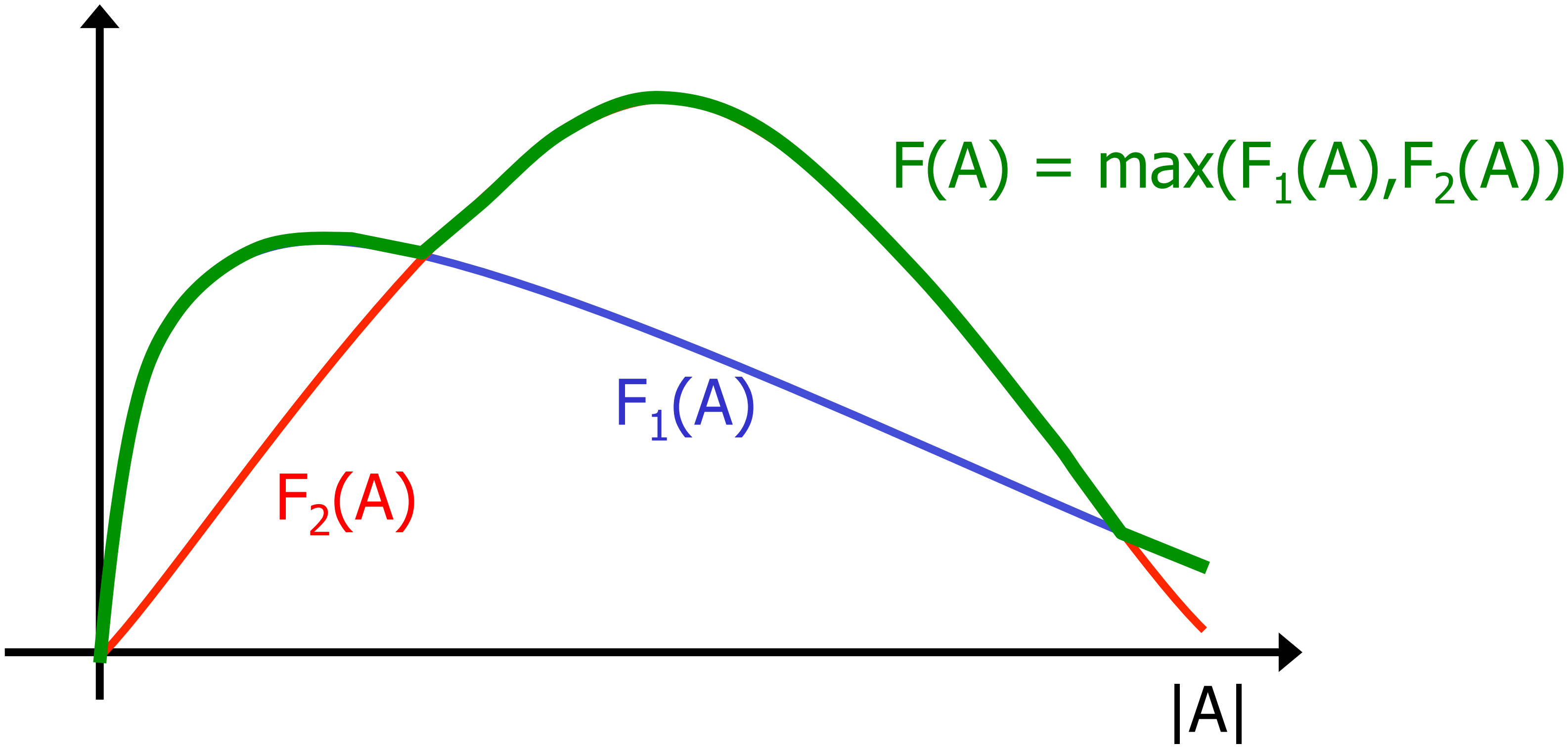
Then $F(A)$ submodular *if and only if* g is concave !



Maximum of submodular functions

Suppose $F_1(A)$ and $F_2(A)$ submodular.

Is $F(A) = \max(F_1(A),F_2(A))$ submodular?



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

Minimum of submodular functions

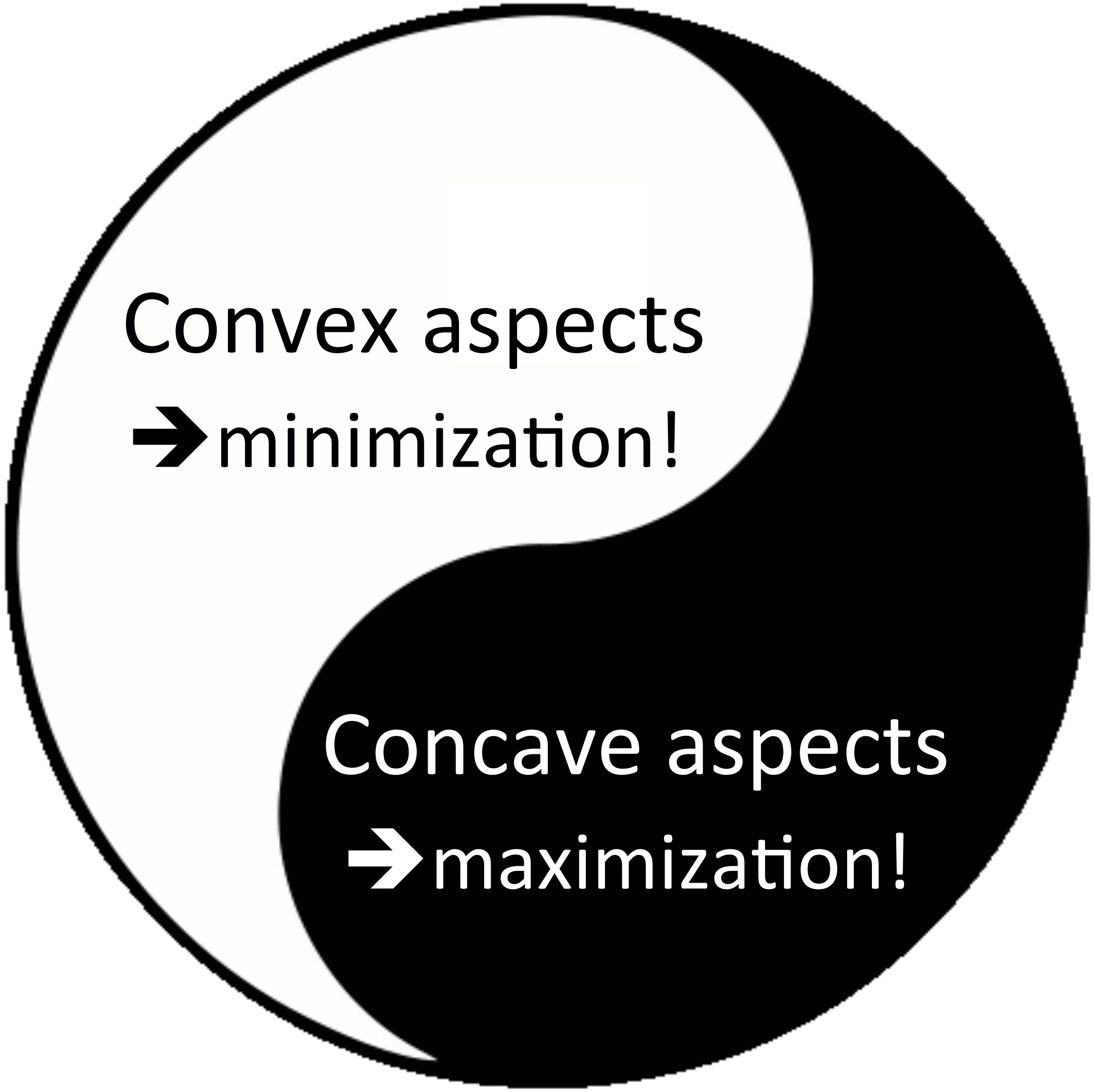
Well, maybe $F(A) = \min(F_1(A),F_2(A))$ instead?

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

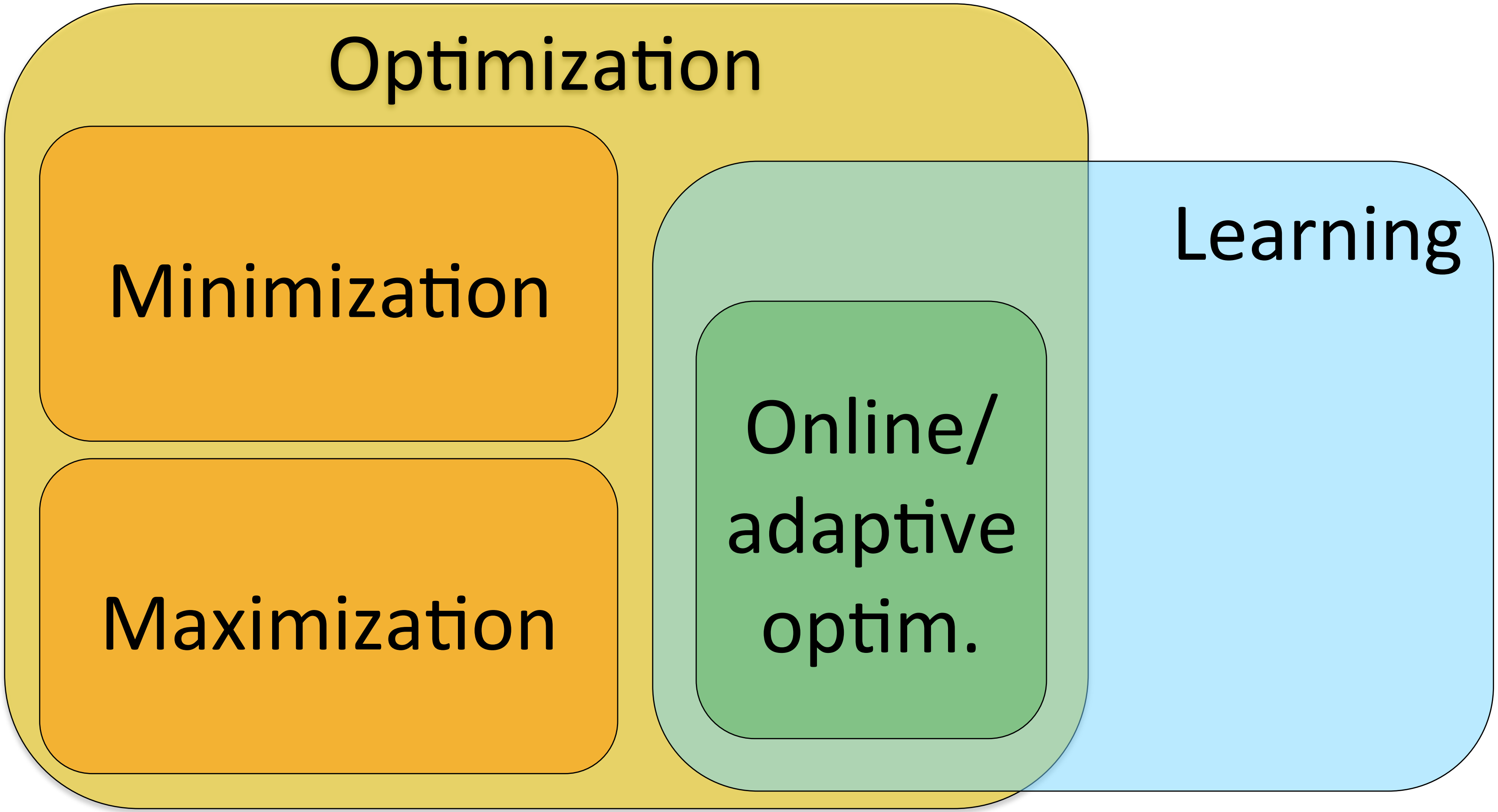
$$F(\{b\}) - F(\{\}) = 0$$
$$<$$
$$F(\{a,b\}) - F(\{a\}) = 1$$

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

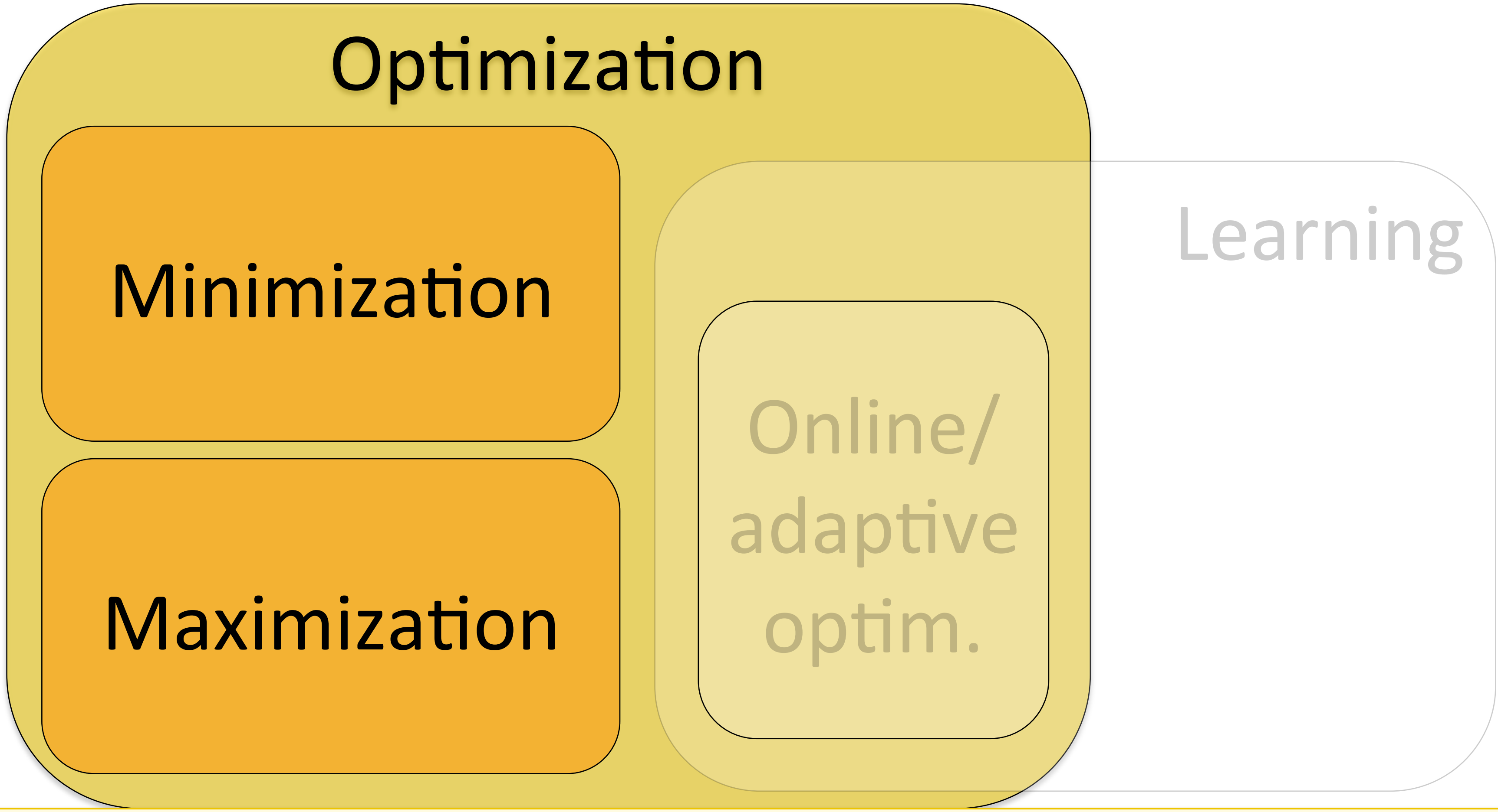
Two faces of submodular functions



What to do with submodular functions

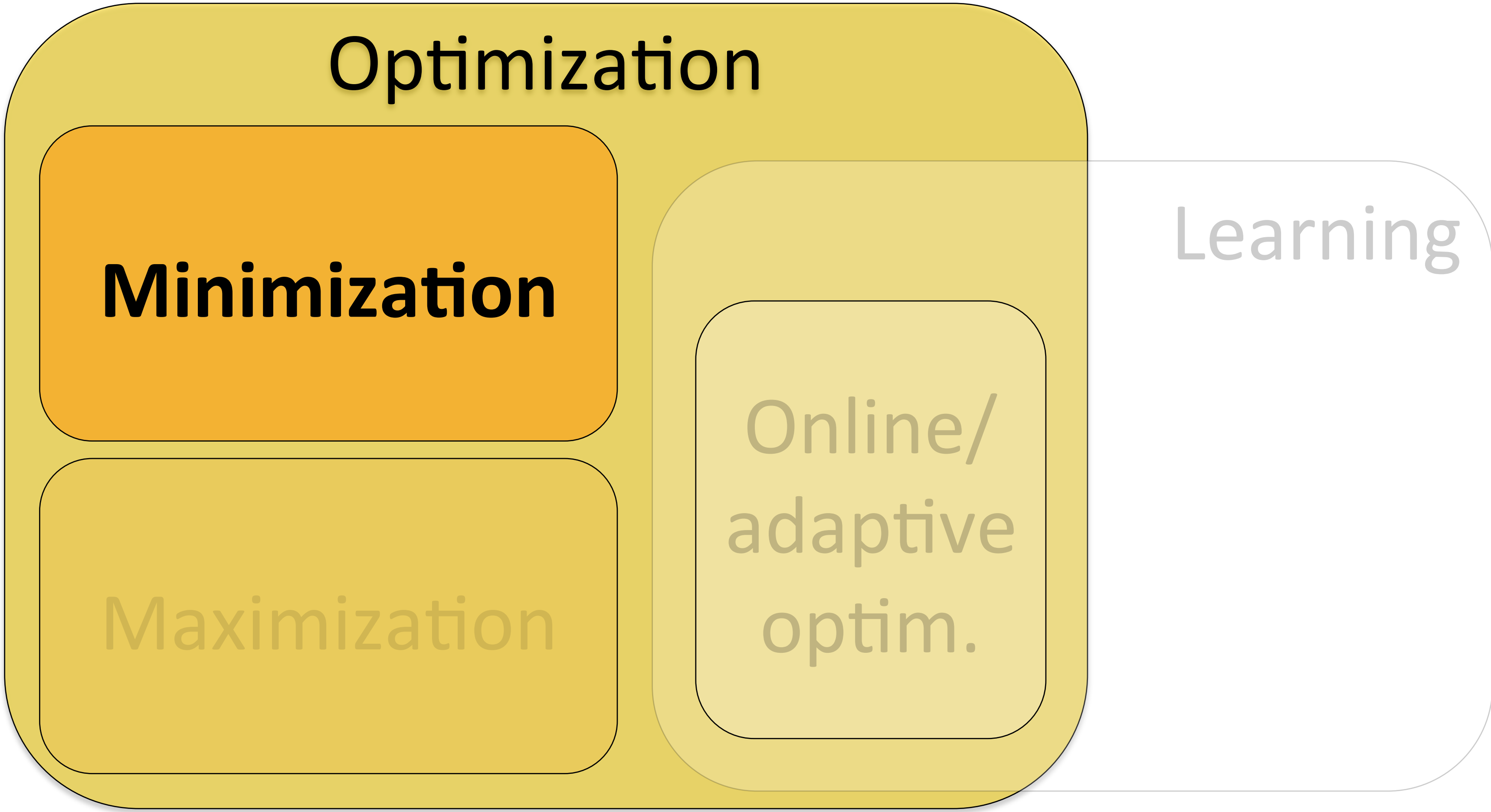


Optimization

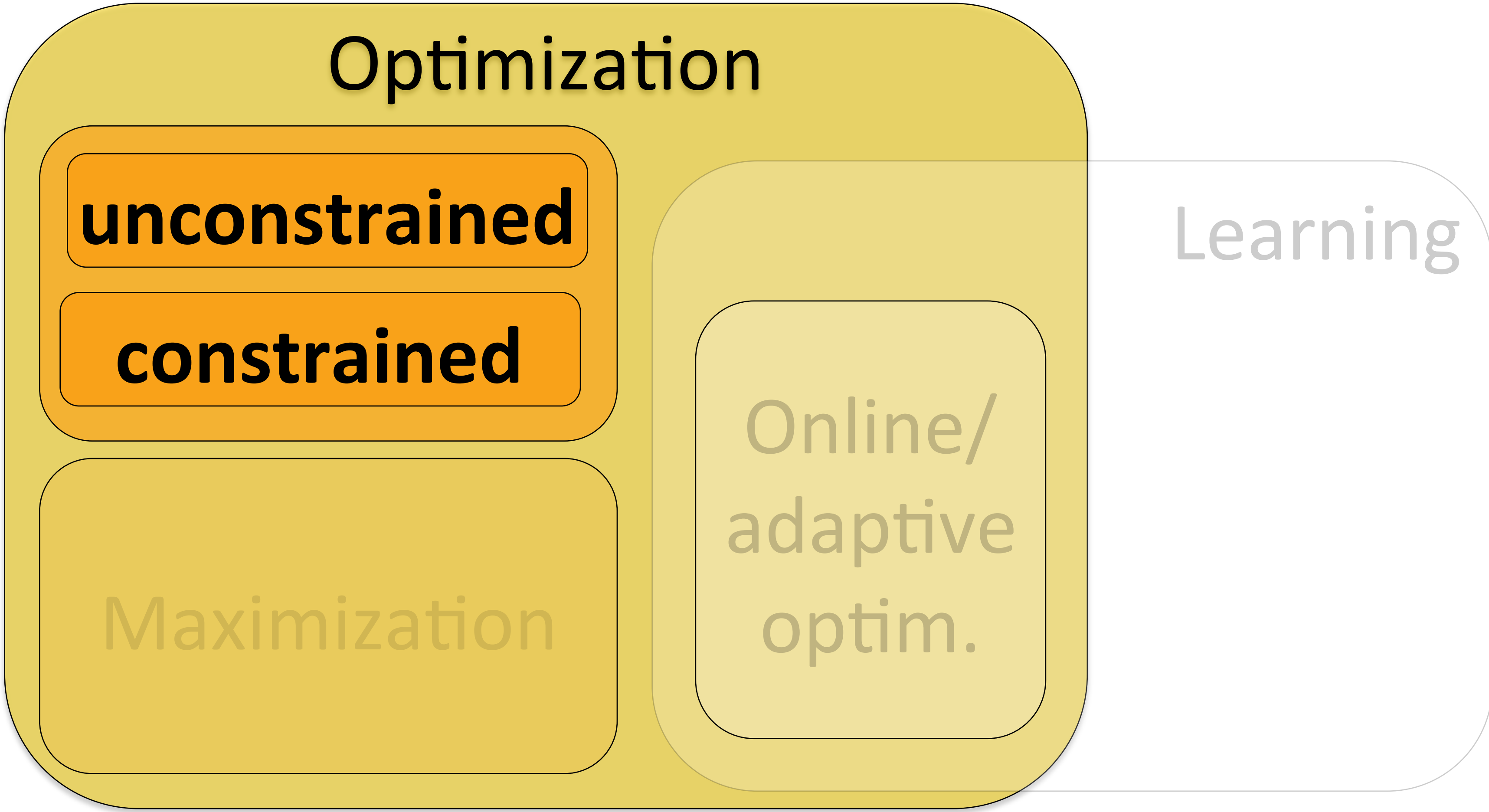


Minimization and maximization not the same??

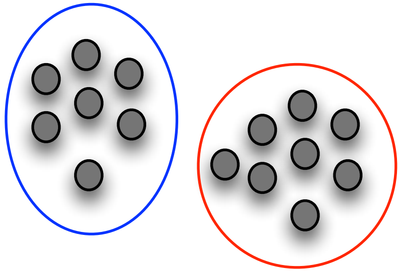
Submodular minimization



Submodular minimization

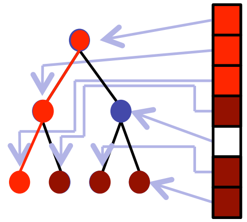


Submodular function minimization

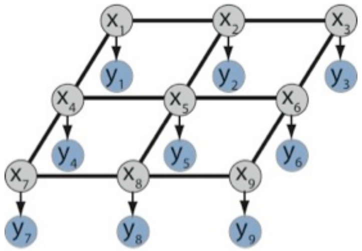


clustering

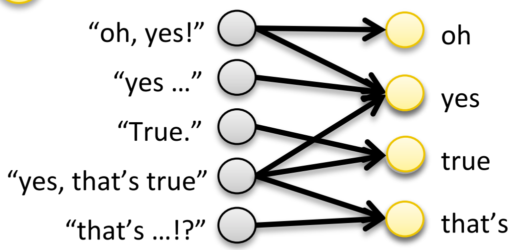
$$\min_{S \subseteq V} F(S)$$



Structured sparsity



MAP inference



Corpus training set extraction

Submodular function minimization

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

- polynomial-time?

➔ submodularity and convexity

Set functions and energy functions

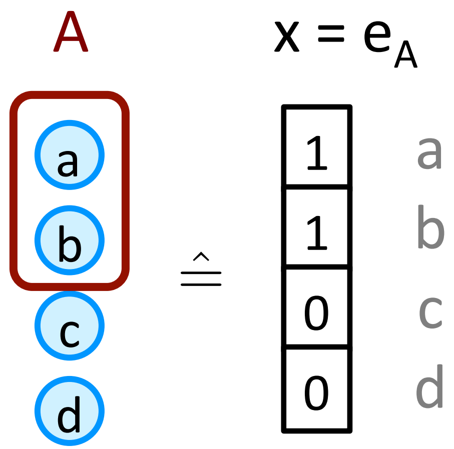
Can view any set function

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

equivalently as function on binary vectors:

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

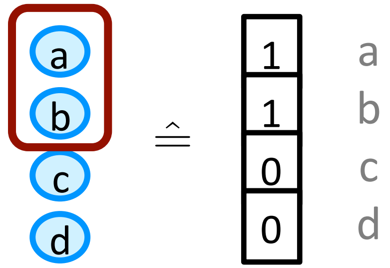
where |V|=n



Submodularity and Convexity

- Extension:

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



Lovász extension

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

convex

- minimum of *F* is a minimum of *f*

➔ submodular minimization reduces to convex min

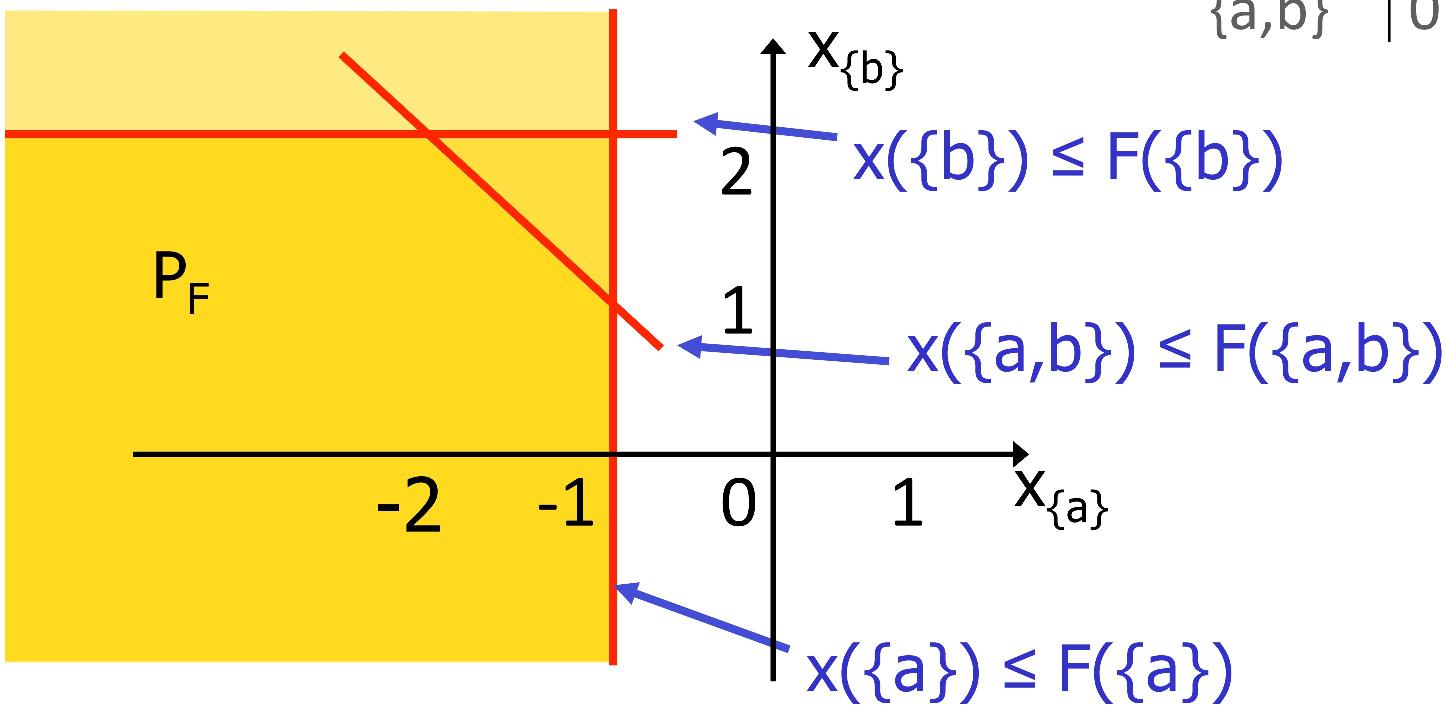
The submodular polyhedron P_F

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

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

Example: V = {a,b}

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



Evaluating the Lovász extension

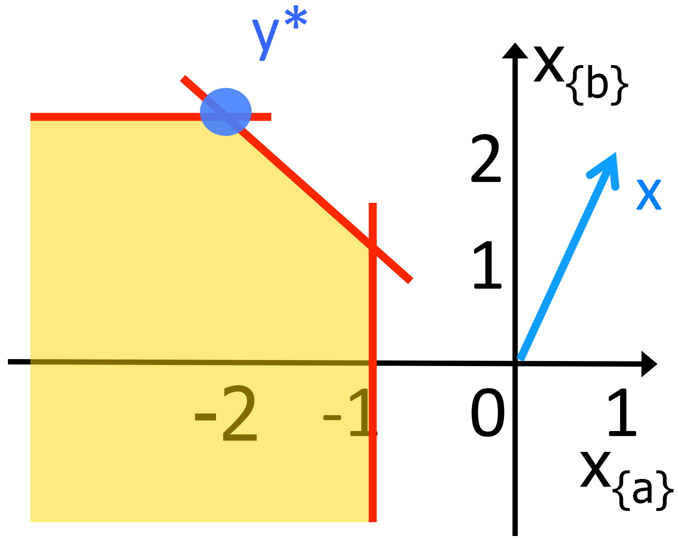
Linear maximization over P_F

$$f(x) = \max_{y \in P_f} x^\top y$$

Exponentially many constraints!!! ☹

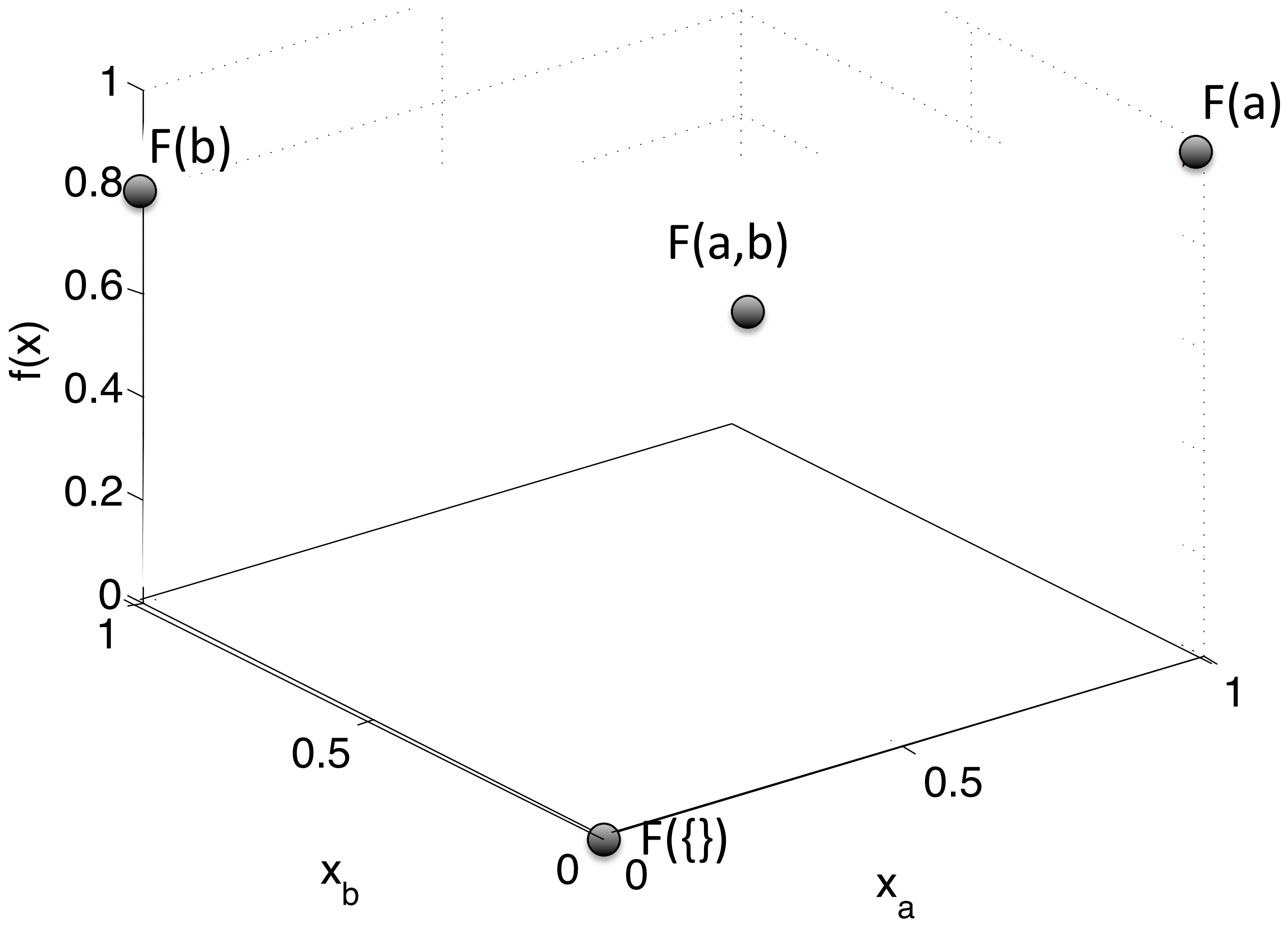
Computable in O(n log n) time ☺

[Edmonds '70]



- Subgradient
- Separation oracle
- Central for optimization

Example Lovasz extension



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

Submodular function minimization

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

- polynomial-time?
  - Yes -- ellipsoid algorithm: Grötschel, Lovasz, Schrijver `81
- Combinatorial algorithm?

Edmonds, Cunningham,...

Iwata, Fujishige, Fleischer `01

Schrijver `00

Iwata `03: $O(n^4T + n^5\log M)$

Orlin `09: $O(n^6 + n^5T)$

T = time for evaluating F

"efficient" ...?

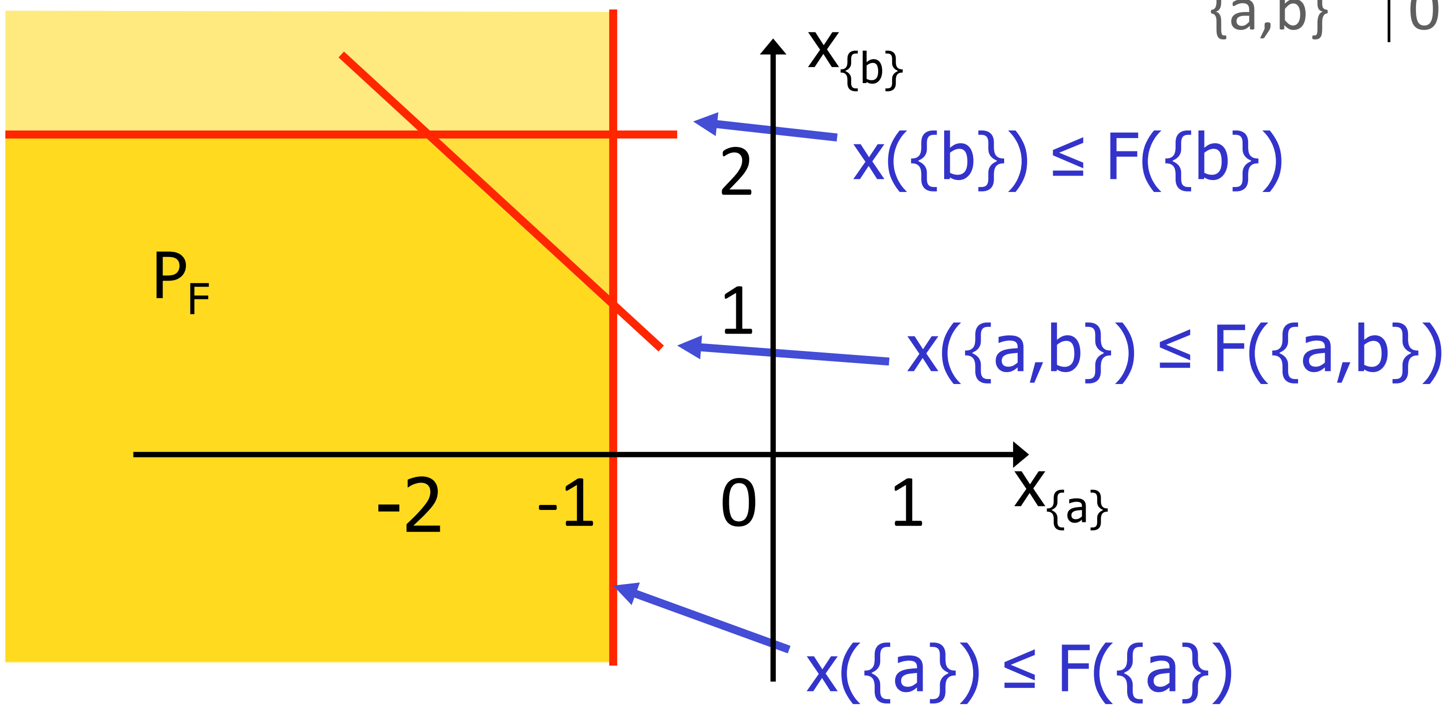
The submodular polyhedron P_F

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

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

Example: V = {a,b}

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



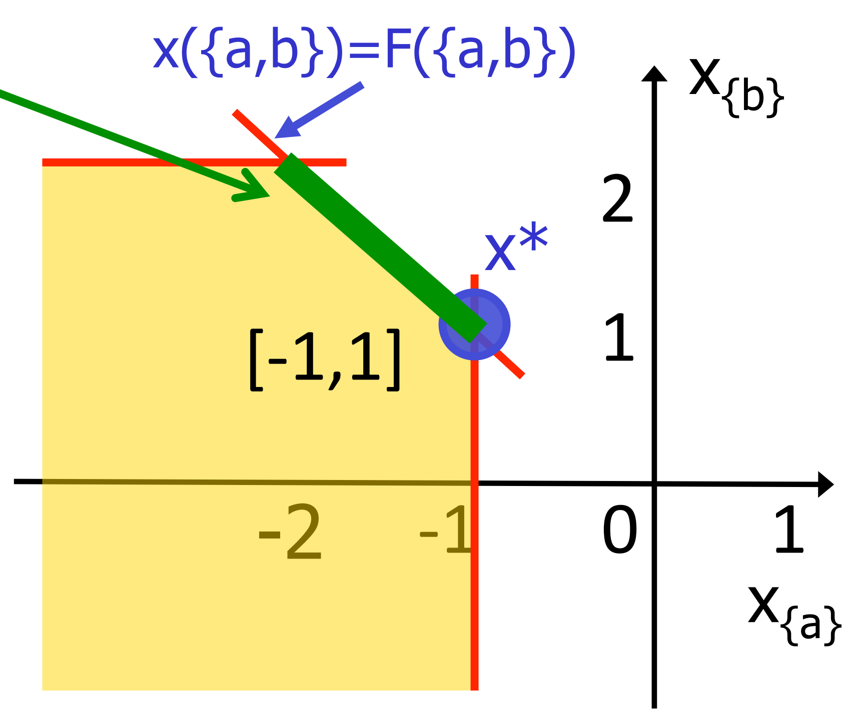
A more practical alternative? [Fujishige '91, Fujishige et al '11]

Minimum norm point algorithm

1. Find $x^* = \arg\min_{x \in B_F} \|x\|_2$

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

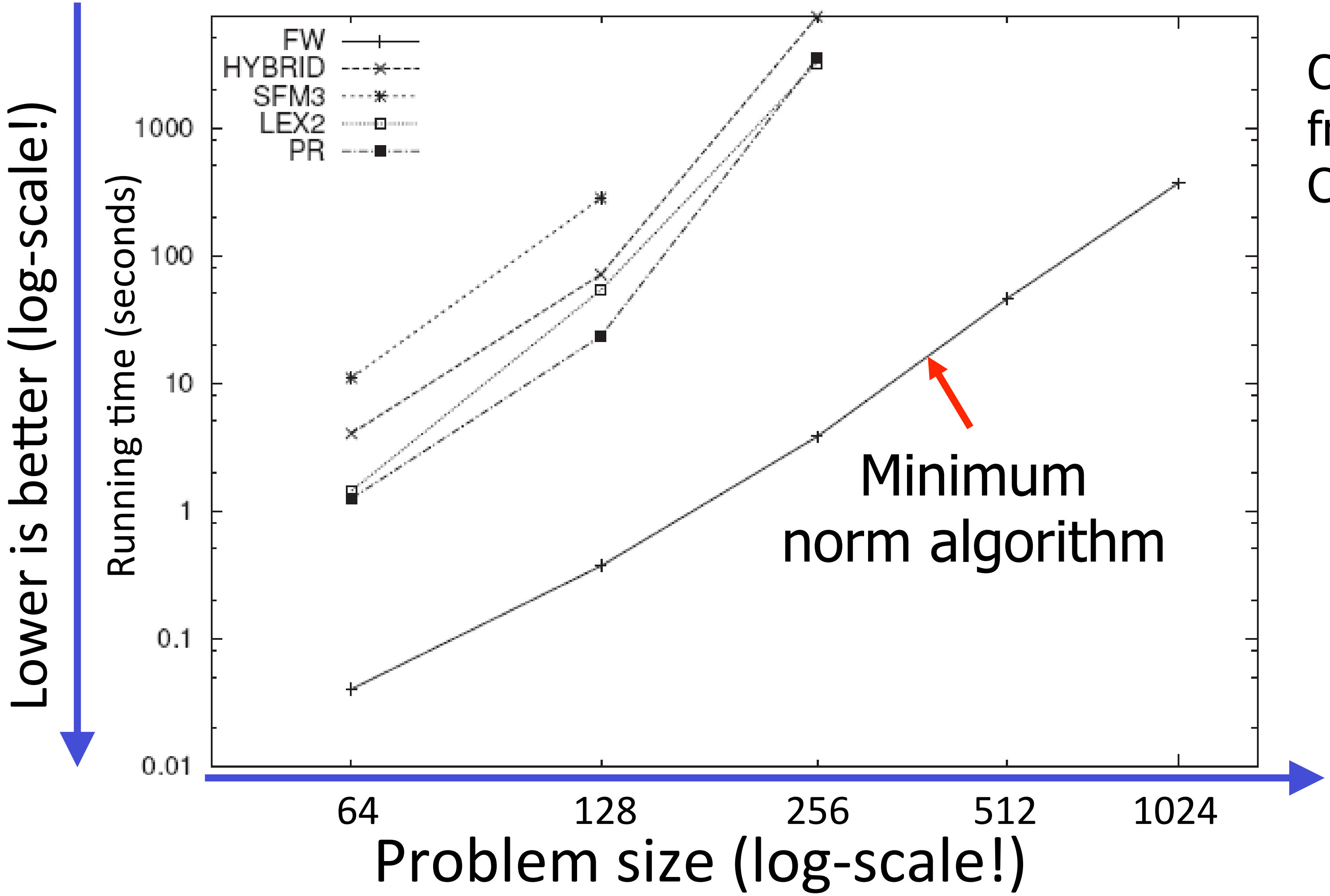
Theorem [Fujishige `91]:
A* is an optimal solution



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

Runtime finite but worst-case complexity open

Empirical comparison [Fujishige et al ’06]

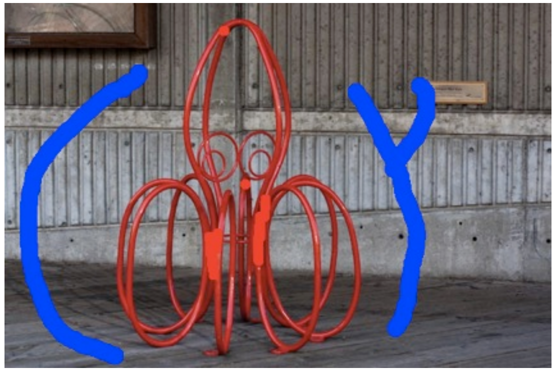


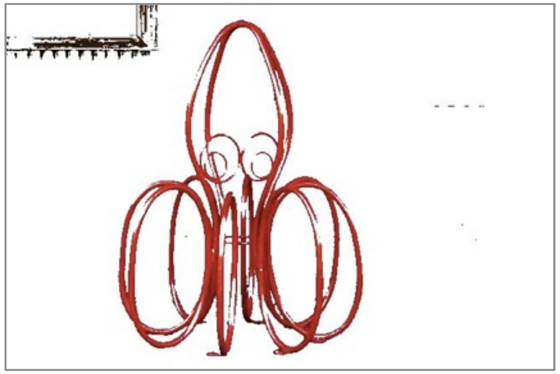
Cut functions from DIMACS Challenge

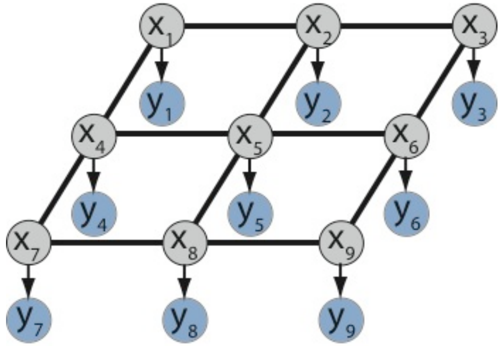
Minimum norm algorithm: usually $O(n^3)$ to $O(n^4)$

➔ orders of magnitude faster

Image segmentation







$$p(x|y) \propto \exp(-E(x;y))$$

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

color, texture, …

smoothness

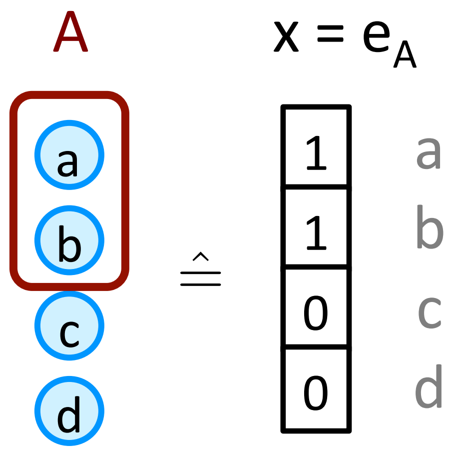
MAP inference = energy minimization

Set functions and energy functions

Can view any set function

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

equivalently as function on binary vectors:



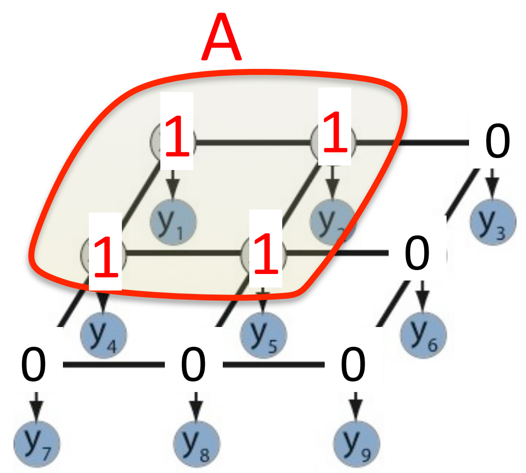
$$F : \{0,1\}^n \to \mathbb{R}$$

where |V|=n

Conversely:
a function on binary variables is a set function!

Set functions & probabilistic models

- Maximum a posteriori:
  observations y: infer binary labels x



maximize

$$p(x \mid y) \propto \exp(-E(x;\, y))$$

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

if F is submodular:
polynomial-time ☺

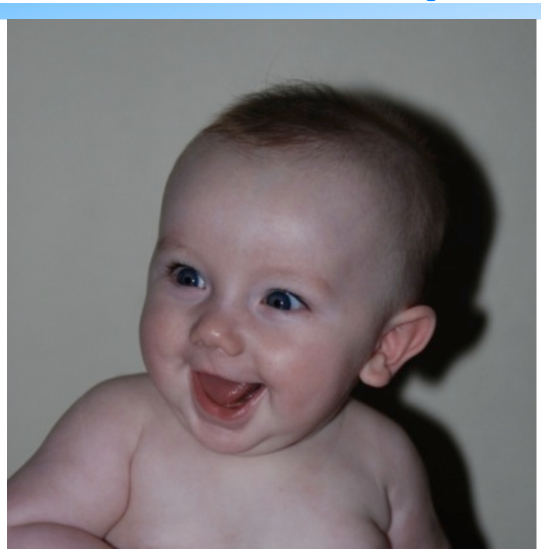
$\Leftrightarrow$ minimize

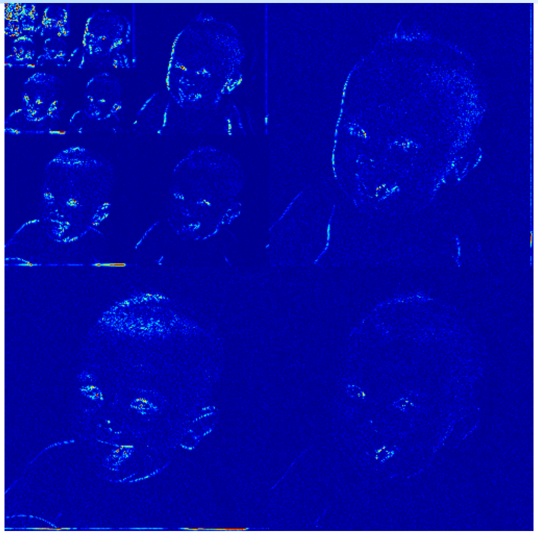
$$F(A) := E(e_A;\, y)$$

Example: Sparsity

d

pixels





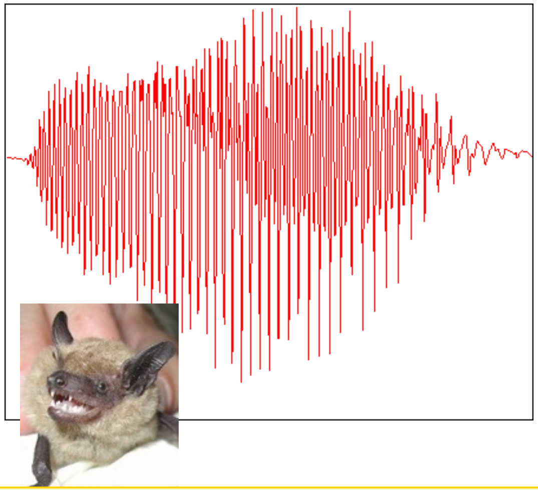
$k \ll d$

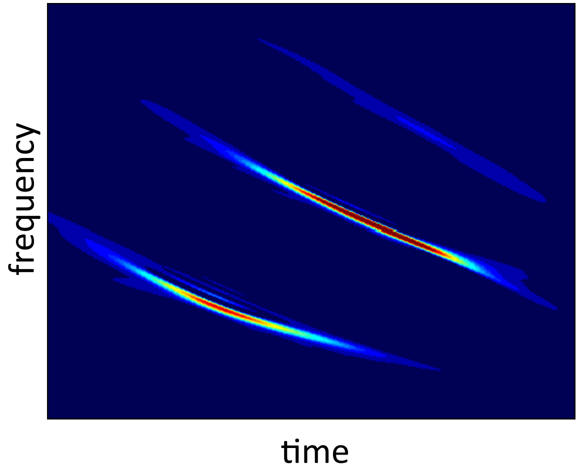
large wavelet coefficients

(blue = 0)

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

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

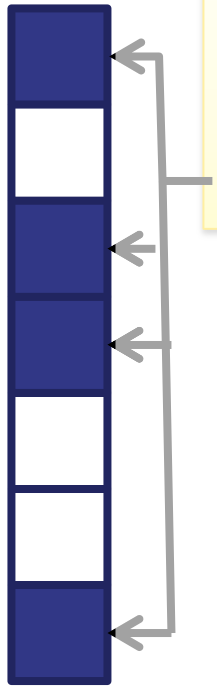
- explain y with few columns of M: few x_i

discrete regularization on support S of x

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

relax to convex envelope

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

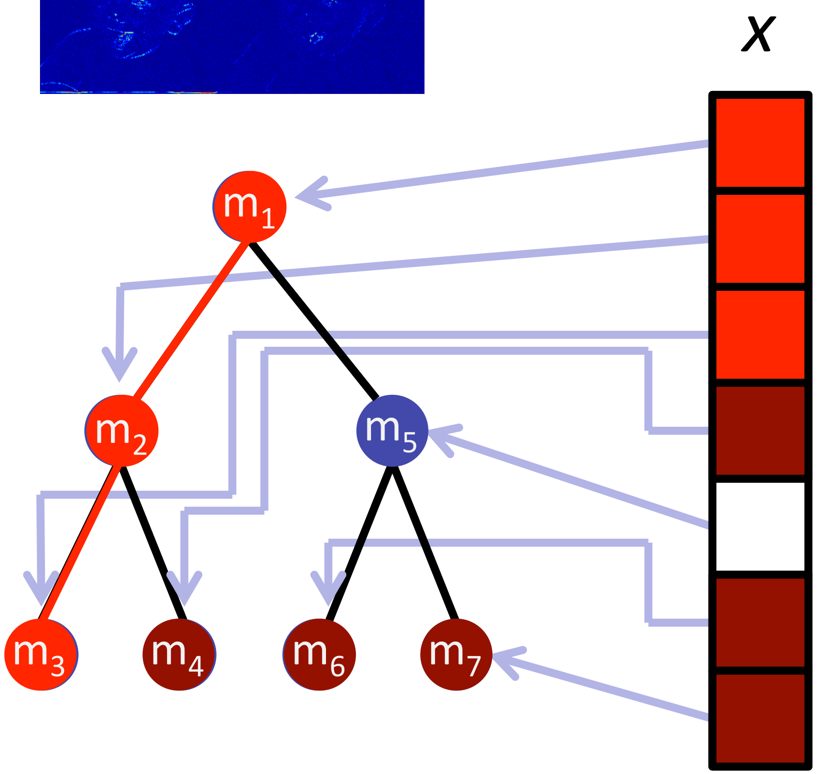


subset selection: $S = \{1,3,4,7\}$

in nature: sparsity pattern often not random…

Structured sparsity

Incorporate tree preference in regularizer?



Set function:

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

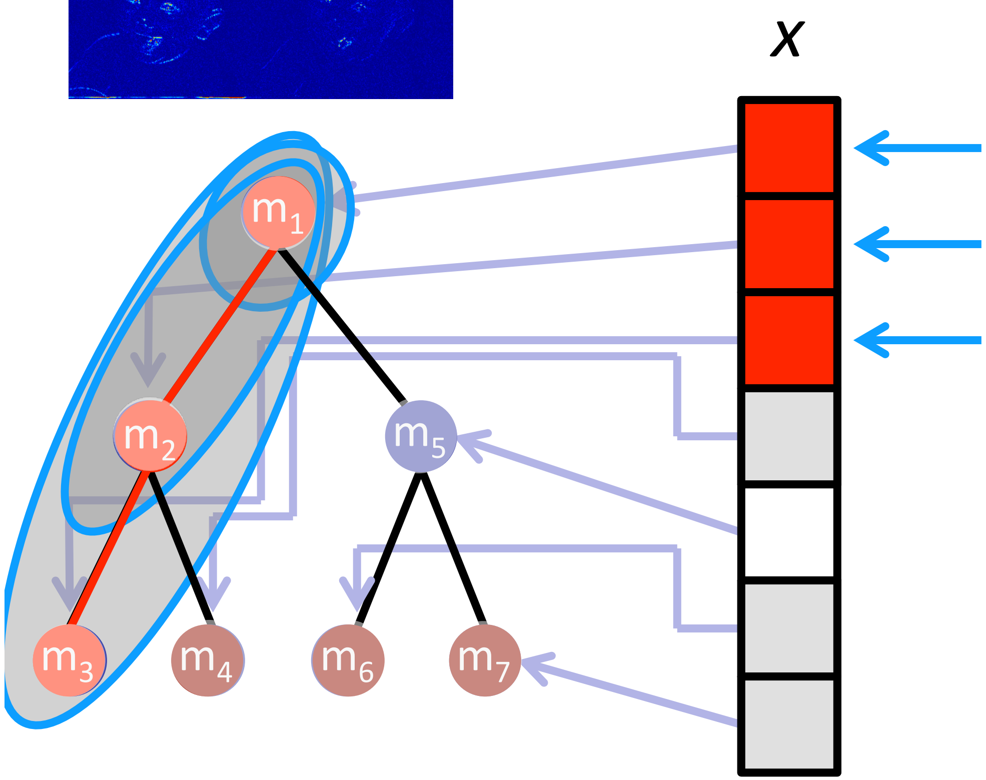
if T is a tree and S not

$|S| = |T|$

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

Structured sparsity

Incorporate tree preference in regularizer?



Set function:

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

If T is a tree and S not,

$|S| = |T|$

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

$$F(T) = 3$$

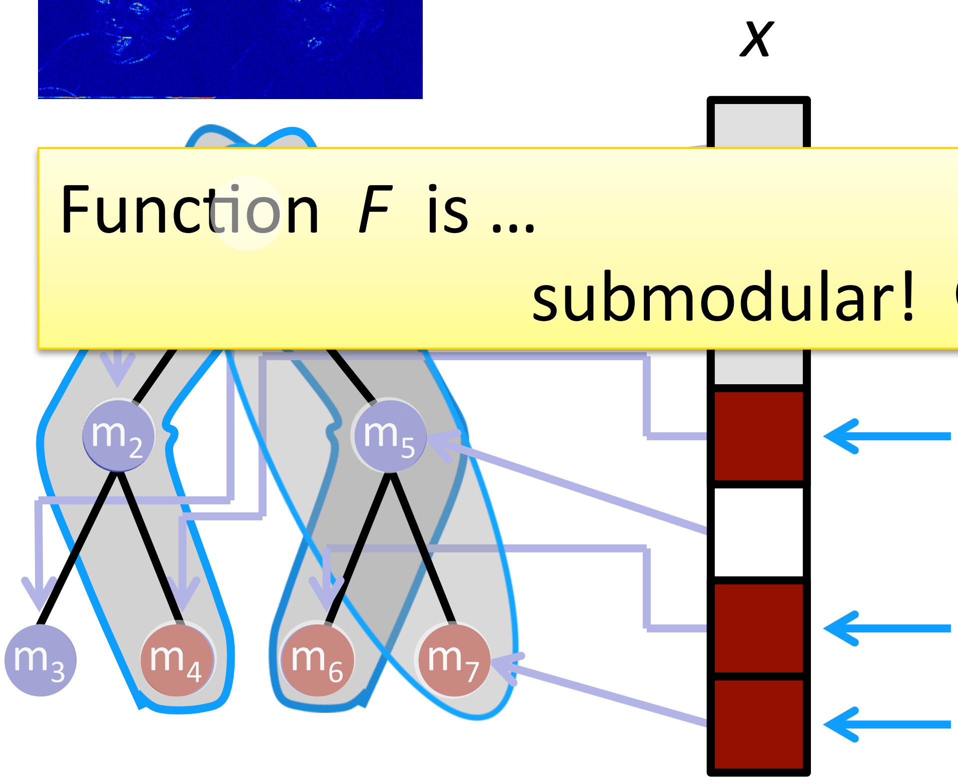
Structured sparsity

Incorporate tree preference in regularizer?

Set function:

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

If T is a tree and S not, $|S| = |T|$



Function F is ...

submodular! ☺

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

$$F(T) = 3$$

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

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

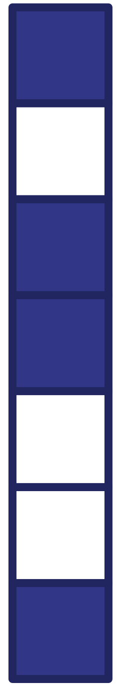
- Optimization: submodular minimization

[Bach`10]

Further connections: Dictionary Selection

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

Where does the dictionary M come from?

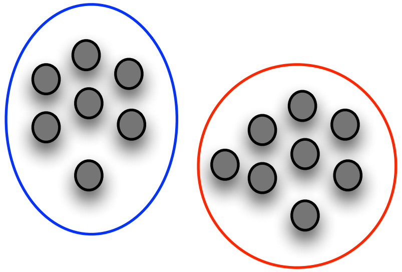


Want to learn it from data: $\{y_1, \ldots, y_n\} \subseteq \mathbb{R}^d$

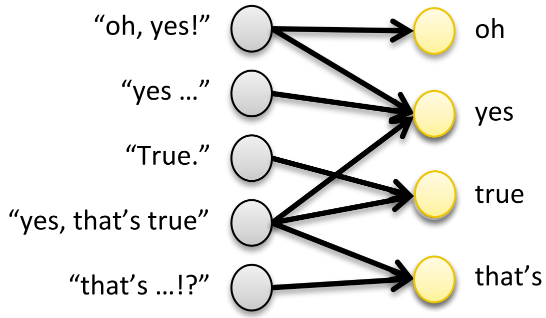
Selecting a dictionary with near-max. variance reduction
$\Leftrightarrow$ Maximization of approximately submodular function

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

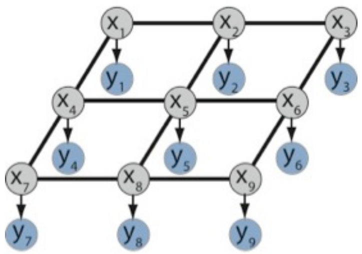
More applications …



clustering



Corpus
training set
extraction



MAP
inference

…

Special cases

Minimizing general submodular functions:

poly-time, but not very scalable

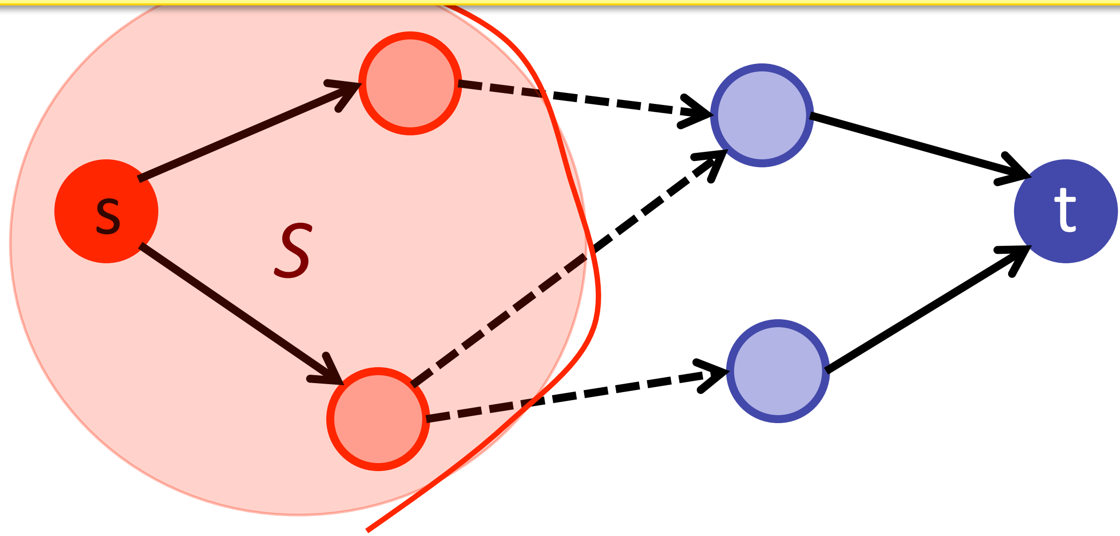
Special structure → faster algorithms

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

Graph cuts

$$\mathrm{Cut}(S) = \sum_{u \in S, v \notin S} w(u, v)$$

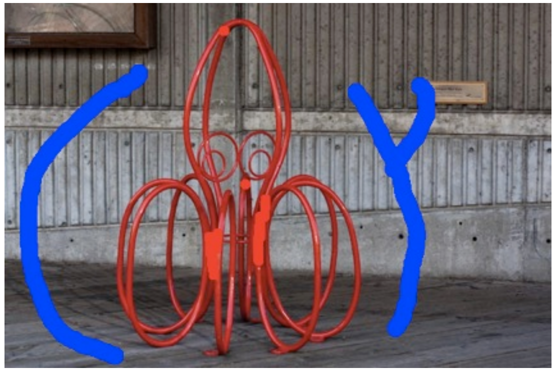
Given *F*, can we build a graph so that *F(S) = Cut(S)*?

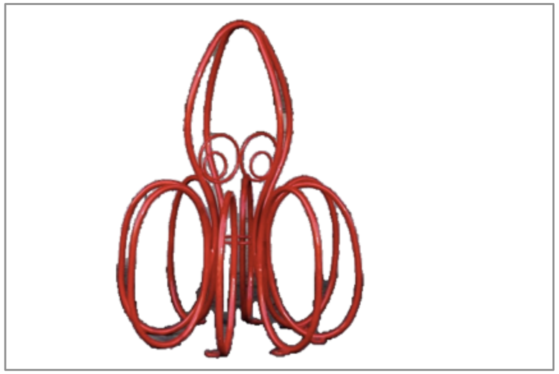


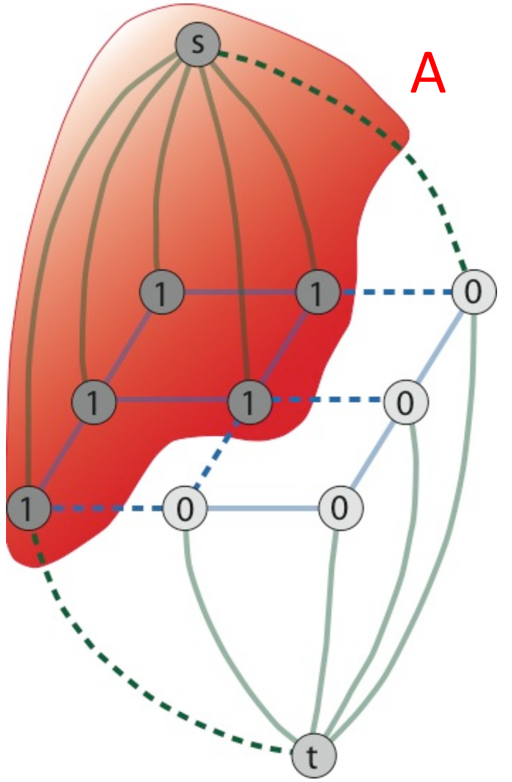
Solving a Min-*(s,t)*-cut: efficient

general case: O(mn)
[Orlin`12]
special cases: linear-time

MAP inference as graph cut







minimize

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

$$\min_A F(A) \equiv \min_{x \in \{0,1\}^n} E(x;y)$$

$$\mathrm{Cut}(A) = E(e_A;\ y)$$

Minimum energy = minimum cut! O(n)

[Boykov&Kolmogorov `04]

Which functions can be minimized as cuts?

- Functions of order two:

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

Need regularity:

$$E_{ij}(0,0) + E_{ij}(1,1) \leq E_{ij}(1,0) + E_{ij}(0,1)$$

➔ Each pairwise term must be submodular

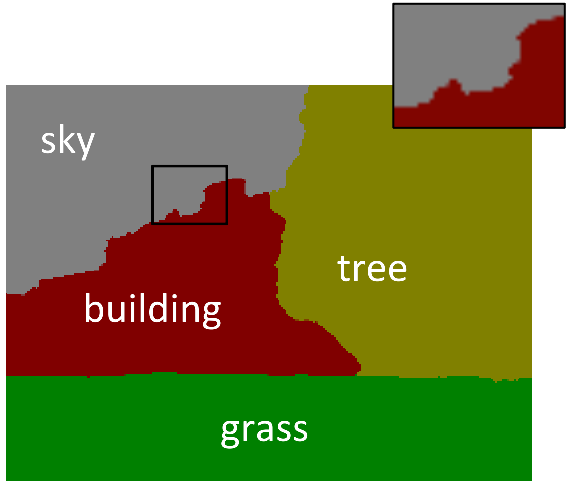
[Queyranne, Picard&Ratliff,...]

But …

Unary preferences
(texture)

+

Pairwise potentials



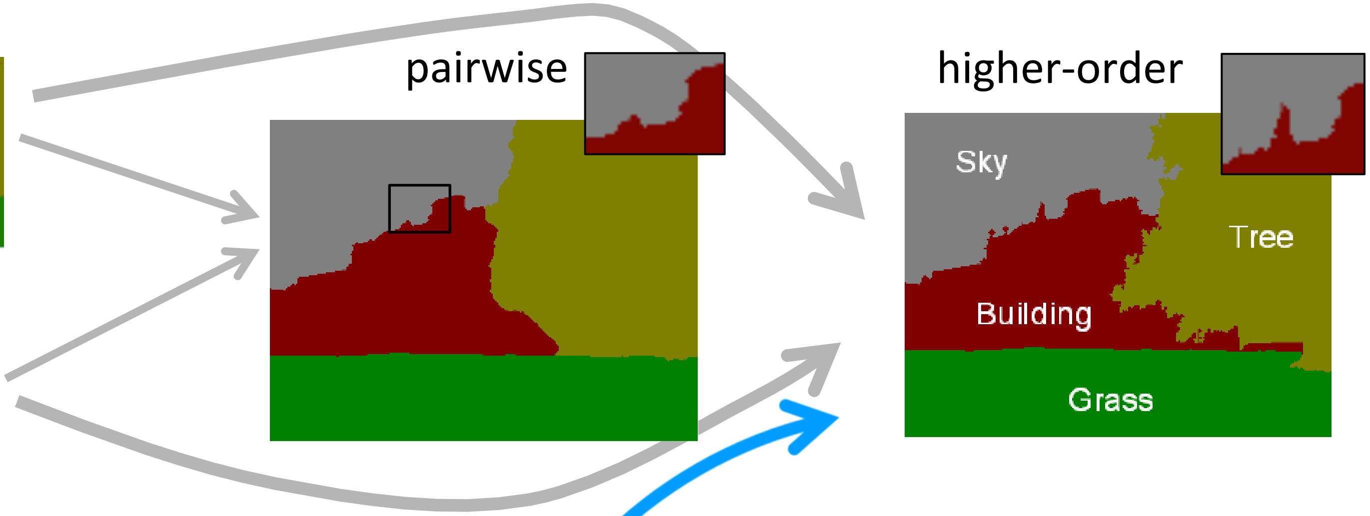
$$E(x) = \sum_i E_i(x_i) + \sum_{ij} E_{ij}(x_i, x_j)$$

P^n potentials

+

pairwise
smoothness
Potentials

+



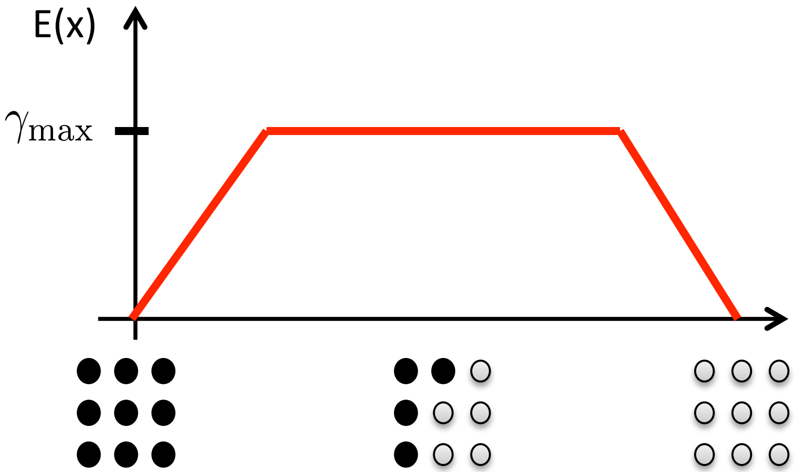
Pixels in one tile should take the same label

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

[Kohli et al.`09]

Enforcing label consistency

Pixels in a superpixel should have the same label



concave function of cardinality ➔ submodular ☺

> 2 arguments: Graph cut ??

Higher-order functions as graph cuts?

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

- 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]

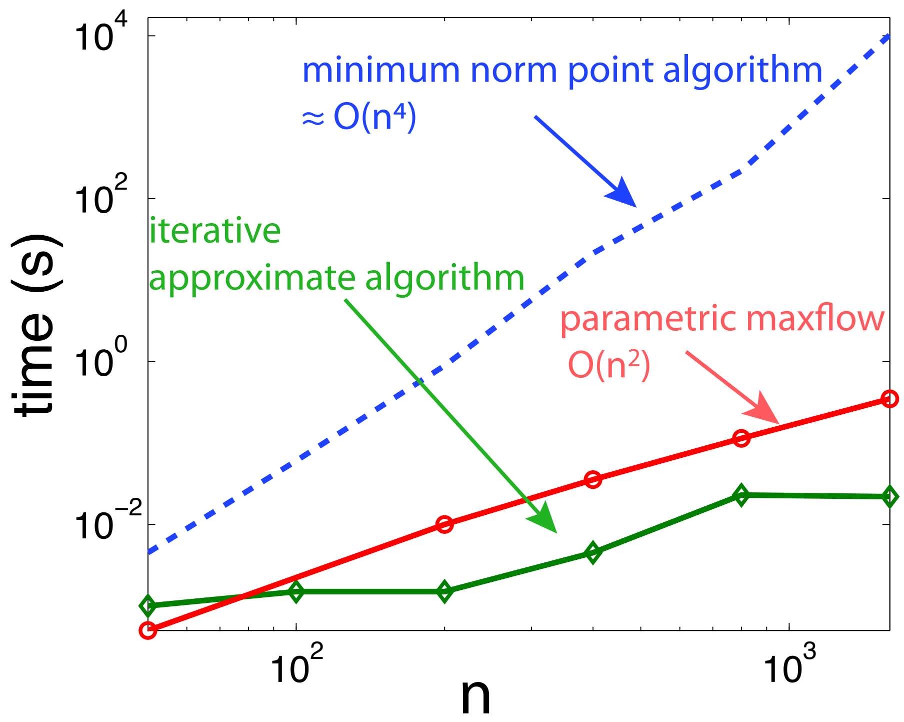
General strategy:
reduce to pairwise case by adding auxiliary variables
⚠ possibly many extra nodes in the graph

Fast approximate minimization

- Not all submodular functions are graph cuts
- Avoid adding too many extra nodes

- parametric maxflow
  [Fujishige & Iwata`99]

- approximate by a series of graph cuts
  [Jegelka, Lin & Bilmes `11]



Other special cases

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

  [Queyranne, 1998]

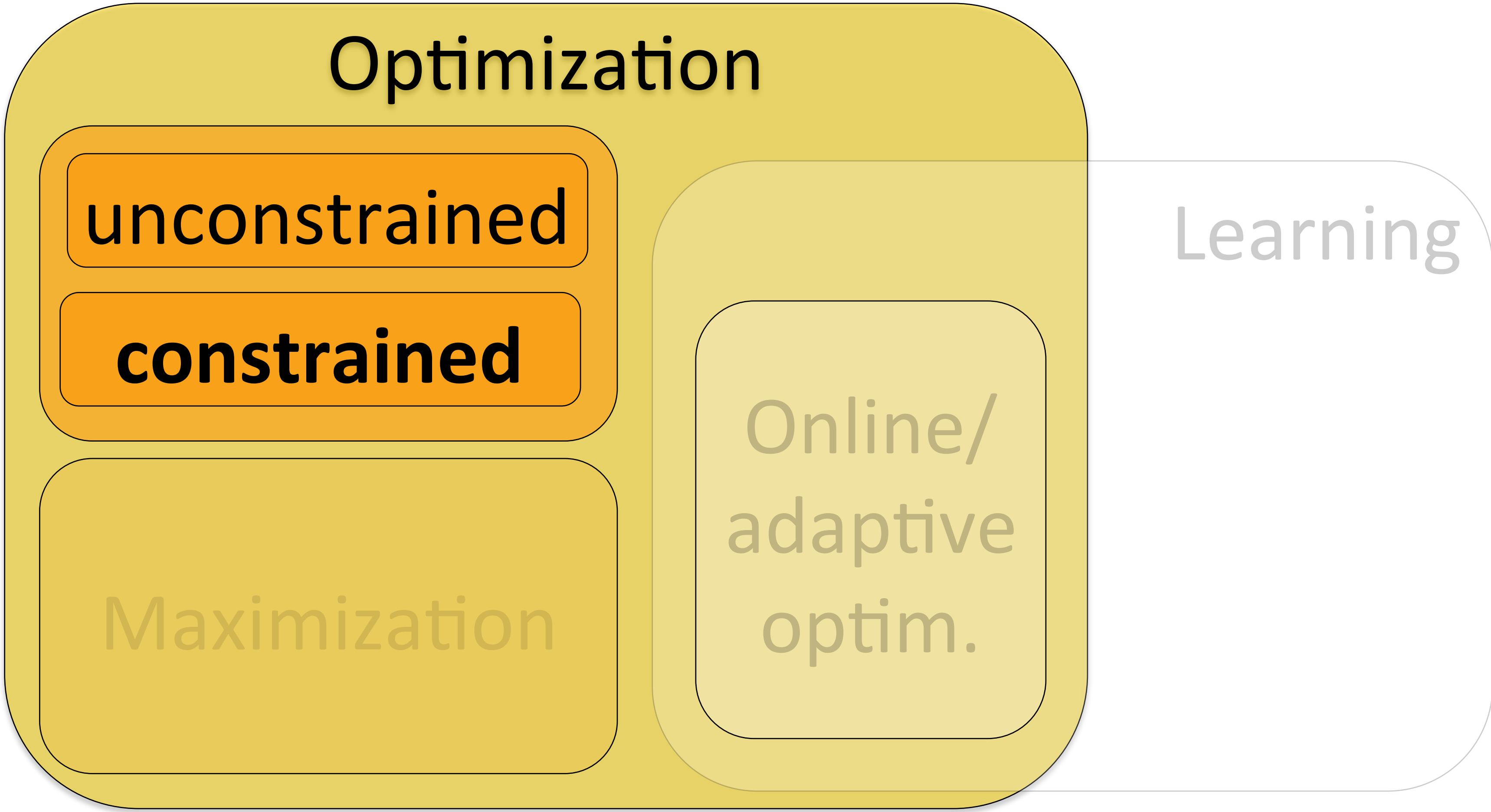
- Concave of modular: $$F(S) = \sum_i g_i\Big(\sum_{s \in S} w(s)\Big)$$

  [Kolmogorov `12, Stobbe & Krause `10, Kohli et al, `09]

- Sum of submodular functions, each bounded support

  [Kolmogorov `12]

Submodular minimization



Submodular Minimization

- Polynomial time
- Empirically better: minimum-norm point algorithm
- Provably better: special cases (graph cuts, concave, ...)

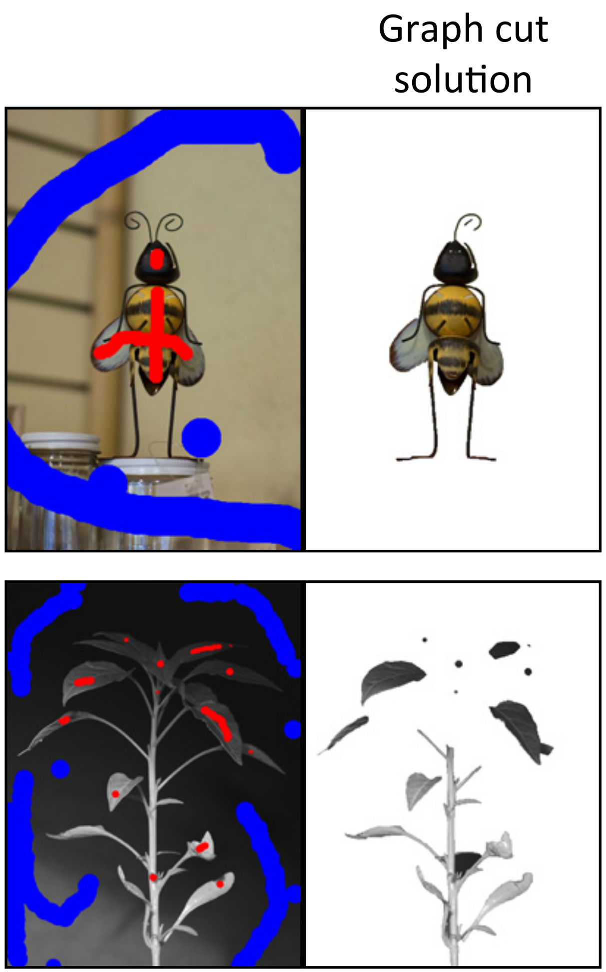
What if we have constraints?

Hint: graph cuts are submodular functions...

limited cases doable:
- odd/even cardinality
- inclusion/exclusion of a set
- ring family
- ...

General case:
NP-hard
polynomial lower bounds

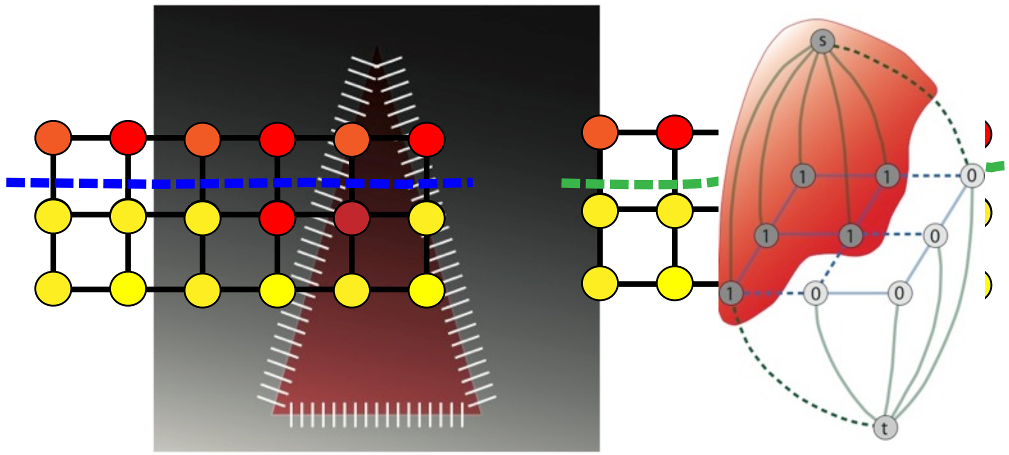
Limitations of graph cuts



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

data fit

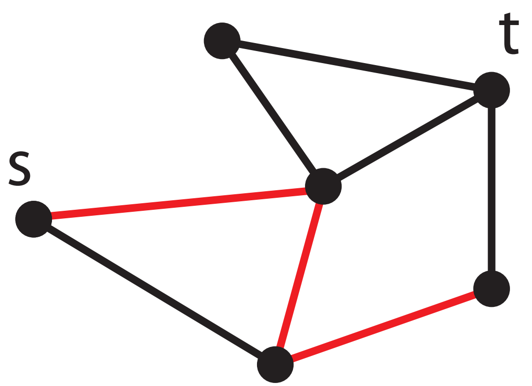
smoothness prior: cut in grid



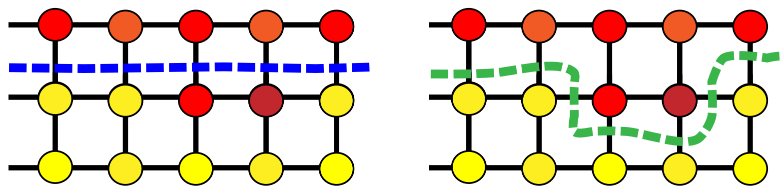
- True boundary not “short”
- Local information scarce

instead: prefer congruous boundary → look at entire cut at once

Graph cut as edge selection



now:
prefer congruous cuts



minimize sum

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

s.t. C is a cut

minimize submodular function

$$F(C)$$

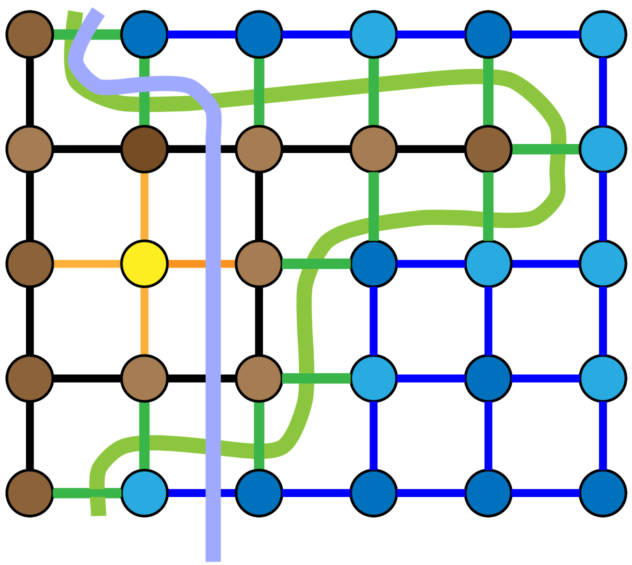
not a sum of weights!

s.t. C is a cut

Rewarding co-occurrence of edges

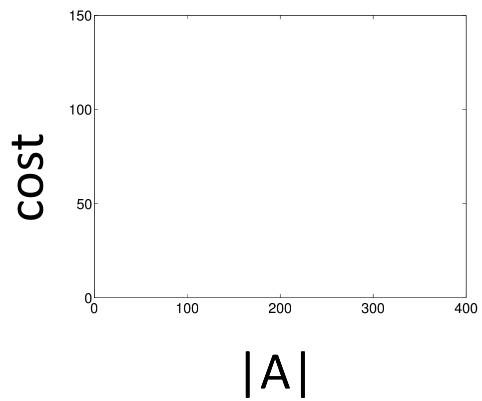
sum of weights:
use few edges

submodular cost function:
use few groups S_i of edges



One group (13 edges)
Many groups (6 edges)

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

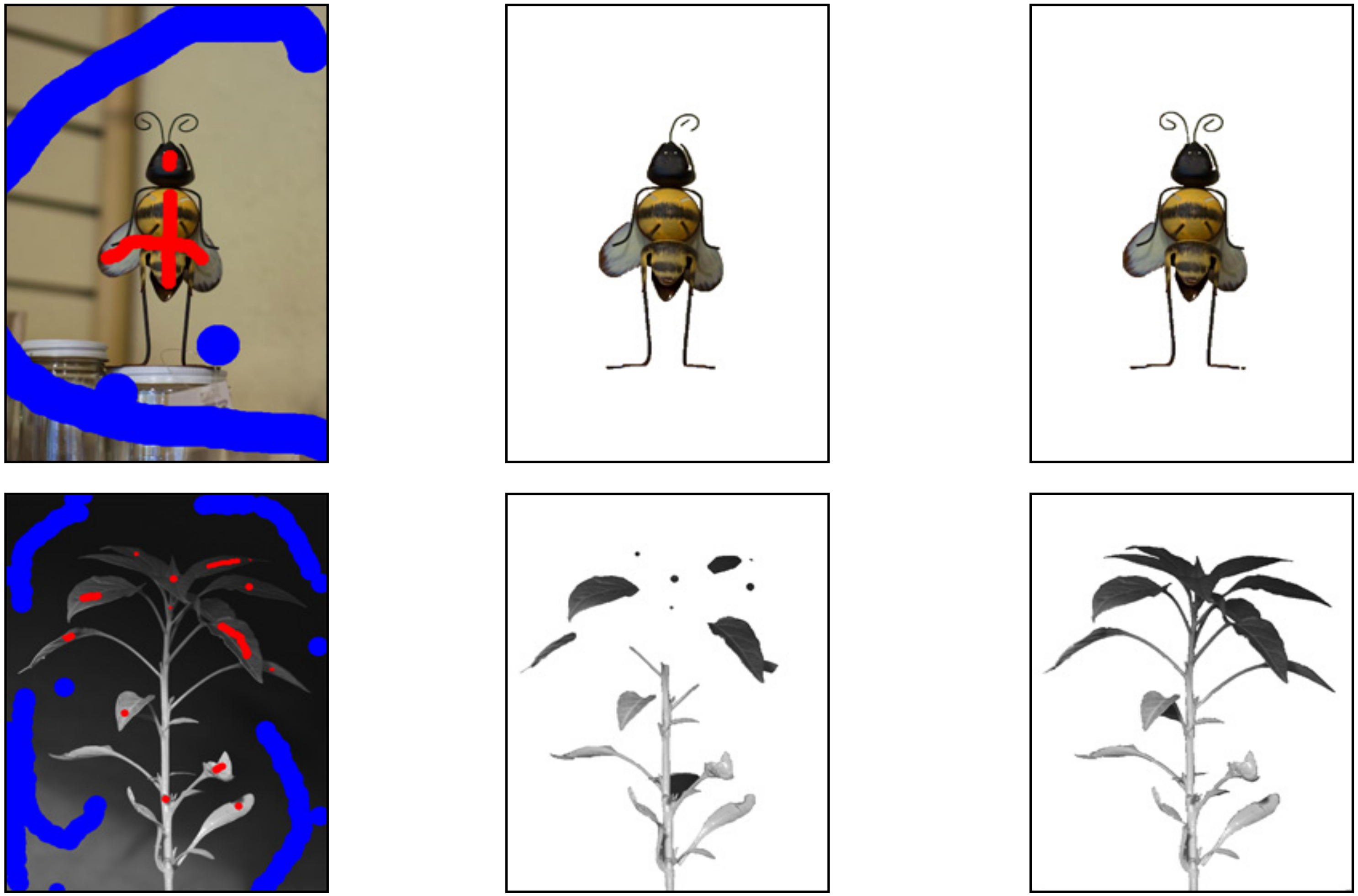


Optimization?
efficient iterative algorithm

Results

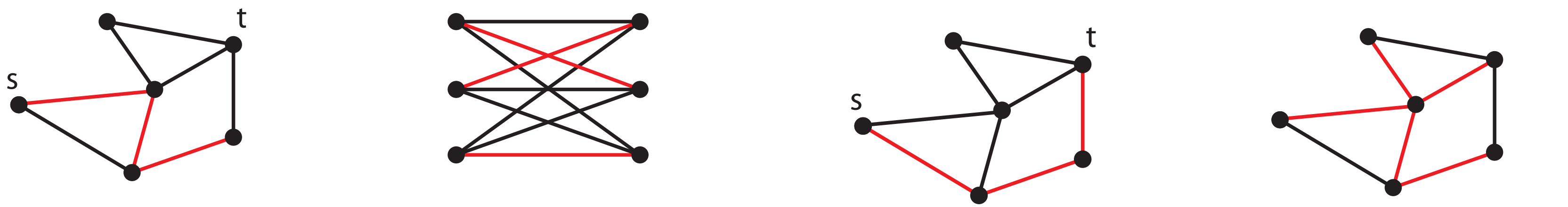
Graph cut solution

Cooperative cut solution



SFM & Combinatorial Constraints

cut | matching | path | spanning tree



... with minimum cost

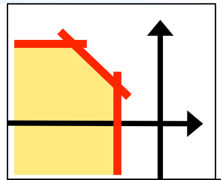
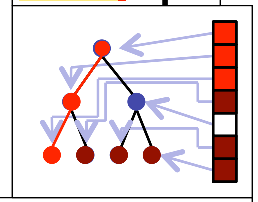
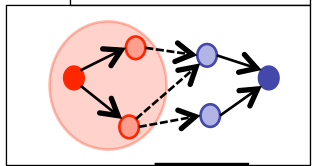
$$\min_{S \in \mathcal{C}} \sum_{e \in S} w(e) \quad \Longrightarrow \quad \min_{S \in \mathcal{C}} F(S)$$

General case: very hard. [Goel et al.`09, Iwata & Nagano `09, Jegelka & Bilmes `11,...]

Approximations:

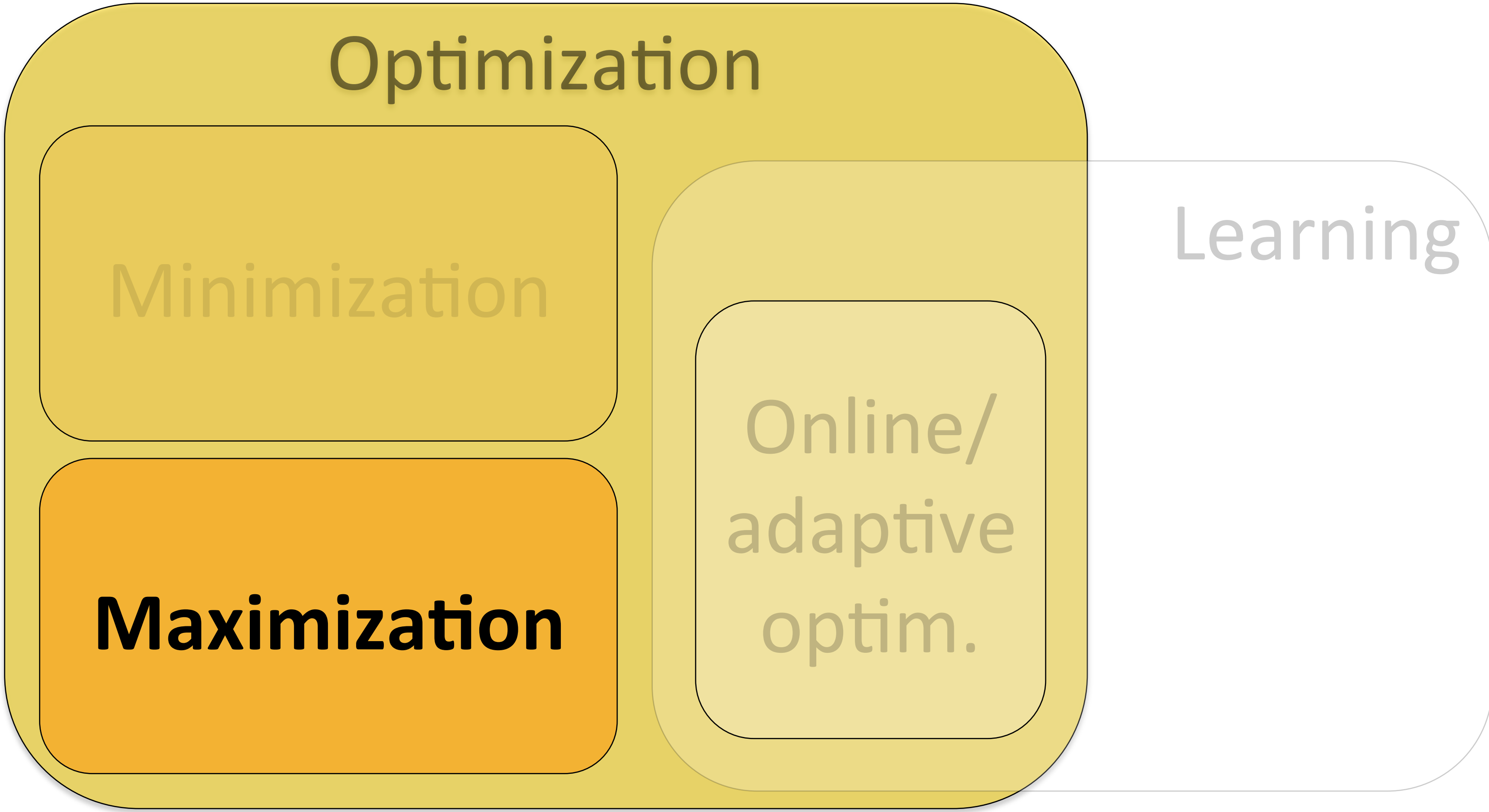
relaxation via Lovasz extension | approximate *F*

Submodular Minimization

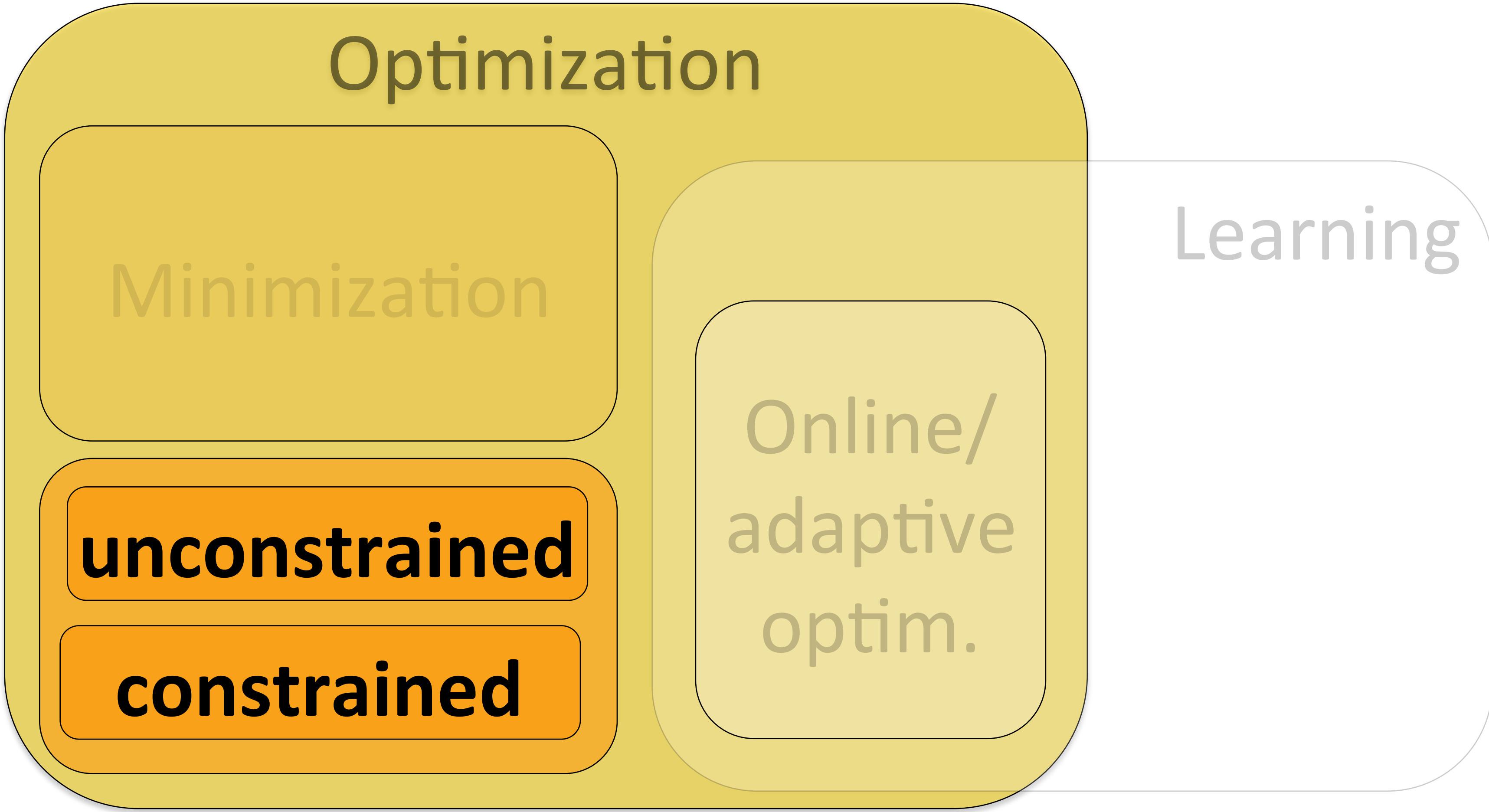





- Convex Lovász extension
- (High-order) polynomial time
- Empirically better: minimum-norm point algorithm
- Provably better: special cases
- Constraints:
  usually hard problems ➔ approximations
- Applications: MAP inference, combinatorial regularizers, clustering, …

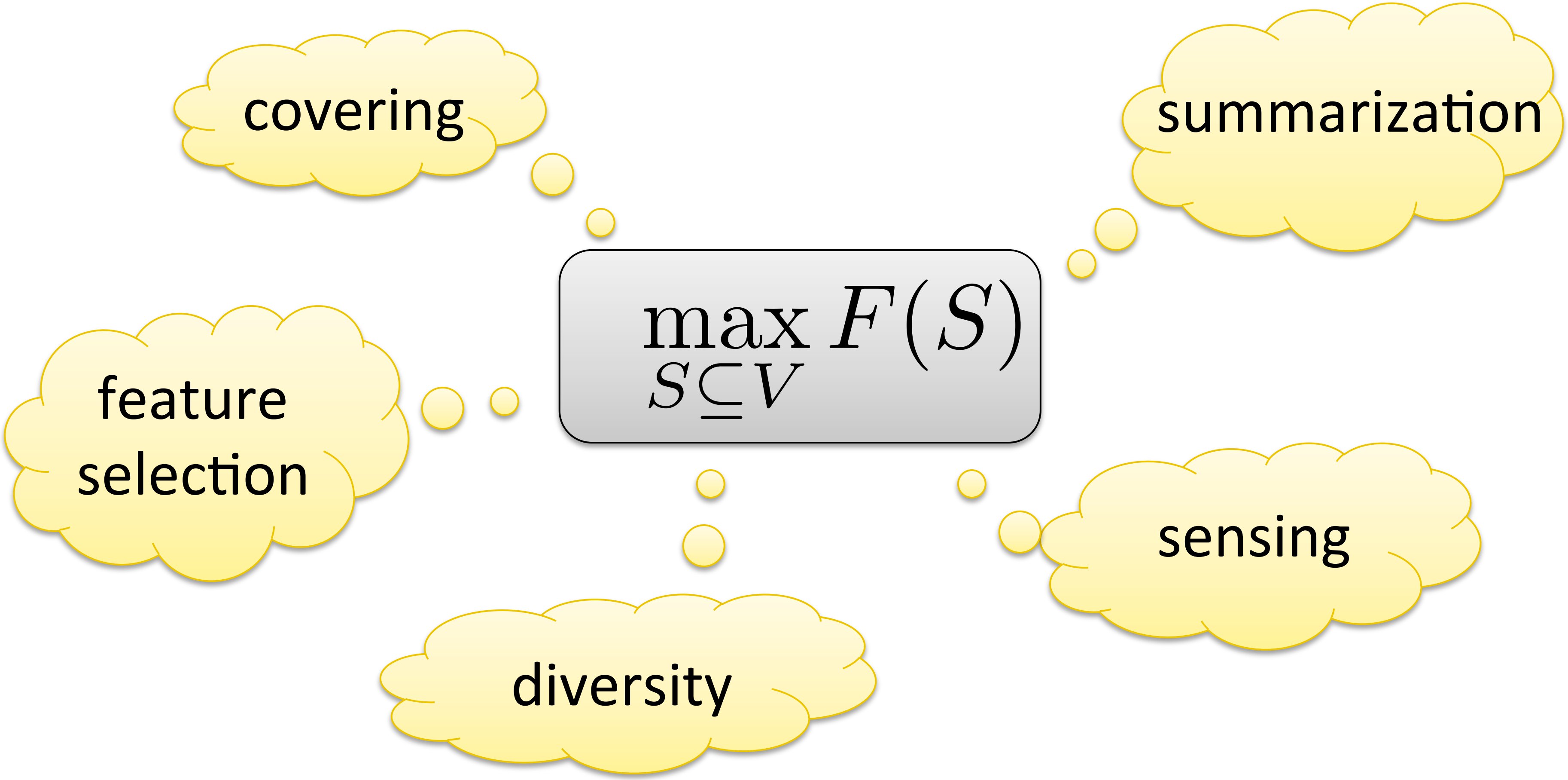
Optimization



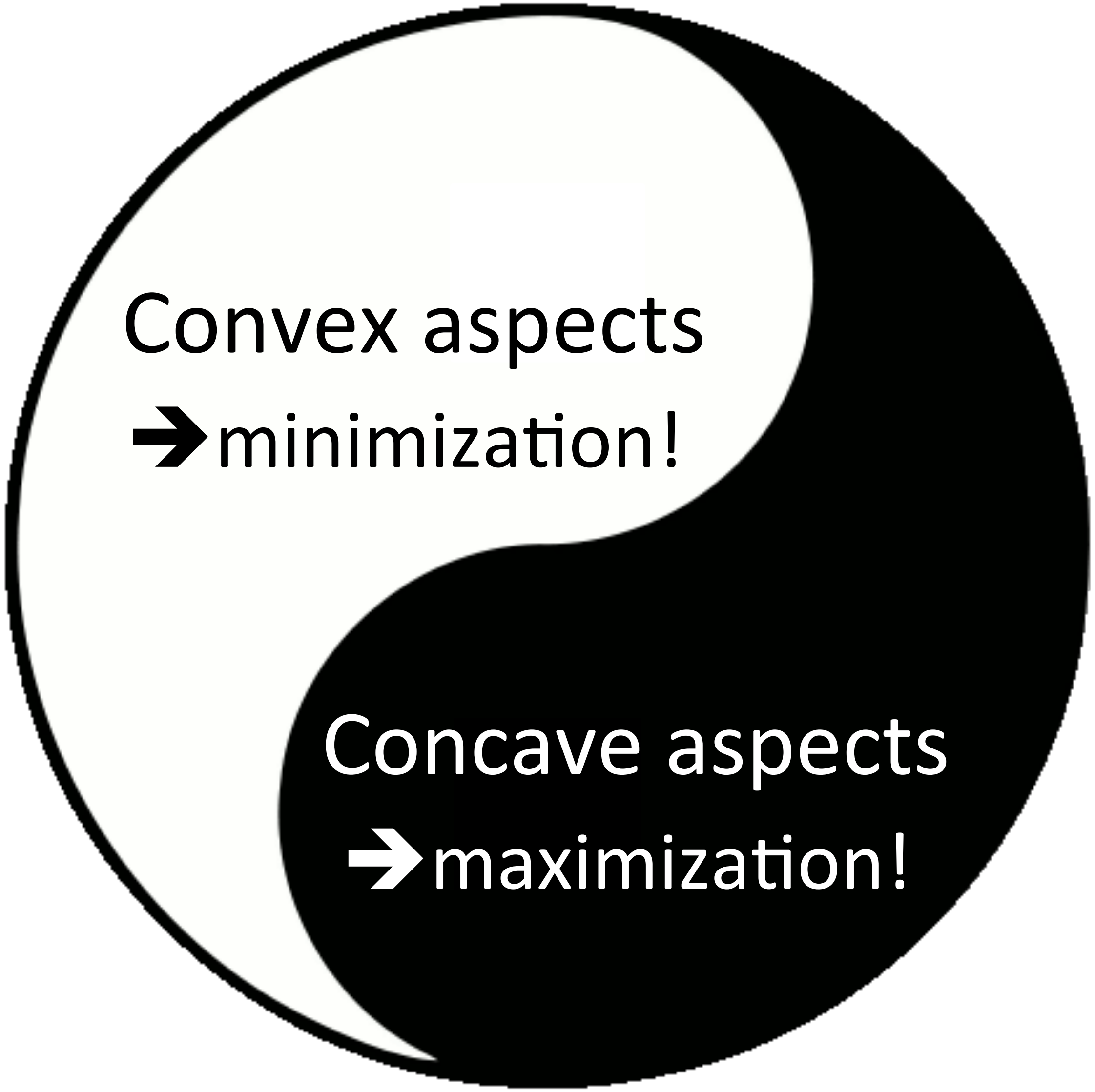
Optimization



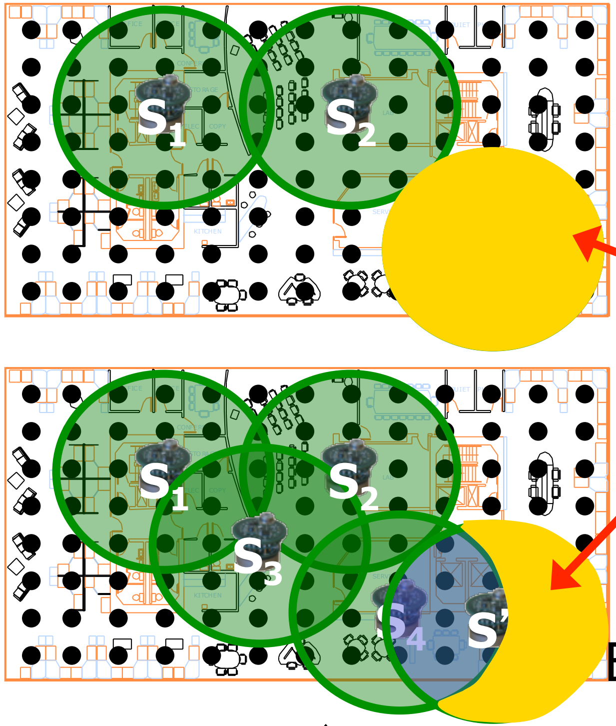
Submodular function maximization



Two faces of submodular functions



Reminder: Diminishing returns



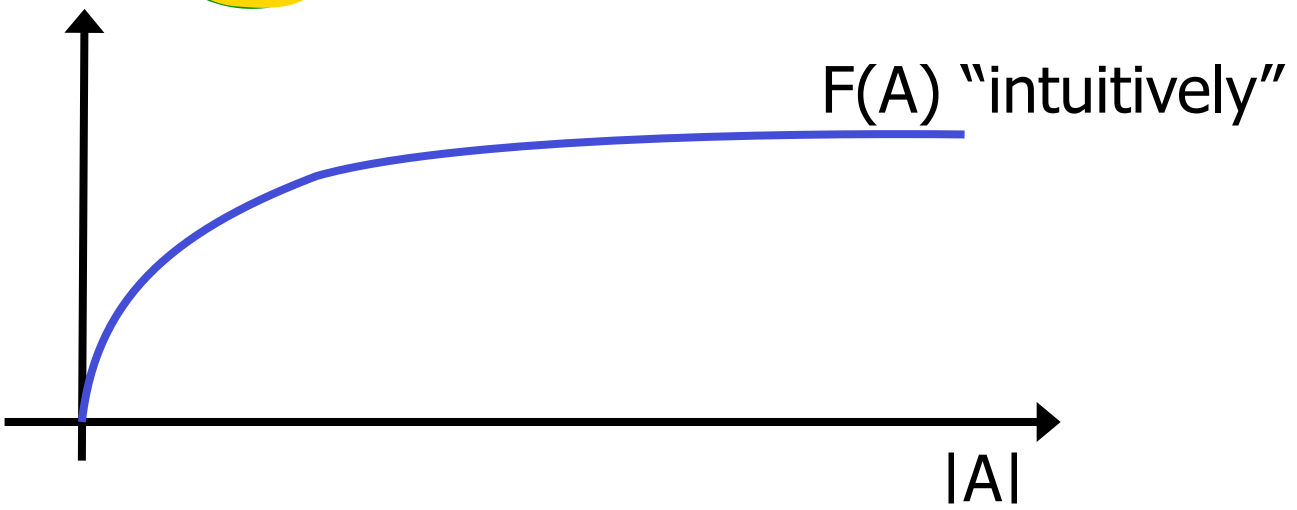
$A=\{S_1,S_2\}$

F(A U {S' })-F(A)

≥

F(B U {S' })-F(B)

$B = \{S_1,S_2,S_3,S_4\}$



Maximizing submodular functions

Minimizing convex functions:
Polynomial time solvable!

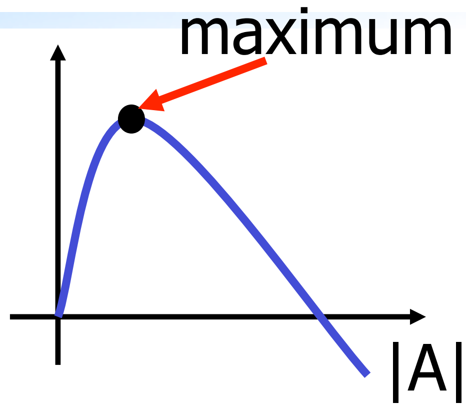
Minimizing submodular functions:
Polynomial time solvable!

Maximizing convex functions:
NP hard!

Maximizing submodular functions:
NP hard!

But can get approximation guarantees ☺

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)

Maximizing positive submodular functions

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

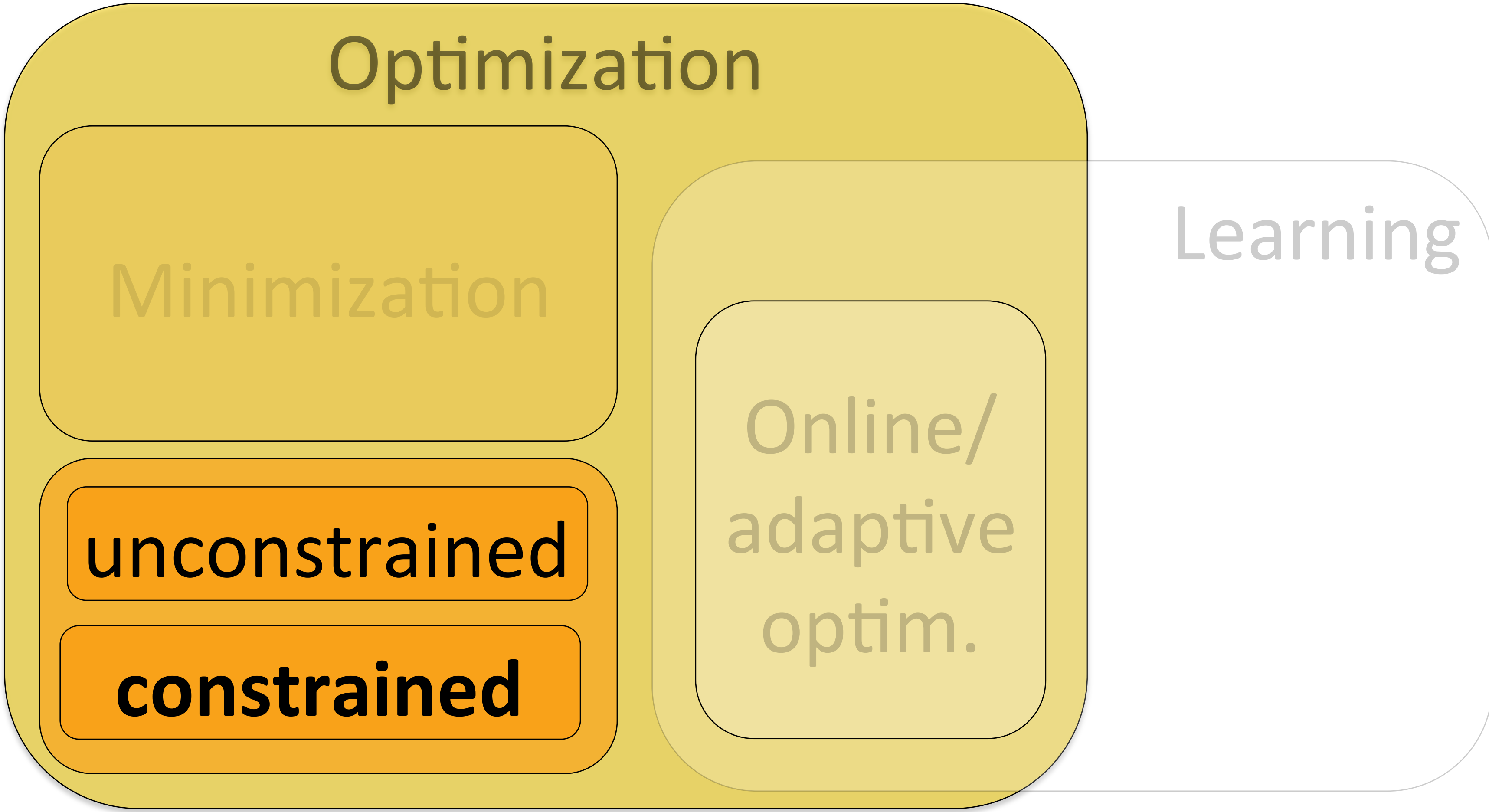
Theorem

There is an efficient algorithm, that, given a positive submodular function F, F({})=0, returns set A_{LS} such that

$$F(A_{LS}) \geq 1/2 \ \max_A F(A)$$

- picking a random set gives ¼ approximation (½ approximation if F is symmetric!)
- we cannot get better than ½ approximation unless P = NP

Optimization



Scalarization vs. constrained maximization

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

Option 1:

$$\max_A F(A) - C(A) \quad \text{s.t. } A \subseteq V$$

"Scalarization"

Can get 1/2 approx…

if F(A)-C(A) ≥ 0

for all sets A

Option 2:

$$\max_A F(A) \quad \text{s.t. } C(A) \leq B$$

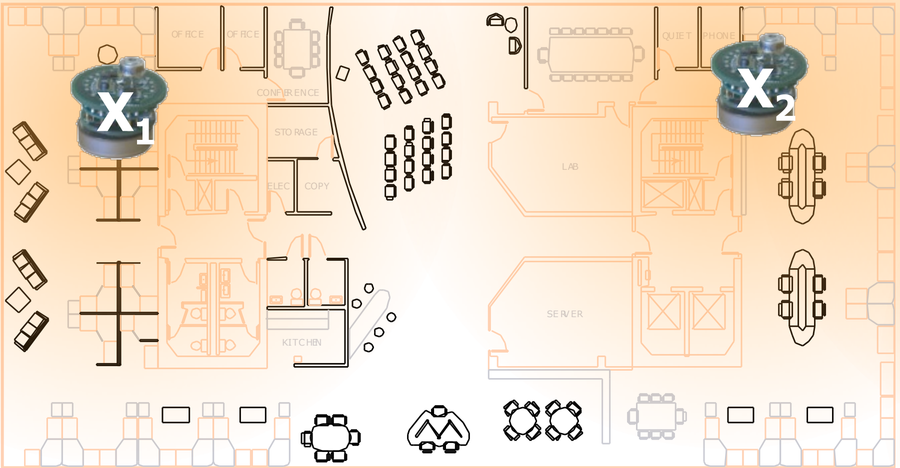
"Constrained maximization"

What is possible?

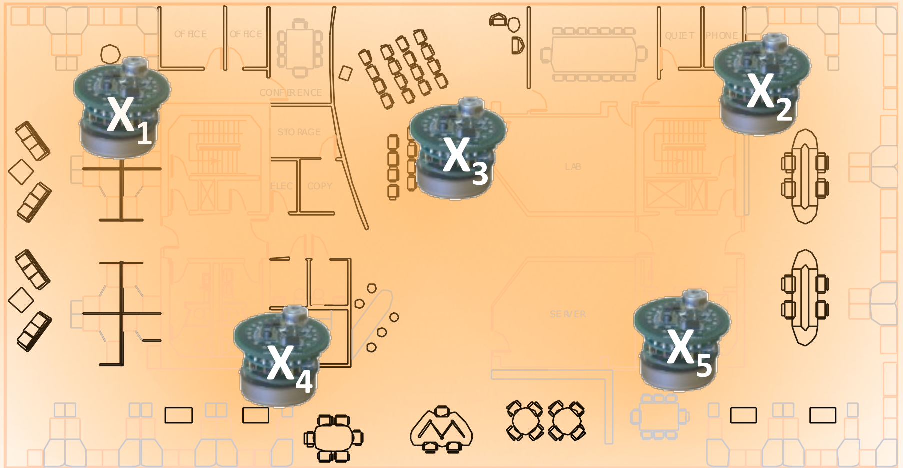
Positiveness is a strong requirement ☹

Monotonicity

Placement A = {1,2}



Placement B = {1,…,5}



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

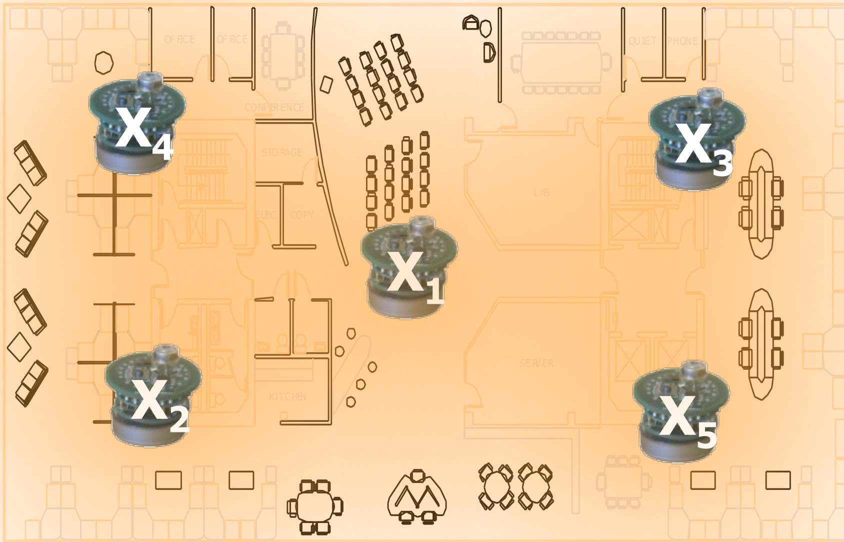
$$\Delta(s \mid A) \geq 0$$

Adding sensors can only help

Constrained maximization

- Given: finite set V of locations
- Want: $\boxed{\begin{array}{l} \mathcal{A}^* \subseteq \mathcal{V} \quad \text{such that} \\ \mathcal{A}^* = \underset{|\mathcal{A}| \leq k}{\operatorname{argmax}} F(\mathcal{A}) \end{array}}$

Typically NP-hard!



Exact maximization of SFs

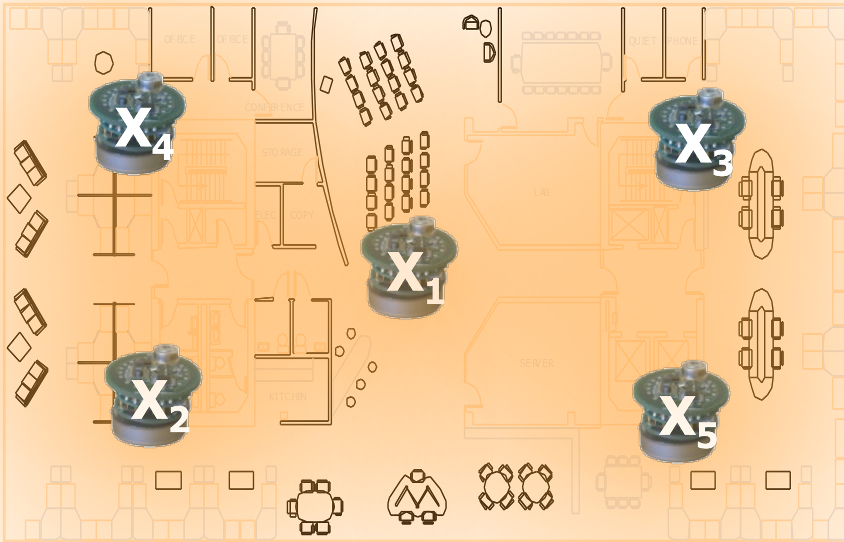
- Mixed integer programming
  - Series of mixed integer programs [Nemhauser et al ‘81]
  - Constraint generation [Kawahara et al ‘09]
- Branch-and-bound
  - „Data-Correcting Algorithm“ [Goldengorin et al ’99]

All algorithms worst-case exponential!

Greedy algorithm

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

Typically NP-hard!



<u>Greedy algorithm:</u>

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

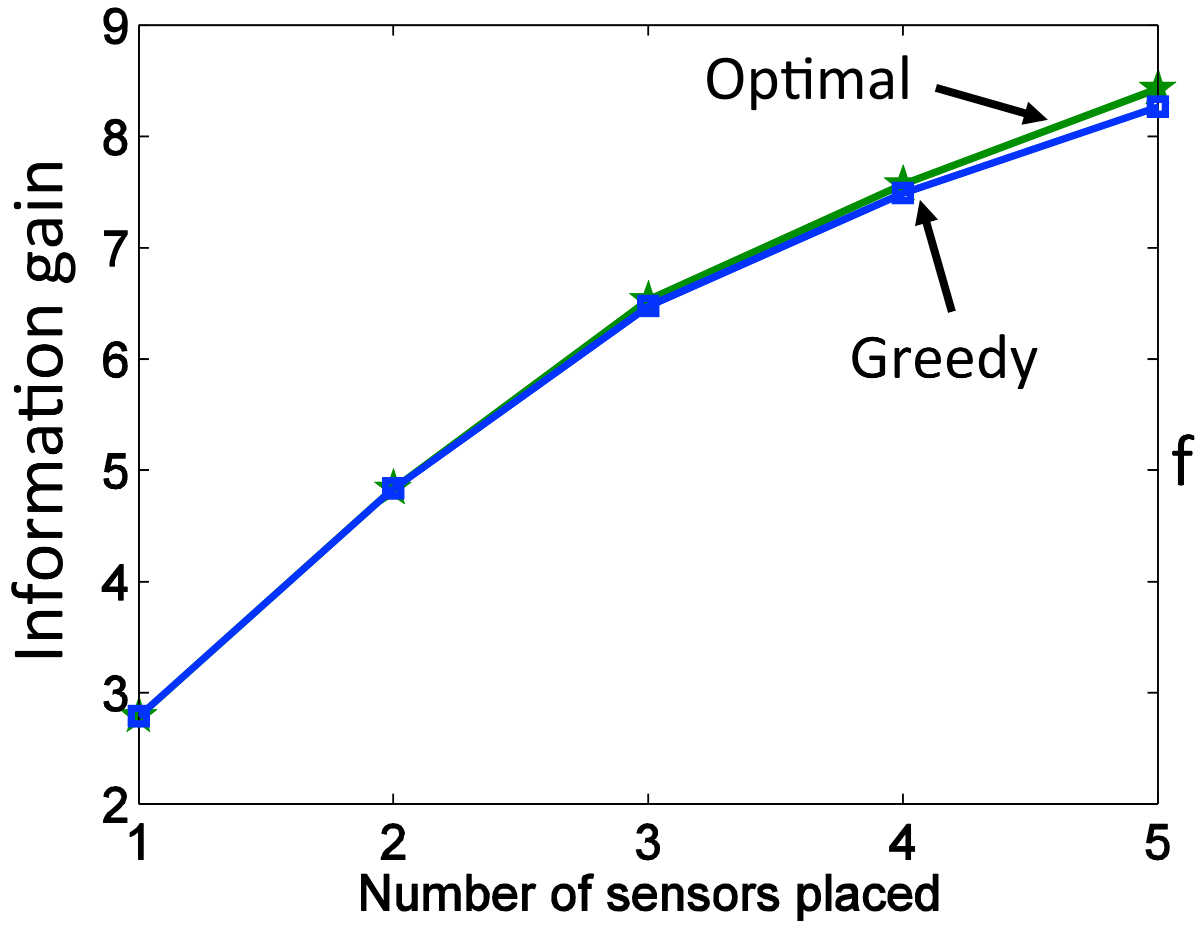
For i = 1 to k

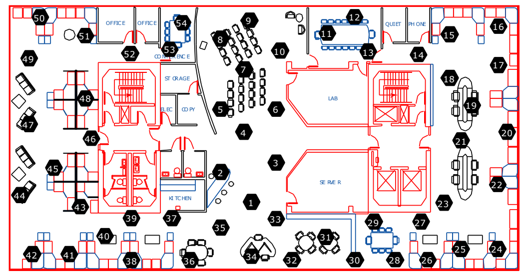
$$s^* \leftarrow \arg\max_s F(\mathcal{A} \cup \{s\})$$

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

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!
- For information gain: Guarantees best possible unless P = NP!
[Krause & Guestrin '05]

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" $\Delta(s \mid A_i)$

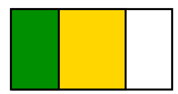
$$\Delta(s \mid A_i) = F(A_i \cup \{s\})\text{-}F(A_i)$$

Key observation: Submodularity implies

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

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

s

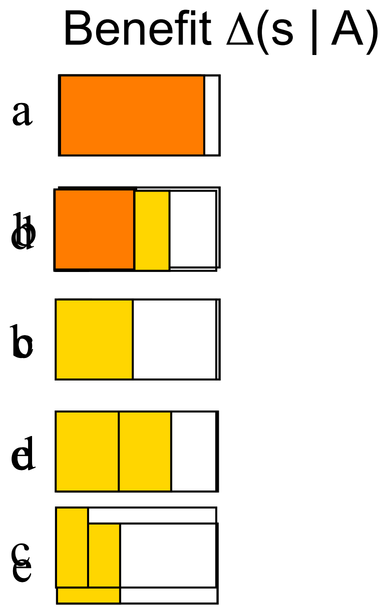


Marginal benefits can never increase!

"Lazy" greedy algorithm [Minoux '78]

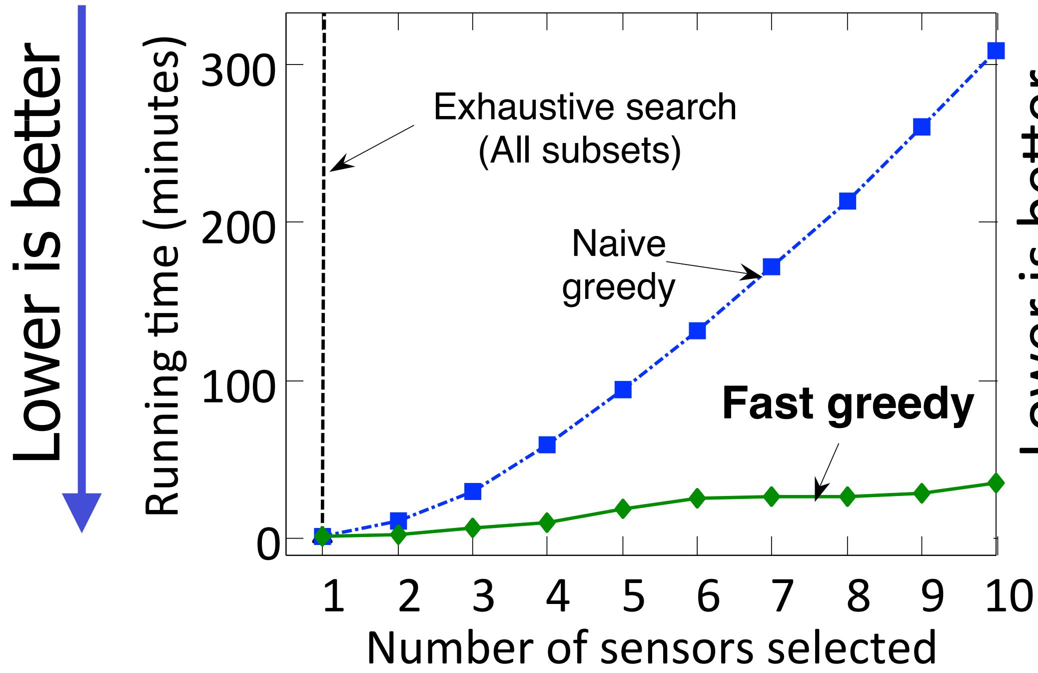
Lazy greedy algorithm: $\mathcal{M}$

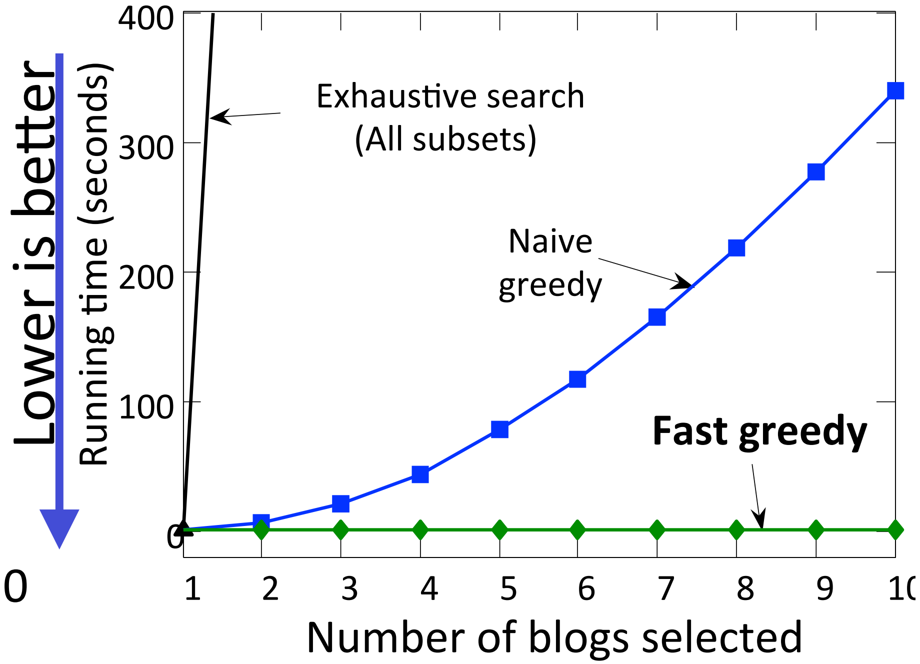
- First iteration as usual
- Keep an ordered list of marginal benefits Δ_i from previous iteration
- Re-evaluate Δ_i only for top element
- If Δ_i stays on top, use it, otherwise re-sort



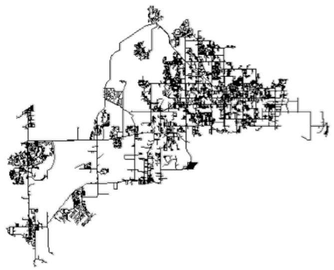
Note: Very easy to compute online bounds, lazy evaluations, etc. [Leskovec, Krause et al. '07]

Empirical improvements [Leskovec, Krause et al'06]





Sensor placement



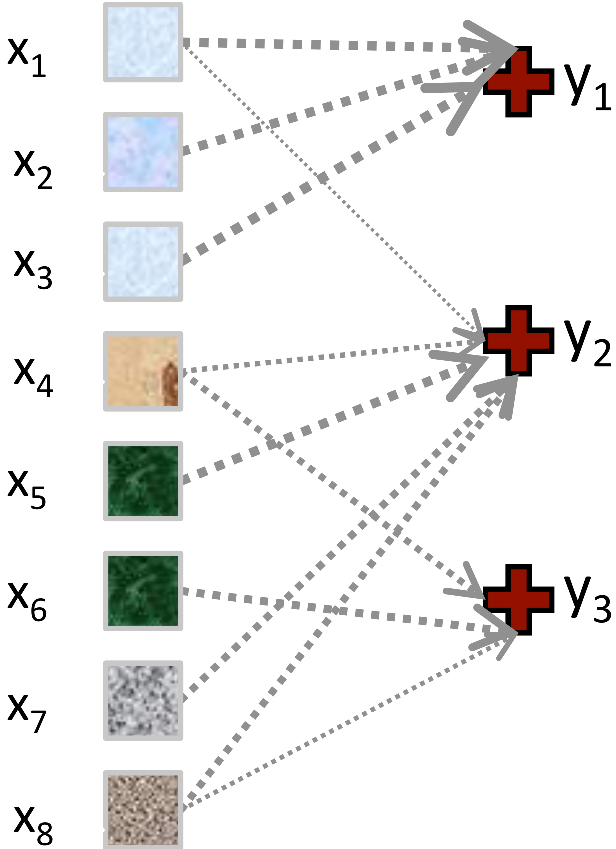
Blog selection

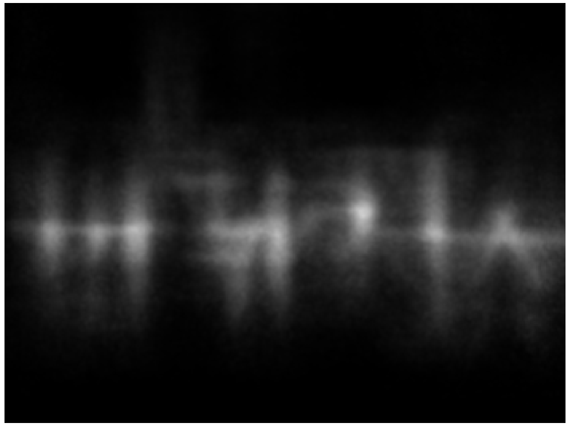
30x speedup

700x speedup

Object detection [Barinova et al.'10]

x_j = index of hypothesis explaining x_j





y_i = 1: object i present

y_i = 0: object i not present

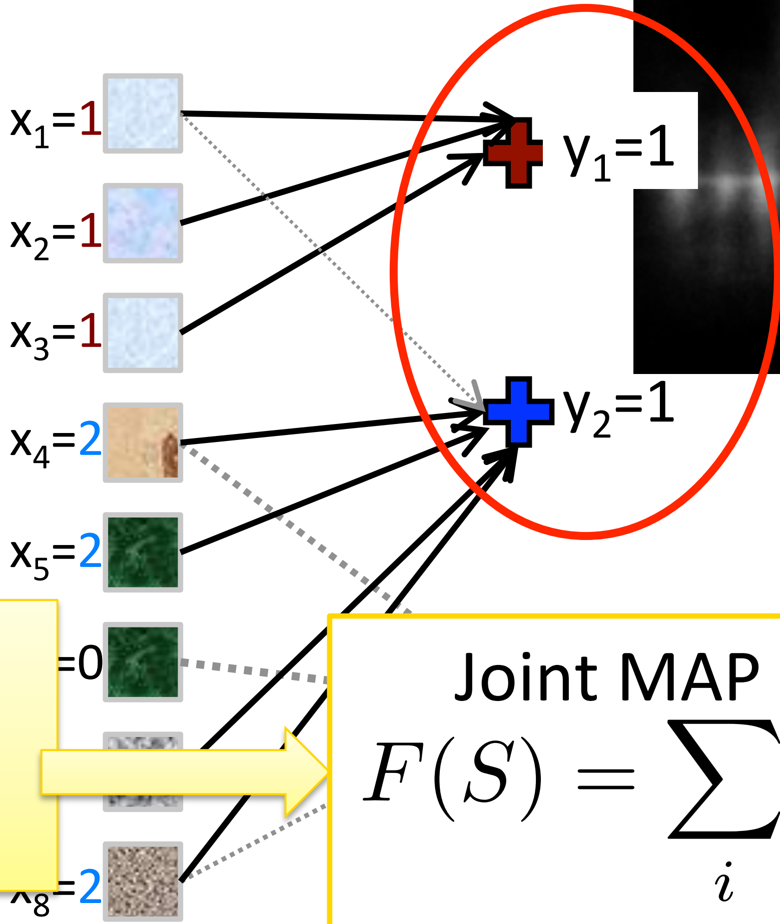
Voting elements

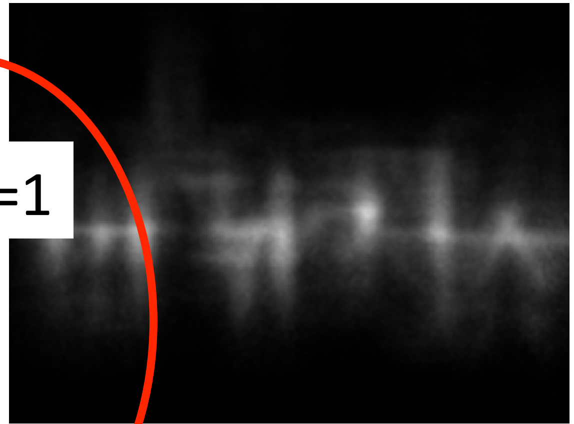
Hypotheses

Illustrations courtesy of Pushmeet Kohli

Object detection

x_j = index of hypothesis explaining x_j





y_i = 1: object i present

submodular maximization ☺

Joint MAP inference:

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

Weight element i wrt hyp. j

Voting elements

Illustrations courtesy of Pushmeet Kohli

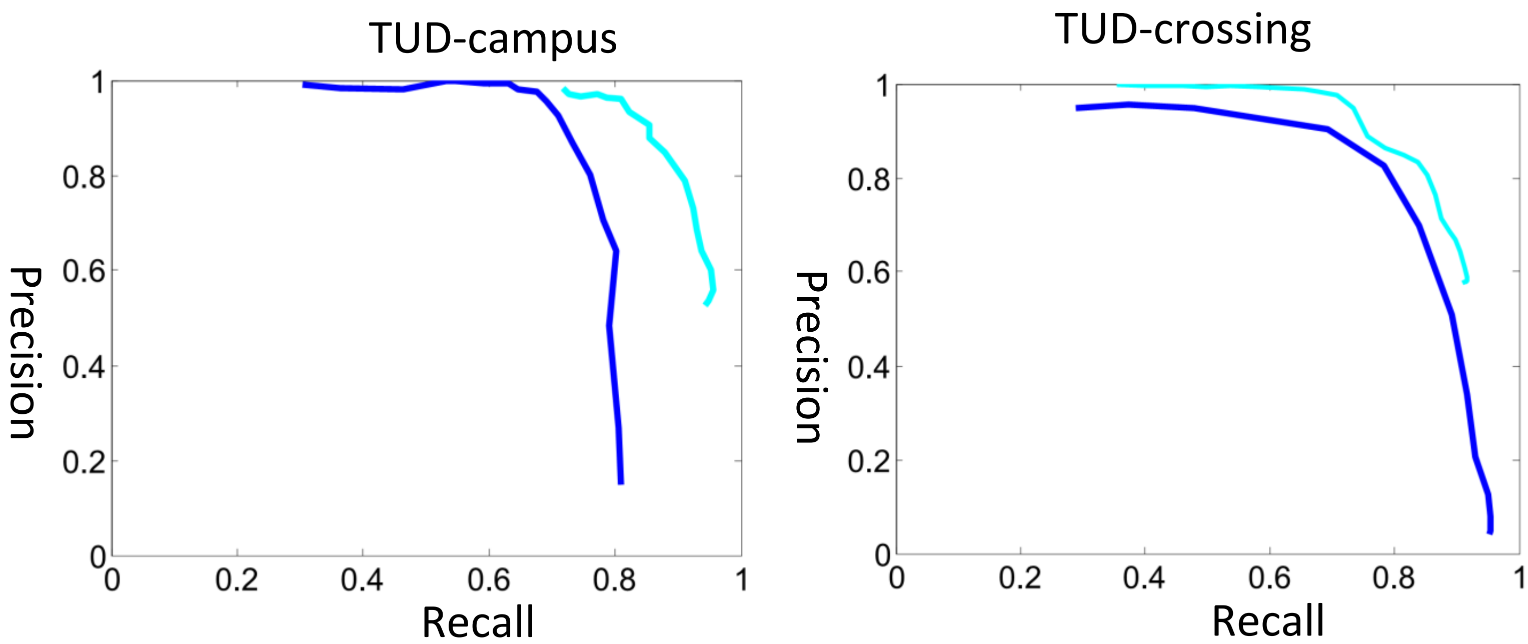
Inference

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

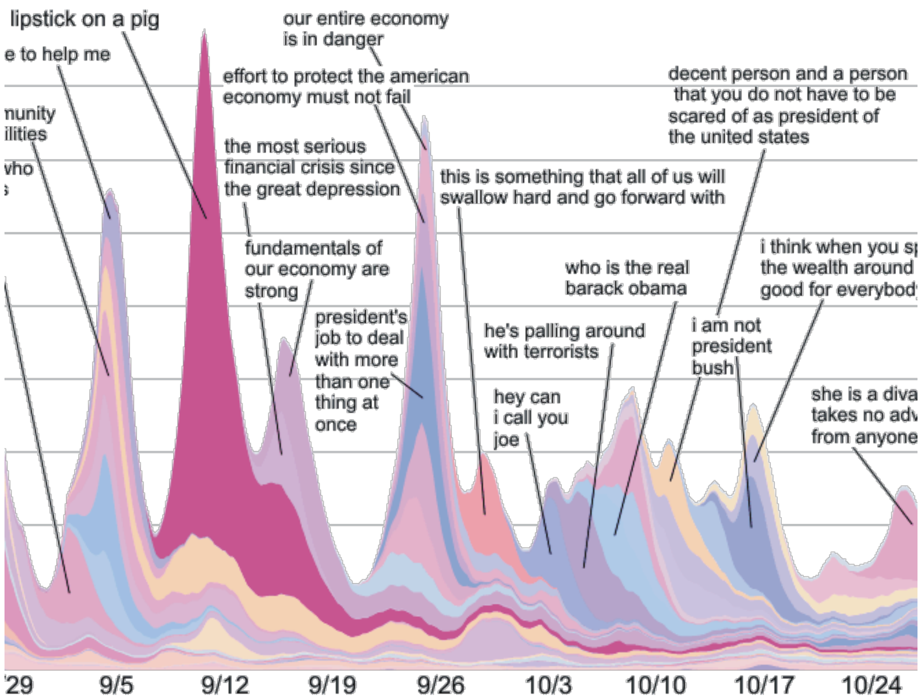


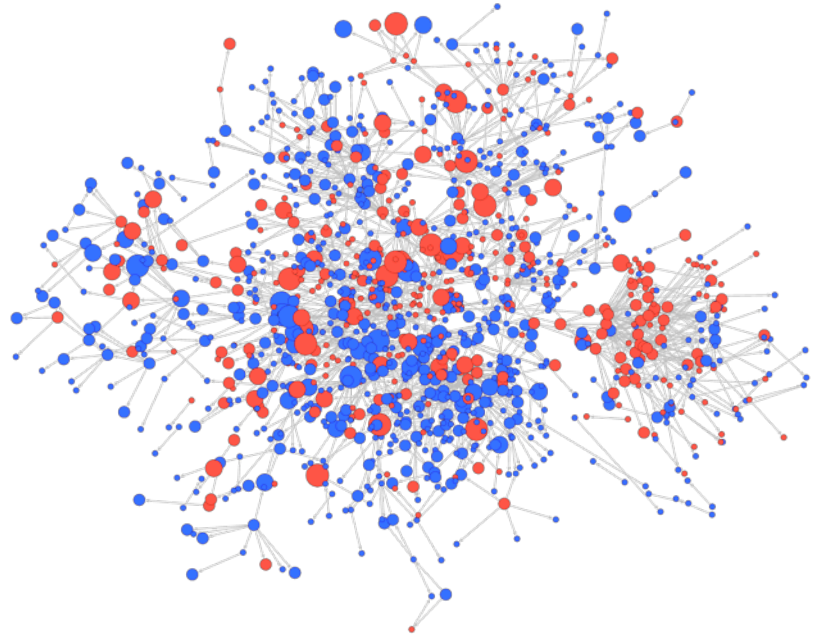
Blue = Hough transform + non-maximum suppression

Light-blue = greedy detection

submodularity for detection also in [Blaschko'11]

Network inference

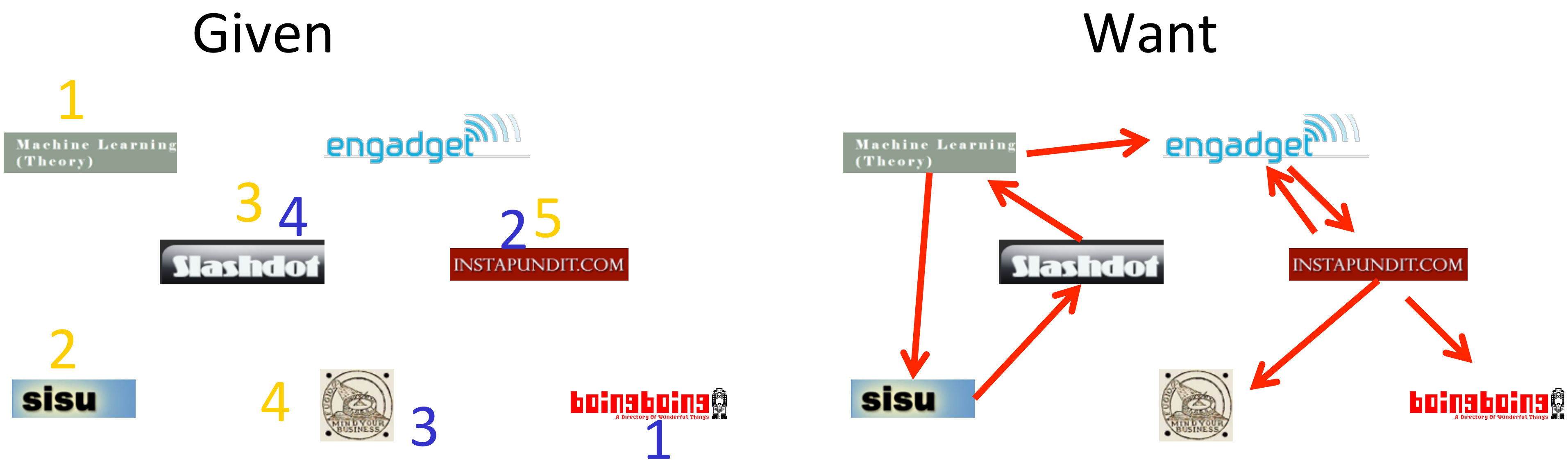




How can we learn who influences whom?

Inferring diffusion networks

[Gomez Rodriguez, Leskovec, Krause ACM TKDE 2012]

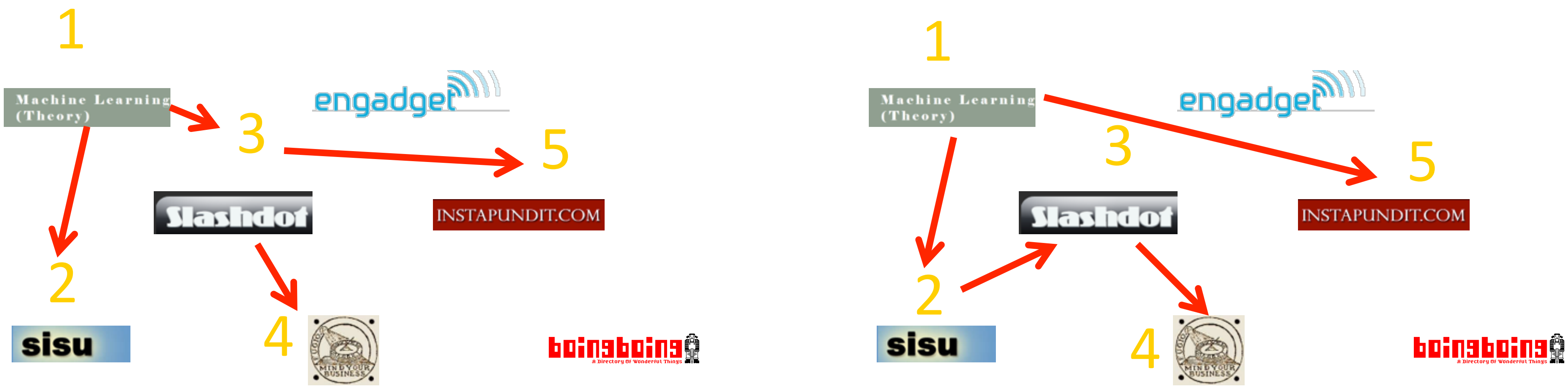


Given traces of influence, wish to infer sparse directed network G=(V,E)

➔ Formulate as optimization problem

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

Estimation problem

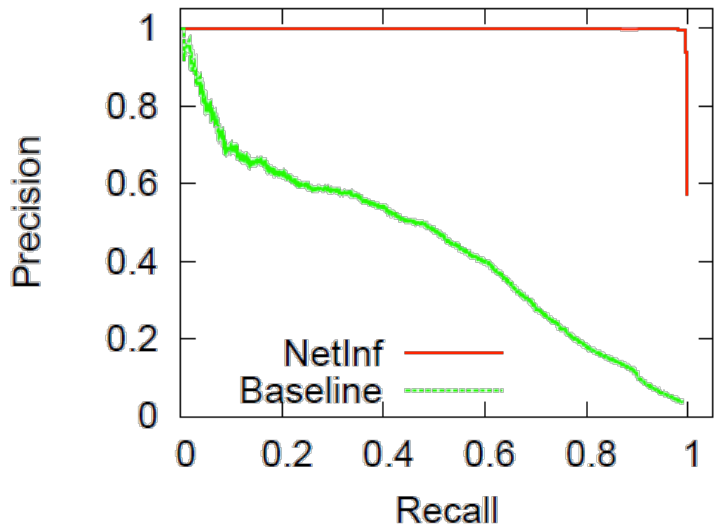


- Many influence trees T consistent with data
- For cascade C_i, model $P(C_i \mid T)$
- Find sparse graph that maximizes likelihood for all observed cascades

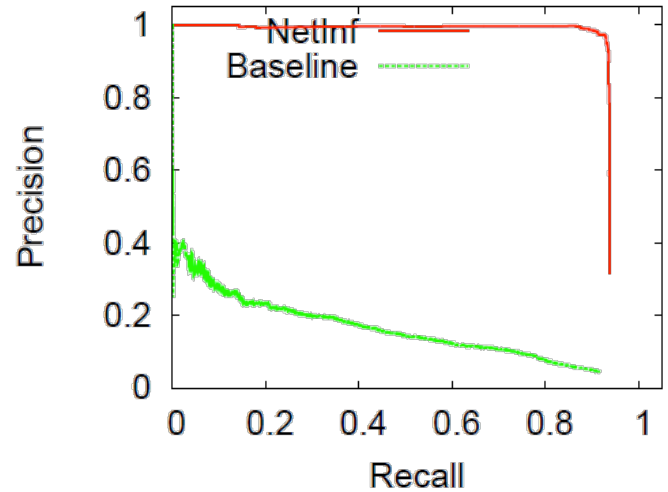
➔ Log likelihood monotonic submodular in selected edges

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

Evaluation: Synthetic networks



1024 node hierarchical Kronecker exponential transmission model

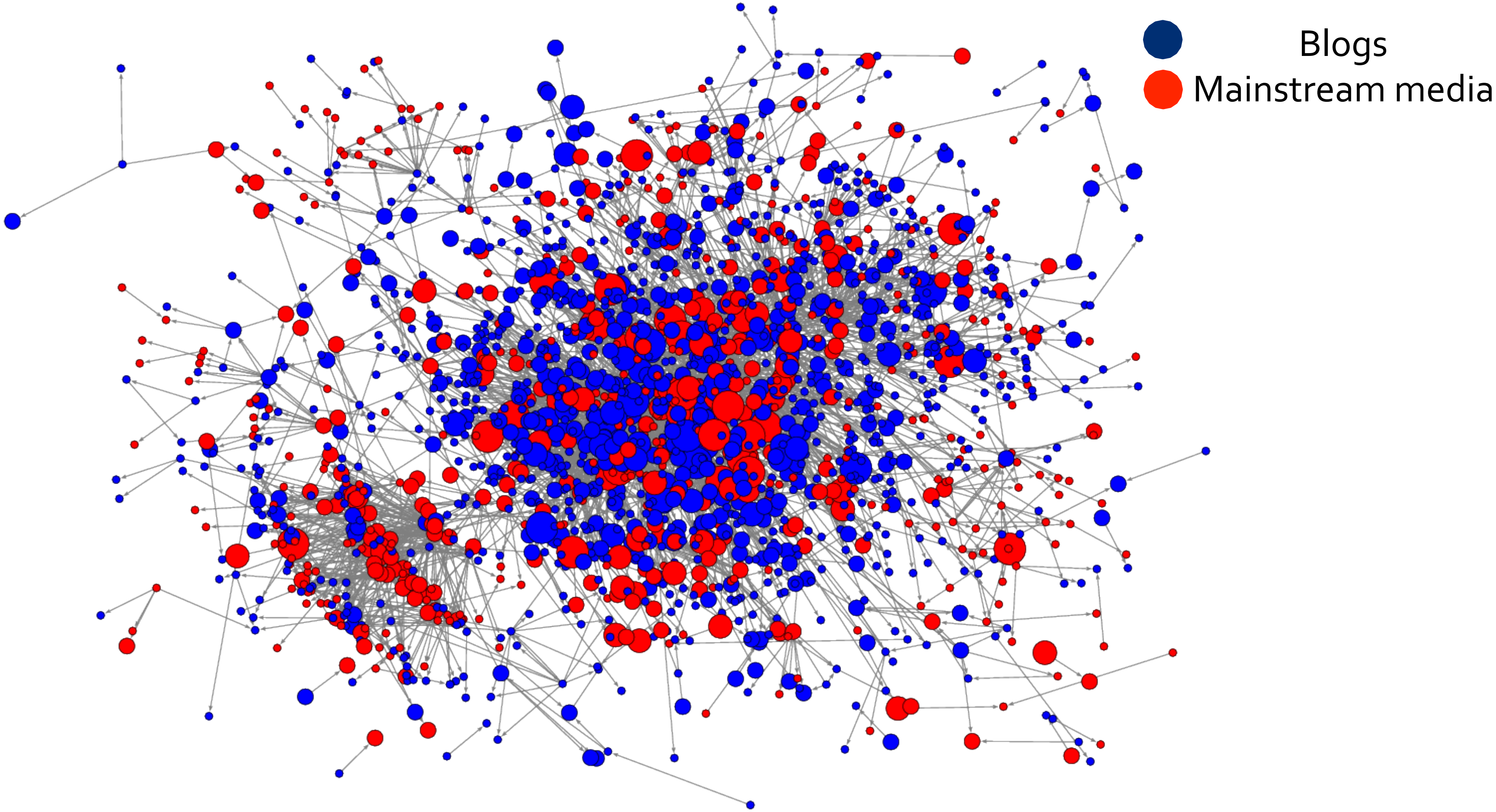


1000 node Forest Fire (α = 1.1) power law transmission model

- 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%

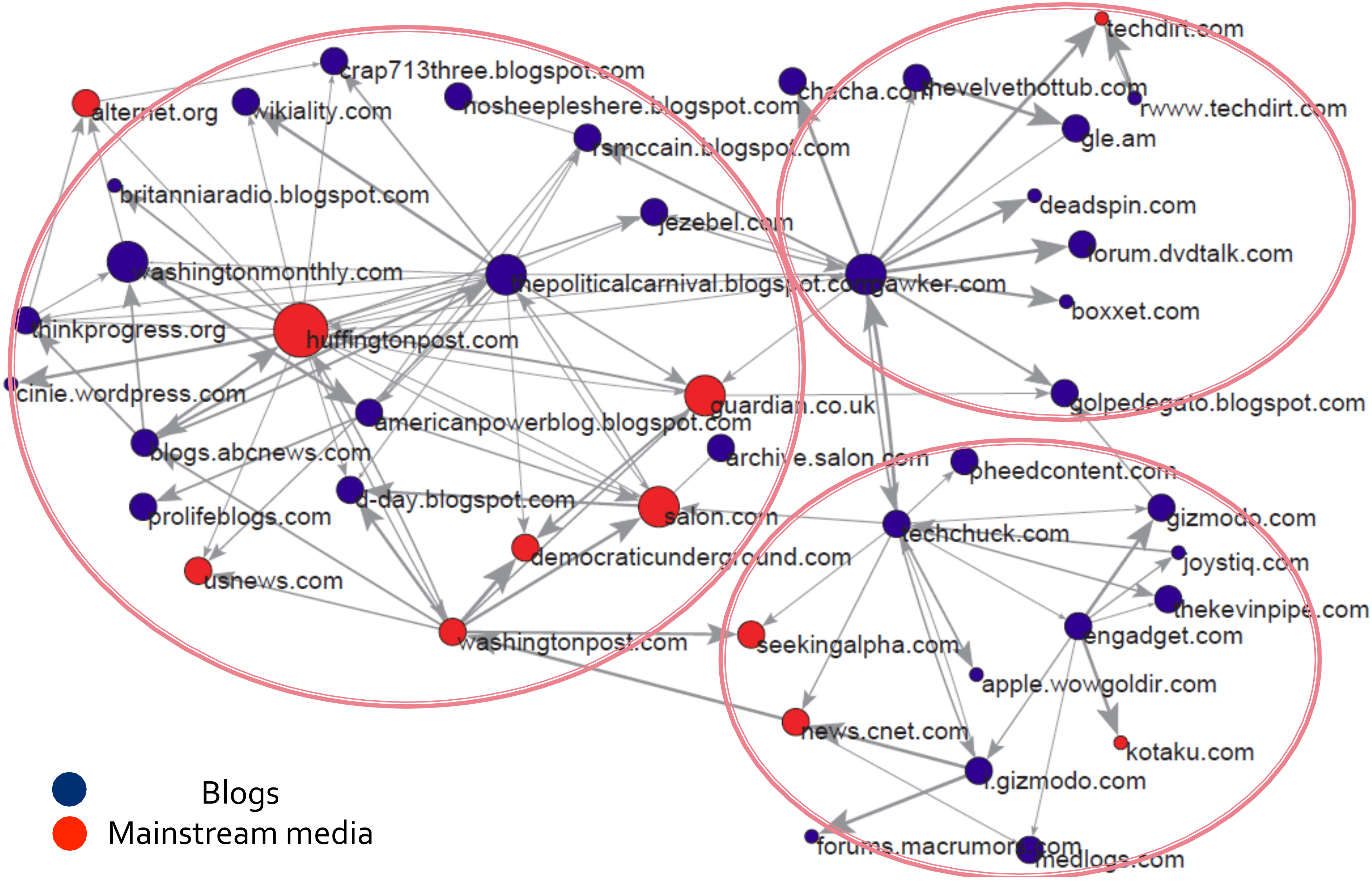
Diffusion Network

[Gomez Rodriguez, Leskovec, Krause ACM TKDE 2012]



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

Diffusion Network (small part)



Document summarization [Lin & Bilmes '11]

- Which sentences should we select that best summarize a document?

Marginal gain of a sentence

- Many natural notions of „document coverage“ are submodular [Lin & Bilmes ‘11]

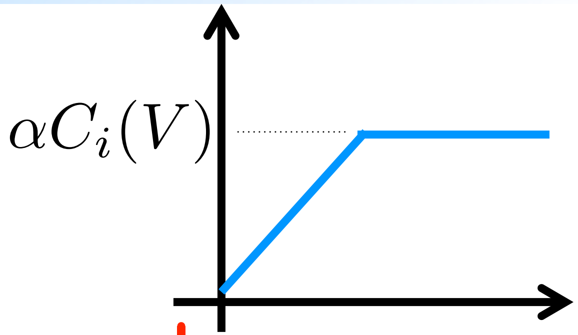
Document summarization

$$F(S) = R(S) + \lambda D(S)$$

Relevance Diversity

Relevance of a summary

$$F(S) = R(S) + \lambda D(S)$$



$$R(S) = \sum_i \min\{C_i(S), \alpha C_i(V)\}$$

How well is sentence i „covered“ by S

$$C_i(S) = \sum_{j \in S} w_{i,j}$$

Similarity between i and j

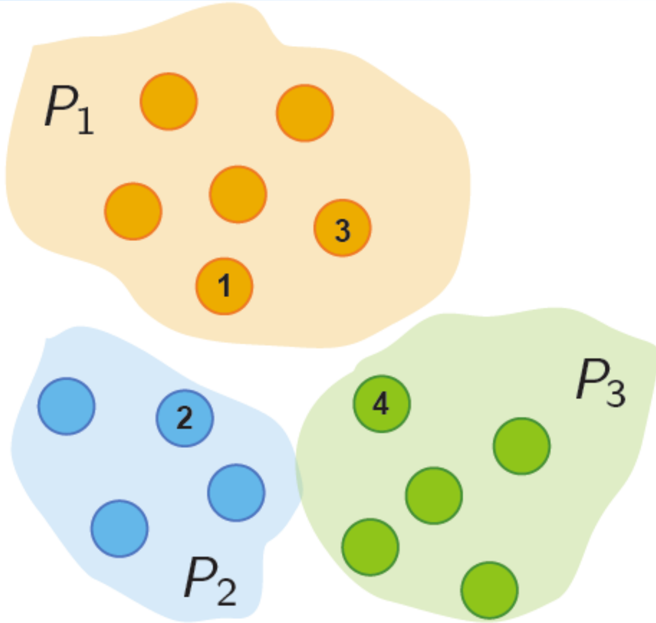
Diversity of a summary

$$D(S) = \sum_{i=1}^{K} \sqrt{\sum_{j \in P_i \cap S} r_j}$$

Relevance of sentence j to doc.

$$r_j = \frac{1}{N} \sum_i w_{i,j}$$

Similarity between i and j



Clustering of sentences in document

Can be made query-specific; multi-resolution; etc.

Empirical results [Lin & Bilmes '11]

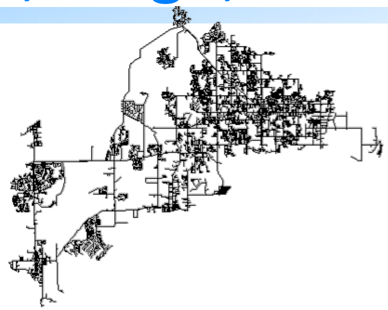
	R	F
$\mathcal{L}_1(S) + \lambda\mathcal{R}_Q(S)$	12.18	12.13
$\mathcal{L}_1(S) + \sum_{\kappa=1}^{3} \lambda_\kappa \mathcal{R}_{Q,\kappa}(S)$	**12.38**	$\boxed{\textbf{12.33}}$
Toutanova et al. (2007)	11.89	11.89
Haghighi and Vanderwende (2009)	11.80	-
Celikyilmaz and Hakkani-tür (2010)	11.40	-
Best system in DUC-07 (peer 15), using web search	**12.45**	12.29

Best F1 score on benchmark corpus DUC-07!

Submodular Sensing Problems

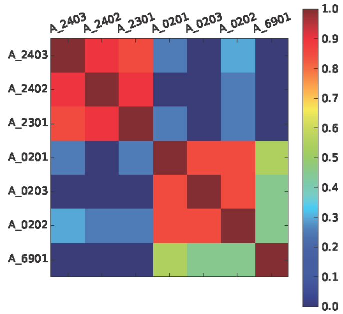
[with Guestrin, Leskovec, Singh, Sukhatme, …]

Environmental monitoring
[JAIR ’08, ICRA ‘10]



Top score in BWSN competition

Water distribution networks
[J WRPM ’08 *Best paper*]



Experiment design [NIPS ‘10, ‘11]

Recommending blogs & news
[KDD ‘07, ’10 *Best paper*]

Can all be reduced to monotonic submodular maximization

More complex constraints

- So far: $$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}| \le k} F(\mathcal{A})$$

- Can one handle more complex constraints?

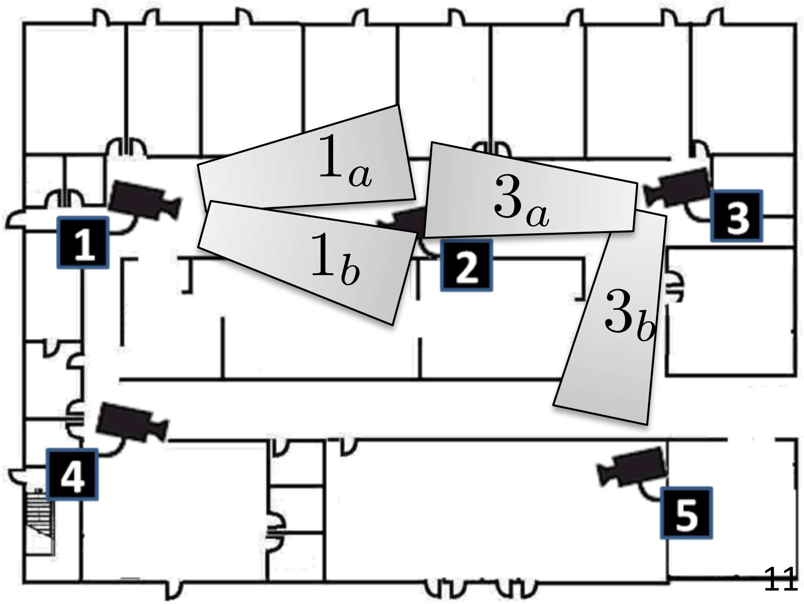
Example: Camera network

Ground set $V = \{1_a, 1_b, \ldots, 5_a, 5_b\}$

Configuration: $S = \{v^1, \ldots, v^k\}$

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

Configuration is feasible if no camera is pointed in two directions at once



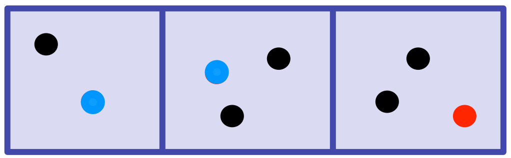
Matroids

- Abstract notion of feasibility: independence

S is independent if …

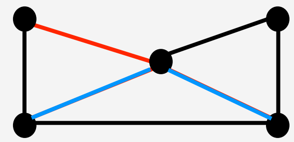
… $|S| \leq k$

Uniform matroid



… S contains at most one element from each square

Partition matroid



… S contains no cycles

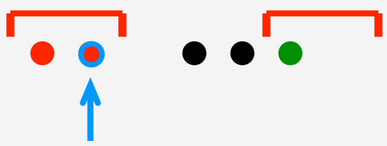
Graphic matroid

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

Matroids

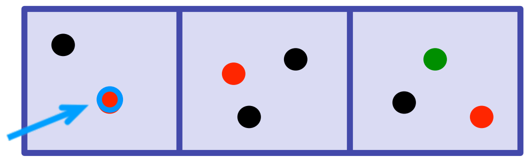
- Abstract notion of feasibility: independence

S is independent if …



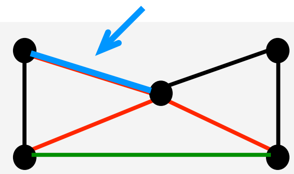
… |S| ≤ k

Uniform matroid



… S contains at most one element from each group

Partition matroid



… S contains no cycles

Graphic matroid

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

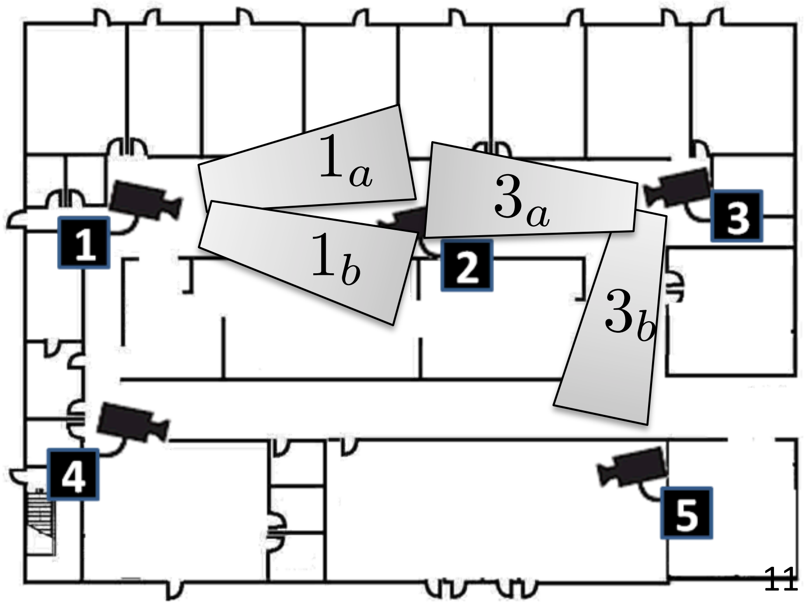
Example: Camera network

Ground set $V = \{1_a, 1_b, \ldots, 5_a, 5_b\}$

Configuration: $S = \{v^1, \ldots, v^k\}$

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

Configuration is feasible if no camera is pointed in two directions at once



This is a partition matroid:

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

Independence:

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

Greedy algorithm for matroids:

- Given: finite set V
- Want: $\mathcal{A}^* \subseteq \mathcal{V}$ such that

$$\mathcal{A}^* = \operatorname*{argmax}_{A \text{ independent}} F(A)$$

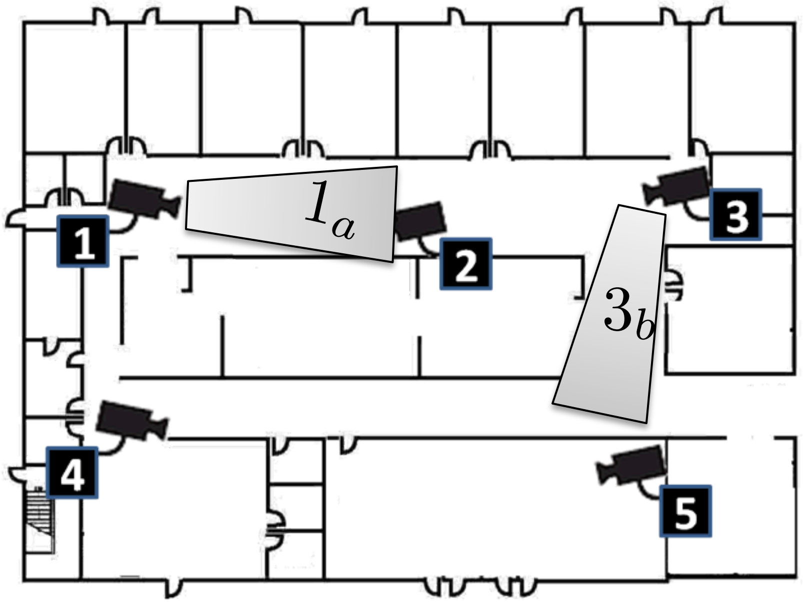
<u>Greedy algorithm:</u>

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

While $\exists s : A \cup \{s\}$ indep.

$$s^* \leftarrow \operatorname*{argmax}_{s:\, A\cup\{s\} \text{ indep.}} F(A \cup \{s\})$$

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



Maximization over matroids

Theorem [Nemhauser, Fisher & Wolsey ’78]

For monotonic submodular functions,
Greedy algorithm gives constant factor approximation

$$F(A_{greedy}) \geq ½ \; F(A_{opt})$$

- Greedy gives 1/(p+1) over intersection of p matroids
  - Can model rankings with p=2!
- Can get also obtain (1-1/e) for arbitrary matroids [Vondrak et al ’08] using continuous greedy algorithm

Non-constant cost functions

- For each $s \in V$, let $c(s)>0$ be its cost (e.g., length of sentence; feature acquisition costs, …)
- Cost of a set $C(A) = \sum_{s \in A} c(s)$
- Want to solve

$$\boxed{A^* = \operatorname{argmax} F(A) \quad \text{s.t.} \quad C(A) \leq B}$$

Cost-benefit optimization

[Wolsey ‘82, Sviridenko ‘04, Krause et al ‘05]

Theorem [Krause and Guestrin ‘05]

Can efficiently find a set A such that

Then $$F(A) \geq \left(1 - \frac{1}{\sqrt{e}}\right) \max_{C(A') \leq B} F(A')$$

~39%

Can still speed up using lazy evaluations

Note: Can also get

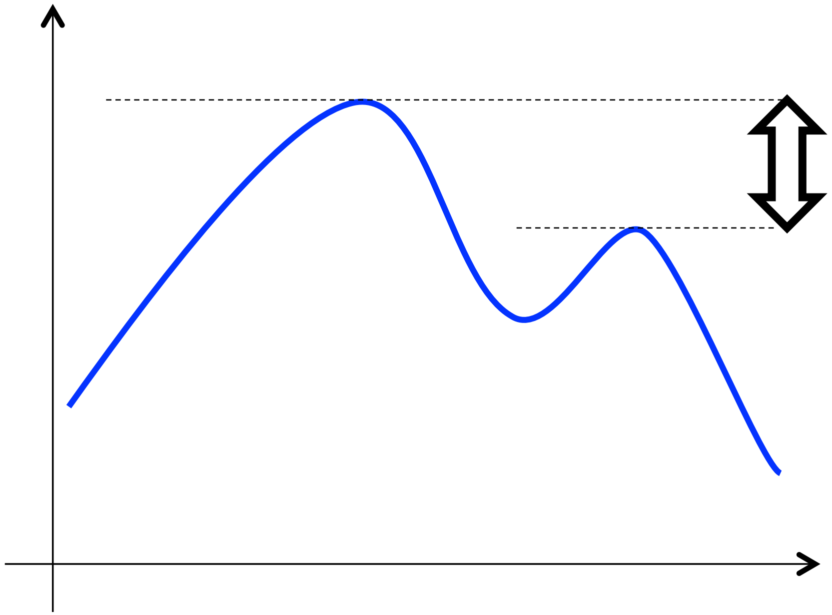
- (1-1/e) approximation in time $O(n^4)$ [Sviridenko ’04]
- (1-1/e) approximation for multiple linear constraints [Kulik ‘09]
- 0.38/k approximation for k matroid and m linear constraints [Chekuri et al ‘11]

Summary: More complex constraints

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

- Often the (best) algorithms are **non-greedy**

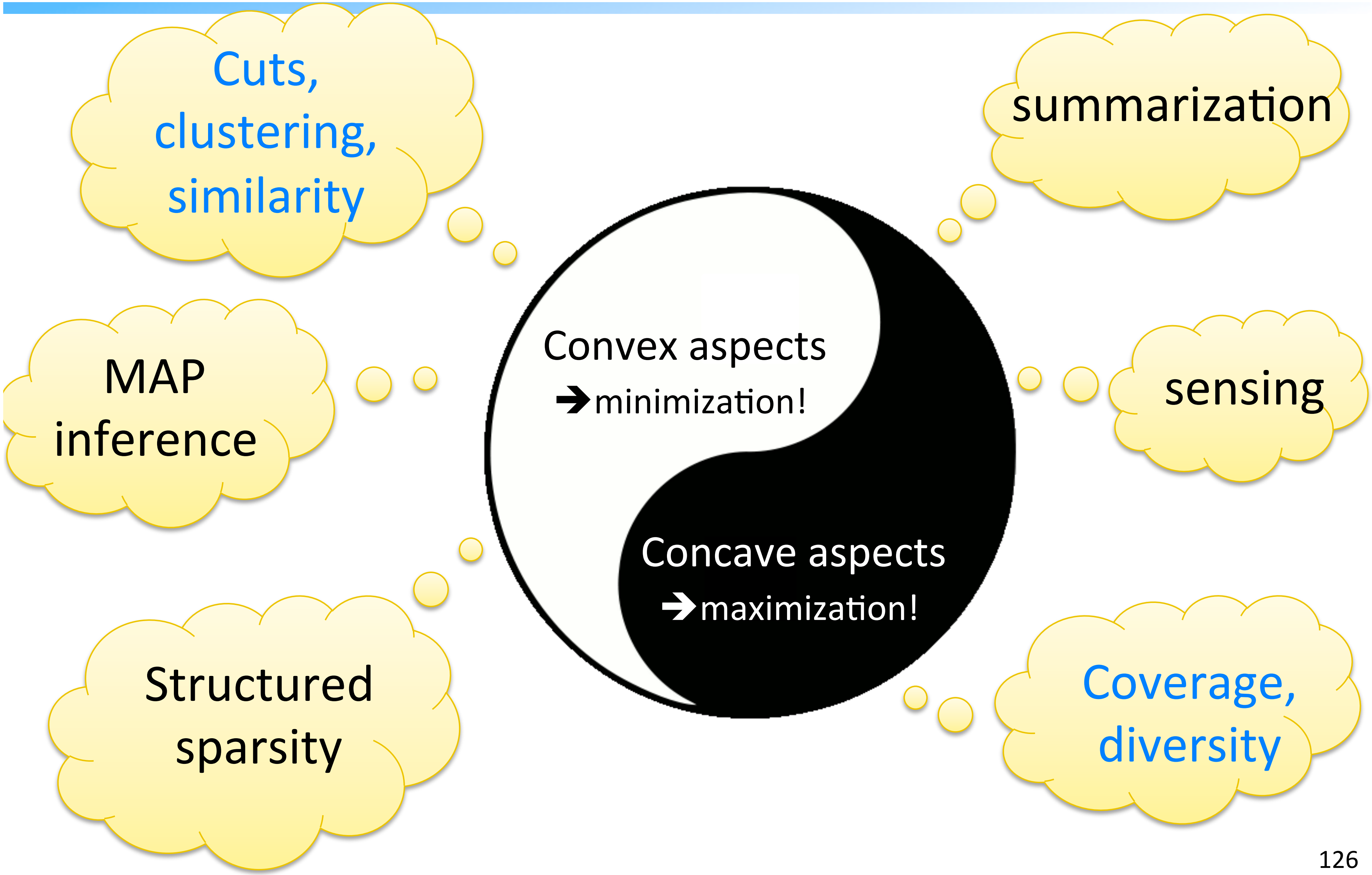
Key intuition for approx. maximization



For submod. functions, local maxima can't be too bad

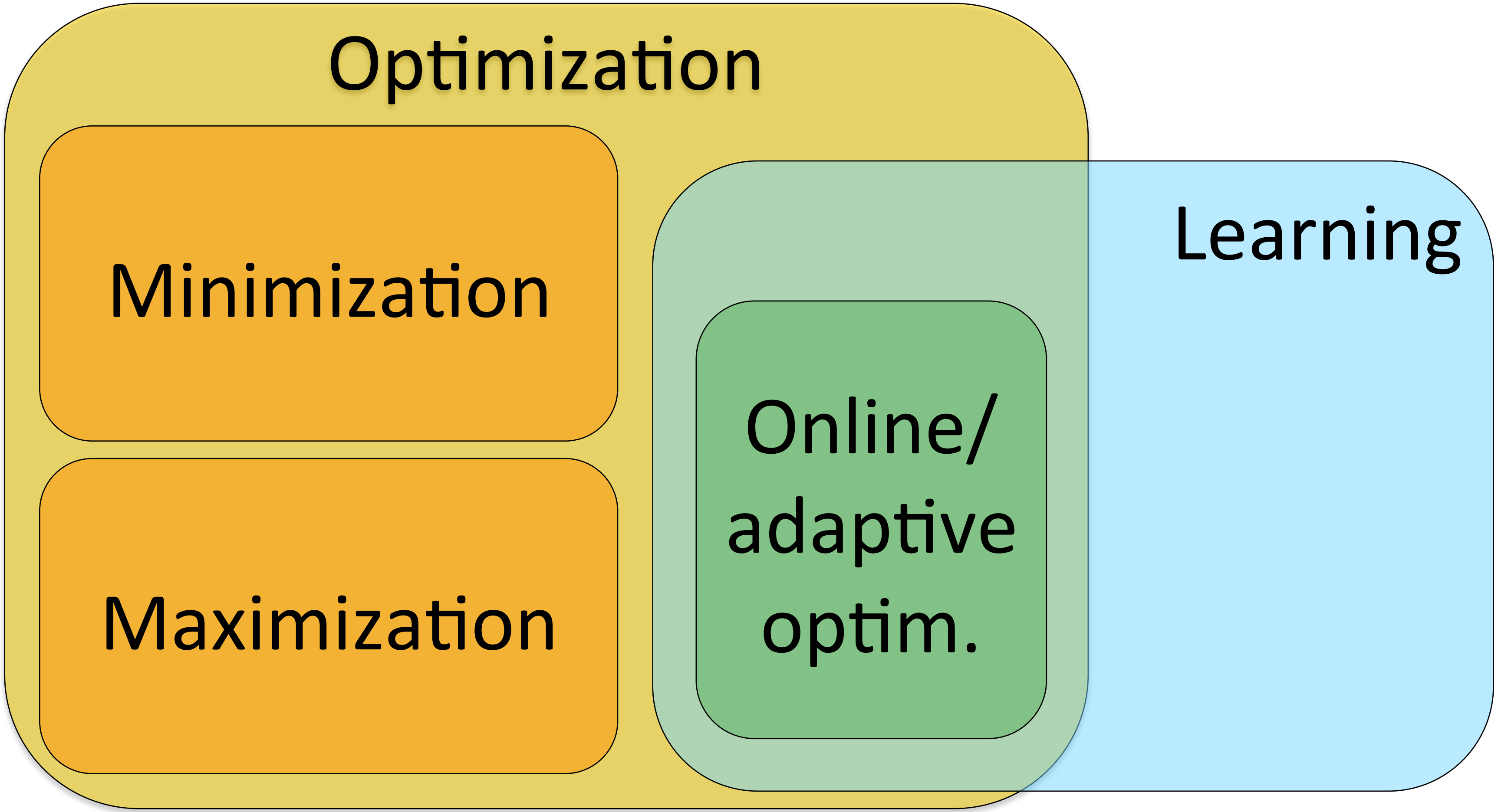
- E.g., all local maxima under cardinality constraints are within factor 2 of global maximum
- Key insight for more complex maximization
  ➔ Greedy, local search, simulated annealing for (non-monotone, constrained, ...)

Two-faces of submodular functions

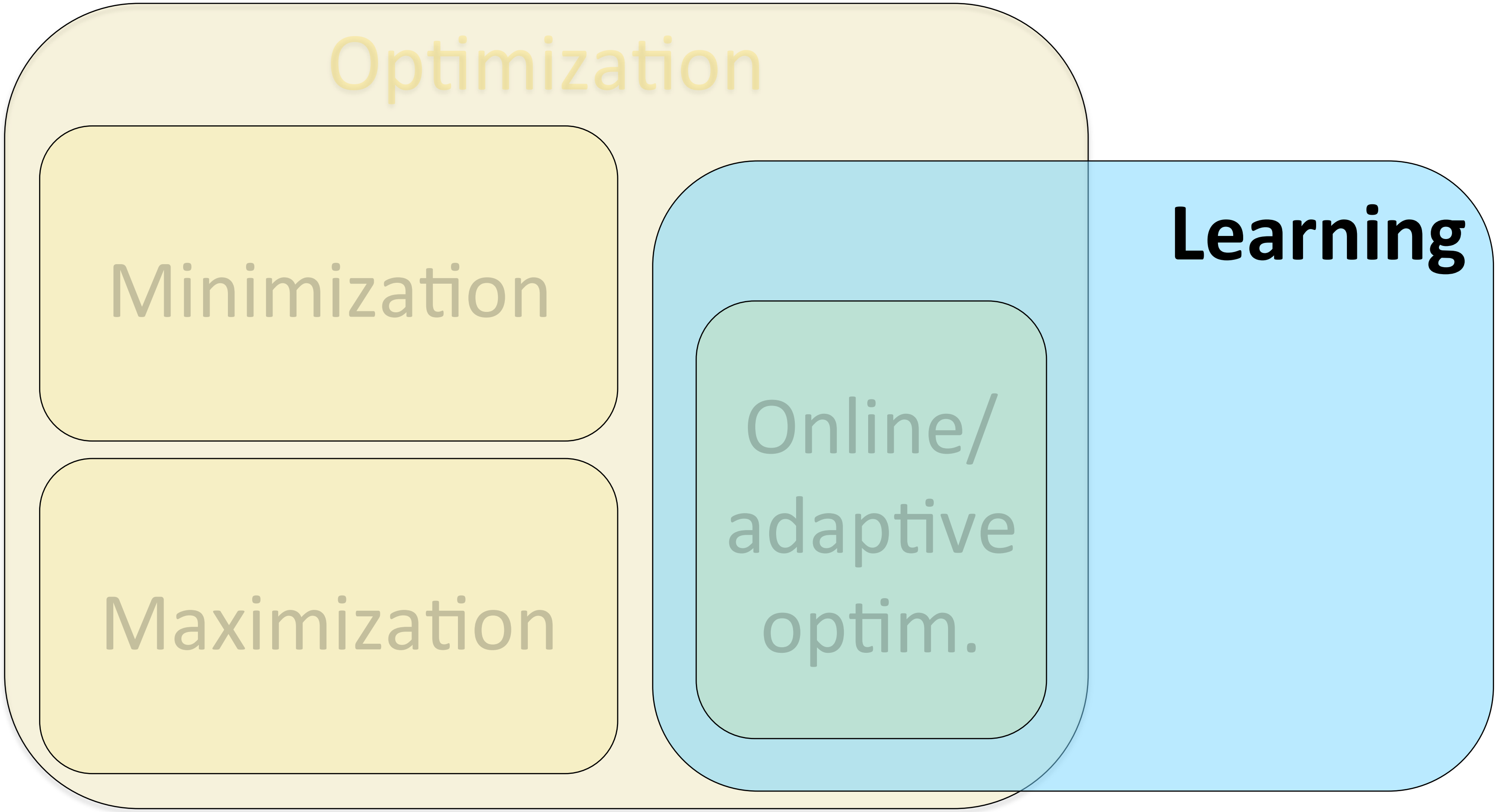


	Maximization	Minimization
Unconstrained	NP-hard, but well-approximable (if nonnegative)	Polynomial time! Generally inefficent (n^6), but can exploit special cases (cuts; symmetry; decomposable; ...)
Constrained	NP-hard but well-approximable „Greedy-(like)“ for cardinality, matroid constraints; Non-greedy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

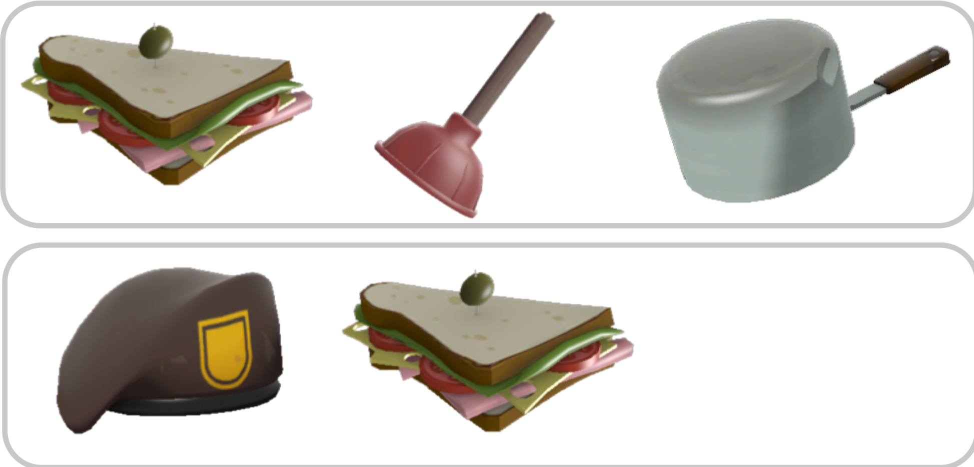
What to do with submodular functions



What to do with submodular functions



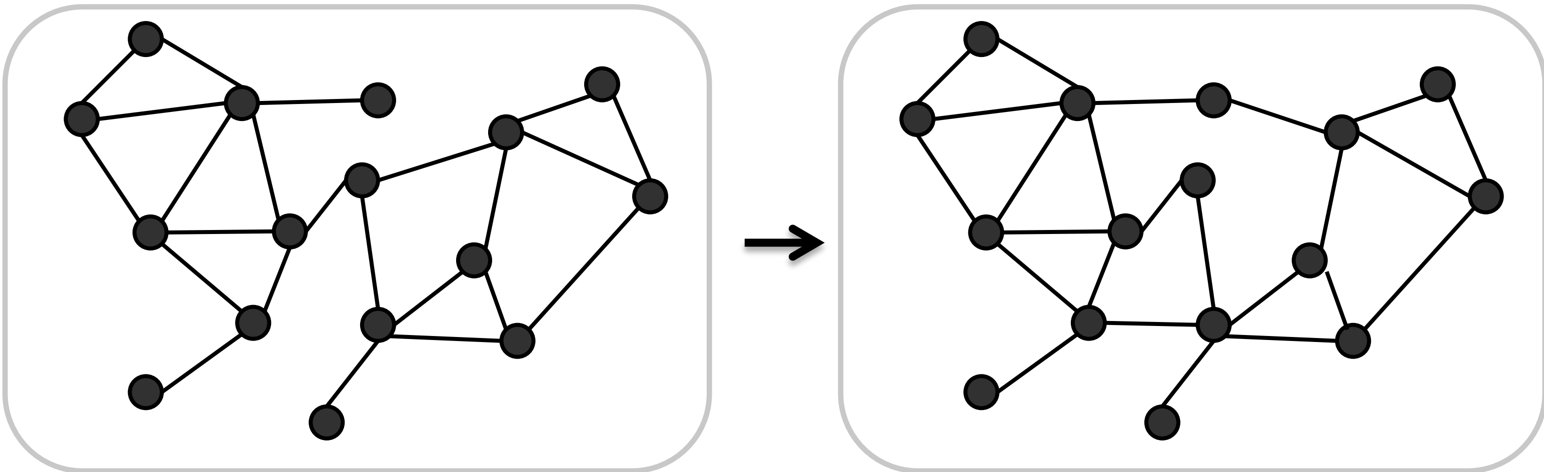
Example 1: Valuation Functions



- For combinatorial auctions, show bidders various subsets of items, see their bids

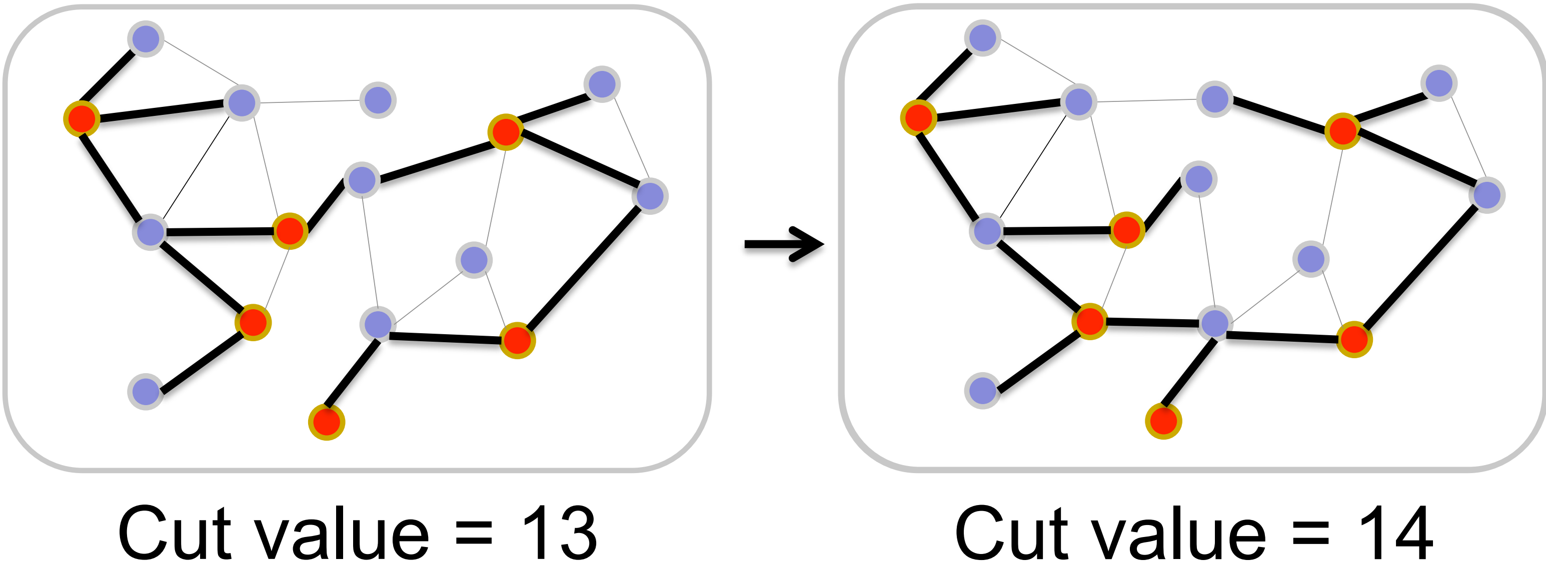
Can we learn a bidder's utility function from few bids?

Example 2: Graph Evolution



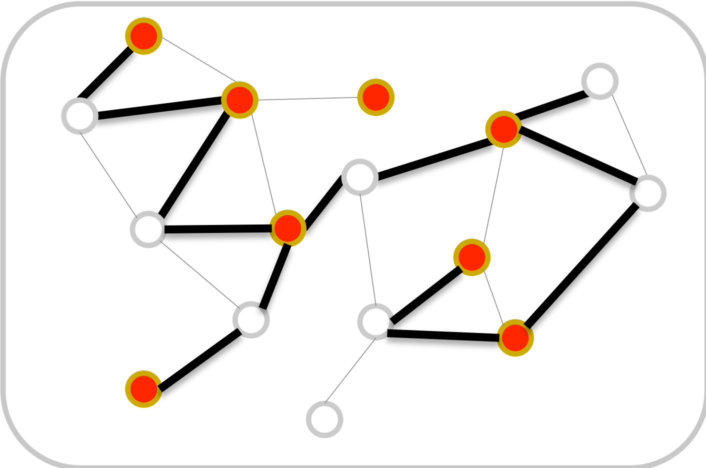
- Want to track changes in a graph
- Instead of storing entire graph at each time step, store some measurements
- # of measurements << # of edges in graph

Random Graph Cut #1

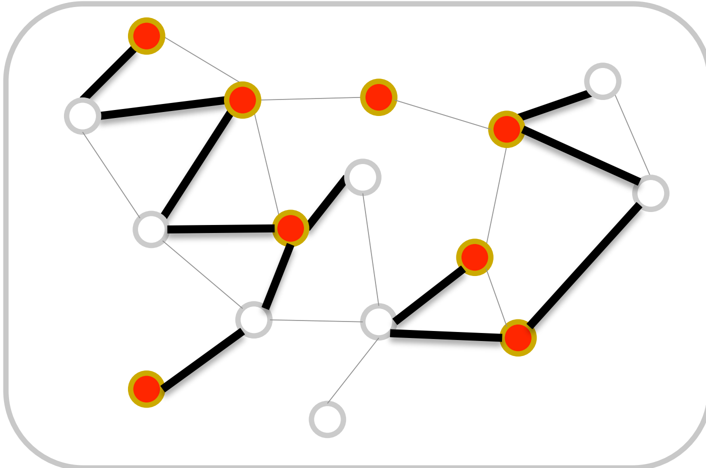


- Choose a random partition of vertices
- Count total # of edges across partition

Random Graph Cut #2



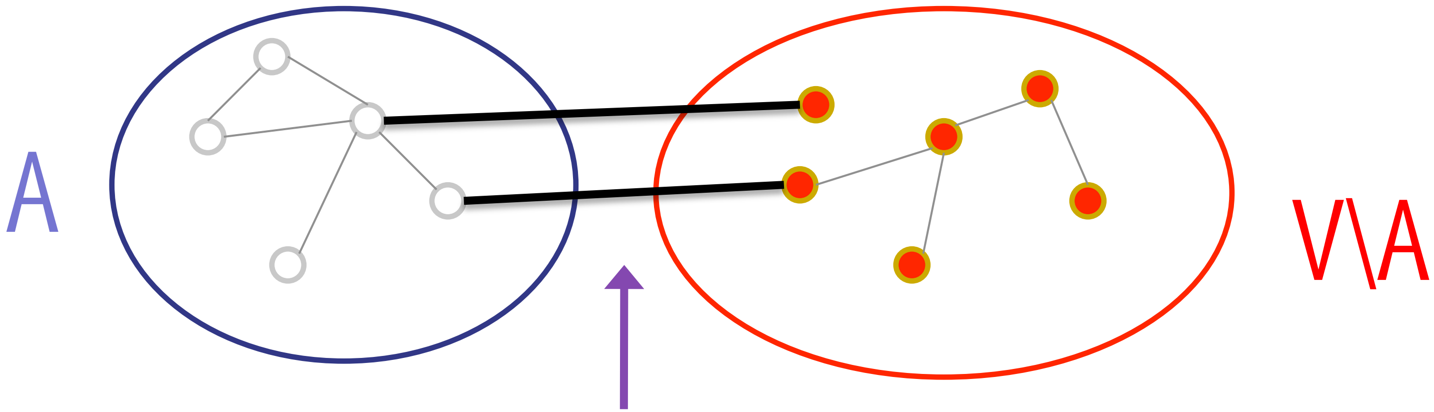
Cut value = 13



Cut value = 12

- Choose *another* random partition of vertices
- Count total # of edges across partition

Symmetric Graph Cut Function



f(A) = sum of weights of edges between A and V\A

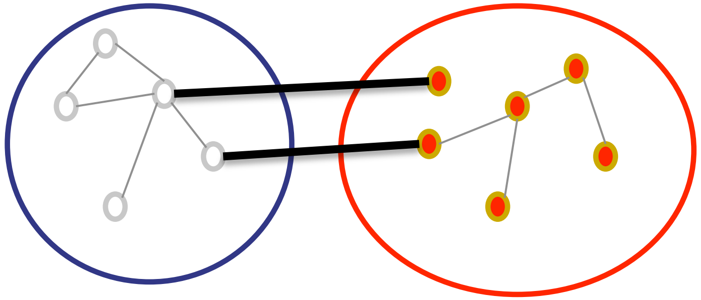
- V = set of vertices
- One-to-one correspondence of graphs and cut functions

Can we learn a graph from the value of few cuts?
[E.g., graph sketching, computational biology, …]

General Problem: Learning Set Functions

Base Set V

Set function $f : 2^V \to \mathbb{R}$



- Wish to learn f from:
  - *Small* measurement collection $\mathcal{M} = \{A_1, \ldots, A_m\}$
  - Function values $f(A_1), \ldots, f(A_m)$

Approximating submodular functions
[Goemans, Harvey, Kleinberg, Mirrokni, '08]

- *Pick* m sets, A_1 … A_m, get to see $F(A_1)$, …, $F(A_m)$
- From this, want to approximate F by $\hat{F}$ s.t.

$$\hat{F}(A) \leq F(A) \leq \alpha \hat{F}(A) \text{ for all } A$$

Theorem: Even if

- F is monotonic
- we can pick polynomially many A_i, chosen adaptively,

cannot approximate better than $\alpha = n^{1/2} / \log(n)$
unless one looks at exponentially many sets A_i

But can efficiently obtain $\alpha = n^{1/2} \log(n)$

Learning submodular functions
[Balcan, Harvey STOC '11]

- *Sample* m sets $A_1 \ldots A_m$, from dist. D; see $F(A_1), \ldots, F(A_m)$
- From this, want to generalize well
- $\hat{F}$ is (α,ε,δ)-PMAC iff with prob. 1-δ it holds that

$$P_{A\sim\mathcal{D}}\left[\hat{F}(A) \le F(A) \le \alpha \hat{F}(A)\right] \ge 1-\varepsilon$$

Theorem: cannot approximate better than

$\alpha = n^{1/3} / \log(n)$

unless one looks at exponentially many samples A_i

But can efficiently obtain $\alpha = n^{1/2}$

What if we have structure?

- To learn effectively, need additional assumptions beyond submodularity.

- Sparsity in Fourier domain [Stobbe & Krause '12]

$$f(A) = \sum_{B \in 2^V} (-1)^{|A \cap B|} \hat{f}(B)$$

Sparsity: Most coefficients ≈0

  - „Submodular“ compressive sensing
  - Cuts and many other functions sparse in Fourier domain!
- Also can learn XOS valuations [Balcan et al '12]

Compressive sensing with set functions

[Stobbe & Krause '12]

Theorem: Suppose number of random measurements is proportional to the sparsity times log factors

$$m = O(k \log^4(p))$$

Then we can recover $\hat{\mathbf{f}}$ by solving:

$$\min ||\mathbf{x}||_1 \text{ s.t. } \mathbf{f}_{\mathcal{M}} = \mathbf{\Phi}_{\mathcal{M}} \mathbf{x}$$

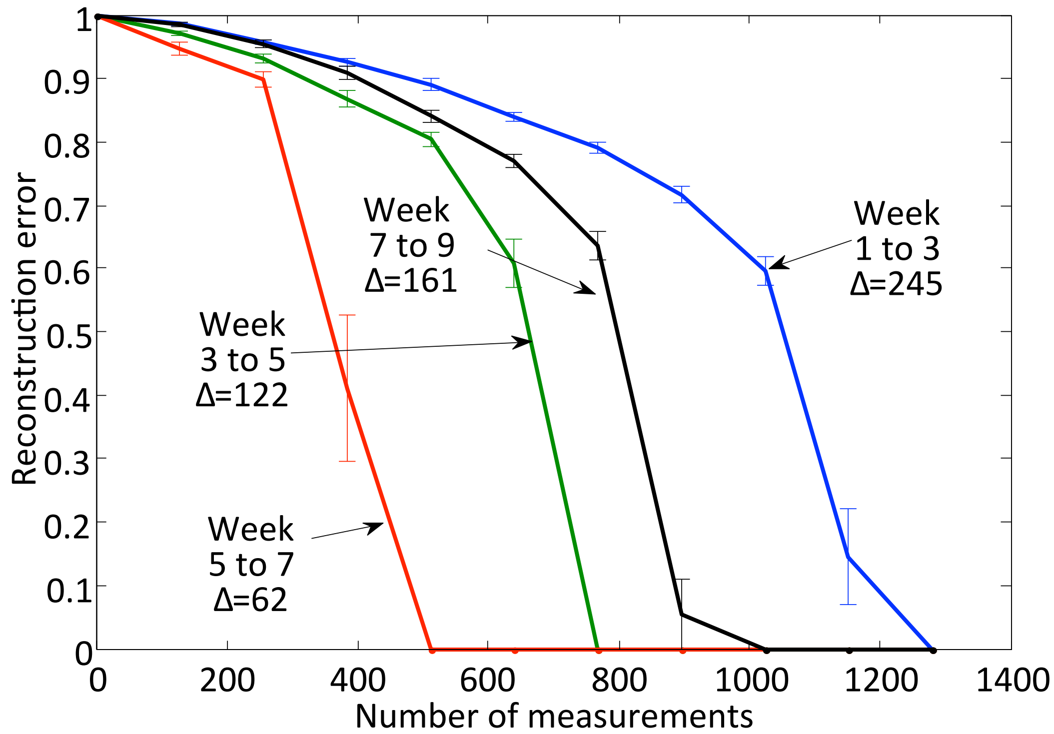
Random Hadamard-Walsh basis functions satisfy RIP w.h.p

Can also handle noisy observations

Many more details [see paper]

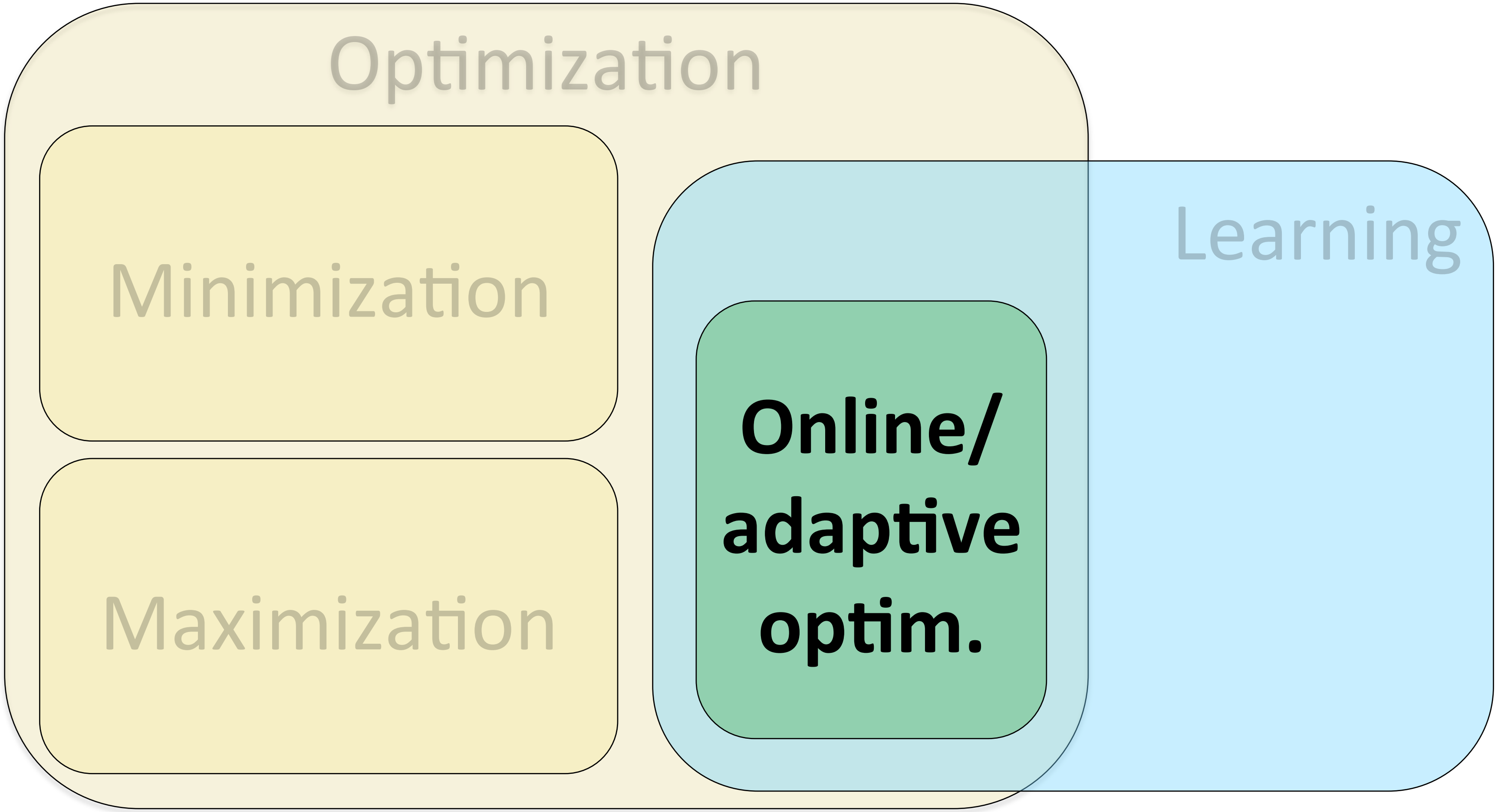
Graph Evolution Results

- Tracking evolution of 128-vertex subgraph using random cuts
- Δ = number of differences between graphs



- Autonomous Systems Graph (from SNAP)
- For low error, observing $m \approx 8\Delta$ random cuts suffices

What to do with submodular functions



Learning to optimize

- Have seen how to
  - optimize submodular functions
  - learn submodular functions

What if we only want to
learn *enough* to optimize?

Learning to optimize submodular functions

- Structured prediction with submodular functions
  - Learn function parameters to achieve target minimizer
  - *Application*: MAP inference; summarization

- Online submodular optimization
  - Learn to pick a sequence of sets to maximize a sequence of (unknown) submodular functions
  - *Application*: Building algorithm portfolios

- Adaptive submodular optimization
  - Gradually build up a set, taking into account feedback
  - *Application*: Sequential experimental design

Structured Prediction with SFs

- Given training data $\{(S_1, \mathbf{x}_1), \ldots, (S_n, \mathbf{x}_n)\}$ and parameterized family of SFs F, can we find parameters θ such that

$$\boxed{S_i \approx \arg\min_{S} F(S; \theta, \mathbf{x}_i)}$$

With margin!

- Approaches
  - Parameter learning in submodular MRFs [Taskar et al. '04] ➔ Minimization
  - Learning mixtures of submodular shells [Lin & Bilmes '12] ➔ Maximization

Learning to summarize [Lin & Bilmes'12]

- Input:
  - Collection of documents d_k and human summaries S_k
- Goal:
  - Learn parameterized submodular relevance & diversity

$$F(S; d_k) = \sum_i \alpha_i R_i(S; d_k) + \sum_j \beta_j D_j(S; d_k)$$

Relevance & diversity instantiated for each document k

Parameters, want to learn

  - Want to achieve that $S_k \approx \operatorname*{argmax}_{S:|S|\leq B} F(S; d_k)$ for all k

- Can efficiently find solution with bounded generalization error!

Learning mixtures for summarization

DUC-07	R	F
Toutanova et al. [53]	11.89	11.89
Haghighi and Vanderwende [16]	11.80	-
Celikyilmaz and Hakkani-tür [4]	11.40	-
Lin and Bilmes [28]	12.38	12.33
Best system in DUC-07 (peer 15)	12.45	12.29
Submodular Shell Mixture	**12.51**	**12.40**

Learning mixtures for summarization

- Training data: Documents with human summaries

Learning to optimize submodular functions

- Structured prediction with submodular functions
  - Learn function parameters to achieve target minimizer
  - *Application*: MAP inference; summarization

- Online submodular optimization
  - Learn to pick a sequence of sets to maximize a sequence of (unknown) submodular functions
  - *Application*: Building algorithm portfolios

- Adaptive submodular optimization
  - Gradually build up a set, taking into account feedback
  - *Application*: Sequential experimental design

Online maximization of submodular functions
[Streeter, Golovin NIPS '08]

Pick sets	A_1	A_2	A_3	…	A_T
SFs	F_1	F_2	F_3	…	F_T
Reward	$r_1=F_1(A_1)$	r_2	r_3	…	r_T

Observe *either* F_t, or only $F_t(A_t)$

Total: $\sum_t r_t \rightarrow \max$

Time

Theorem
Can efficiently choose $A_1,\ldots A_t$ s.t. in expectation

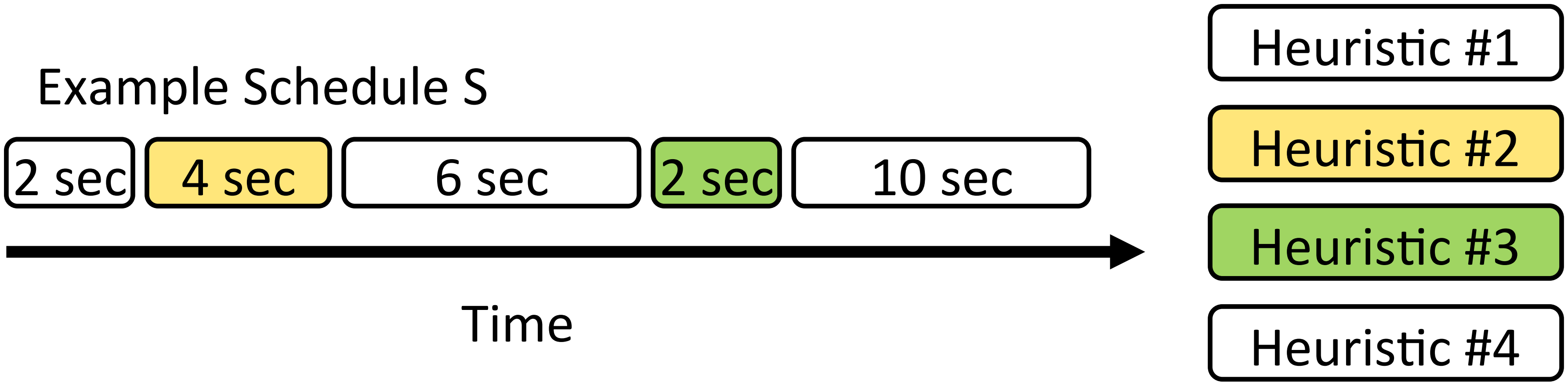
$$\frac{1}{T}\sum_{t=1}^{T} F_t(A_t) \geq \frac{1-1/e}{T} \max_{|A|\leq k} \sum_{t=1}^{T} F_t(A)$$

for any sequence F_i, as $T \rightarrow \infty$

Can get 'no-regret' over 'omniscient' greedy algorithm

Application: Learning to solve SAT quickly
[Streeter, Golovin, Smith]

- Hybridization via *Task-Switching Schedules*



- **Need to learn which schedules work well!**

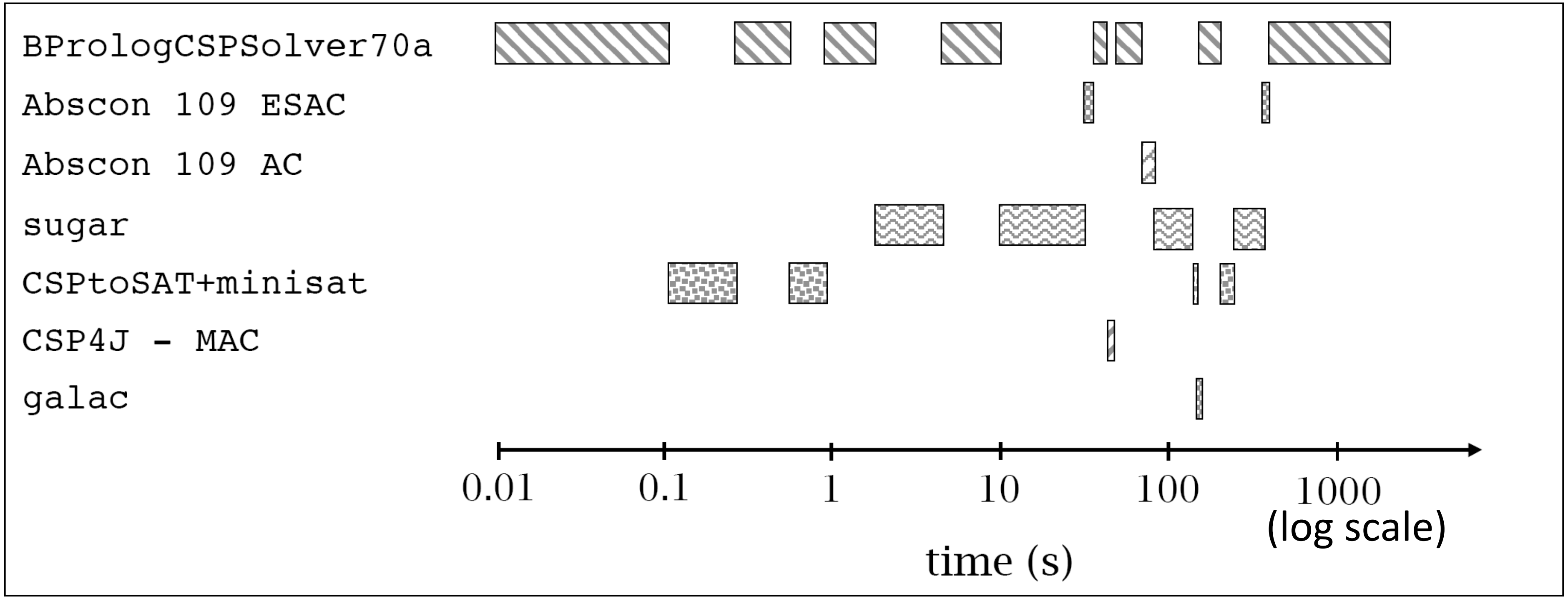
Hybridizing Solvers

[Streeter & Golovin '08]

- V = {run heuristic h for t seconds : h in H, t >0}
- $f_i(S)$ = Pr[schedule S completes job f in time limit].
  - This is a submodular function! ☺

- Task: Select $\{S_i$: i =1, 2, ..., T$\}$ online to maximize # instances solved
- **This is an online submodular maximization problem! ☺**
- **Online greedy algorithm achieves no-(1-1/e)-regret**

Example Schedule [Streeter, Golovin, Smith, AAAI '07 & CSP '08]



SAT 2007 Competition Data

Number of benchmark instances solved within the time limit.

Category	Offline greedy	Online greedy	Parallel schedule	Top solver
Industrial	147	149	132	139
Random	350	347	302	257
Hand-crafted	114	107	95	98

The CADE ATP System Competition (2008)

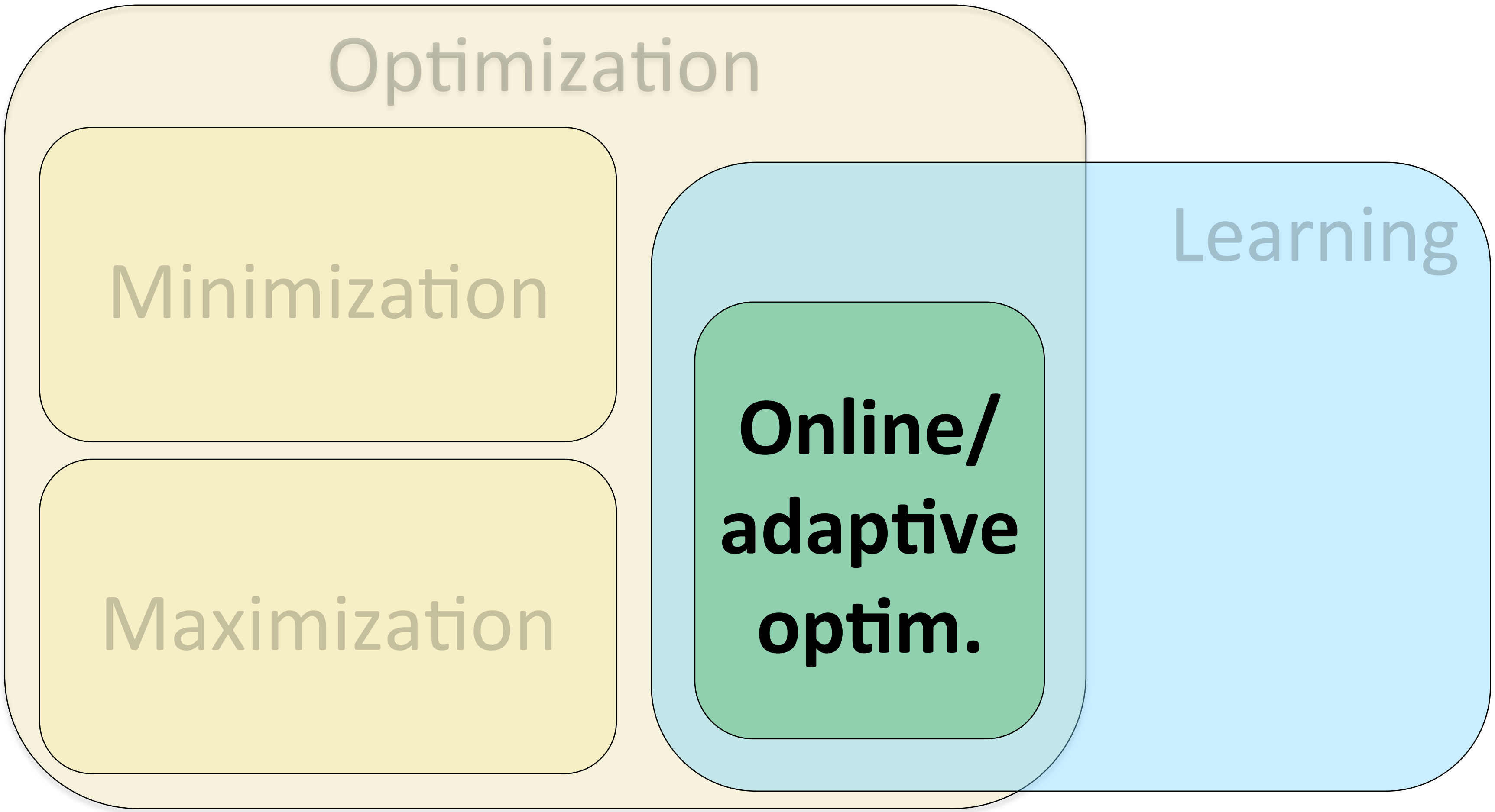
Percentage of benchmark instances solved within the time limit.

Category	MetaProver	2nd best	3rd best
FNT	74%	70%	70%
SAT	100%	97.5%	96.3%

Other results on online submodular optimization

- Online submodular maximization
  - No (1-1/e) regret for ranking (partition matroids) [Streeter, Golovin, Krause 2009]
  - Distributed implementation [Golovin, Faulkner, Krause '2010]
  - Improved bounds for bandits with linear combinations of SFs [Yue, Guestrin, NIPS 2011]
- Online submodular coverage
  - Min-cost / Min-sum submodular cover [Streeter & Golovin NIPS 2008]
  - Guillory & Bilmes [NIPS 2011]
- Online Submodular Minimization
  - Unconstrained [Hazan & Kale NIPS 2009]
  - Constrained [Jegelka & Bilmes ICML 2011]

What to do with submodular functions



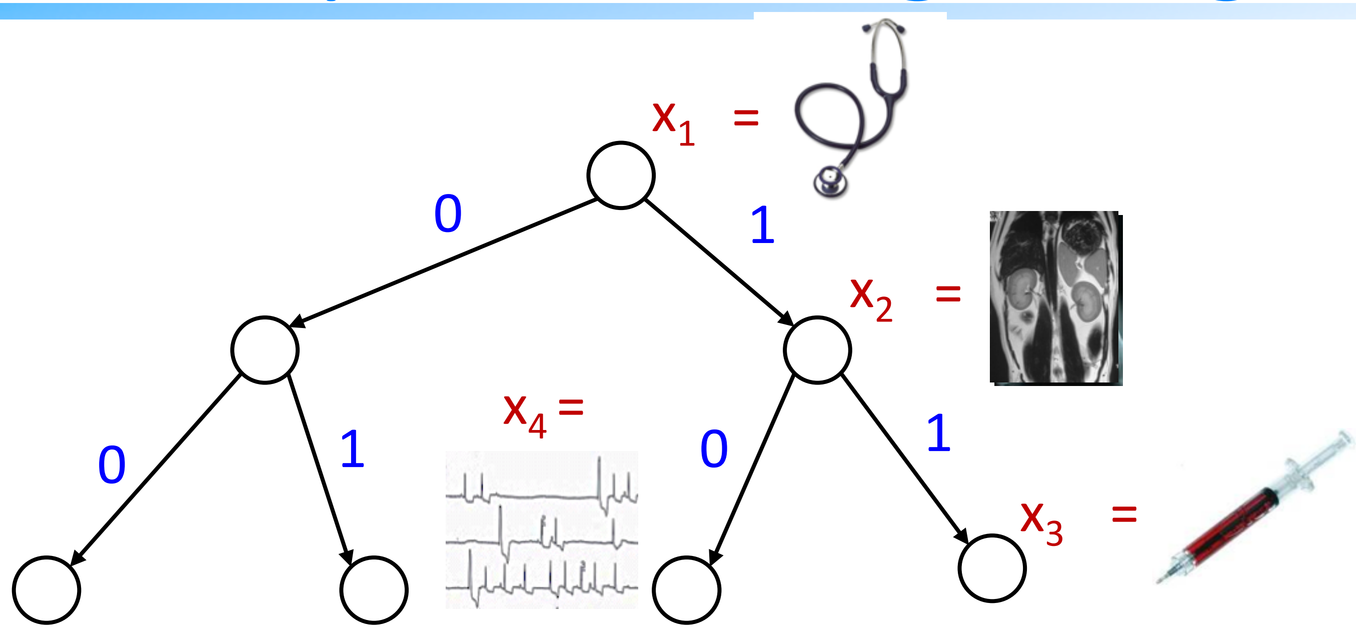
Learning to optimize submodular functions

- Structured prediction with submodular functions
  - Learn function parameters to achieve target minimizer
  - *Application*: MAP inference; summarization

- Online submodular optimization
  - Learn to pick a sequence of sets to maximize a sequence of (unknown) submodular functions
  - *Application*: Building algorithm portfolios

- Adaptive submodular optimization
  - Gradually build up a set, taking into account feedback
  - *Application*: Sequential experimental design

Adaptive Sensing / Diagnosis



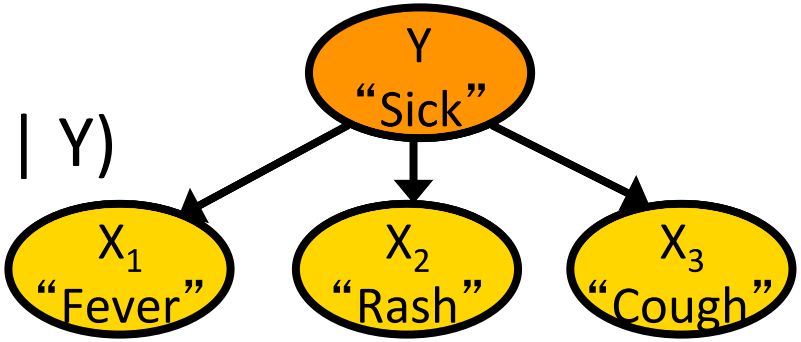
Want to effectively diagnose while minimizing cost of testing!

Classical submodularity does not apply ☹

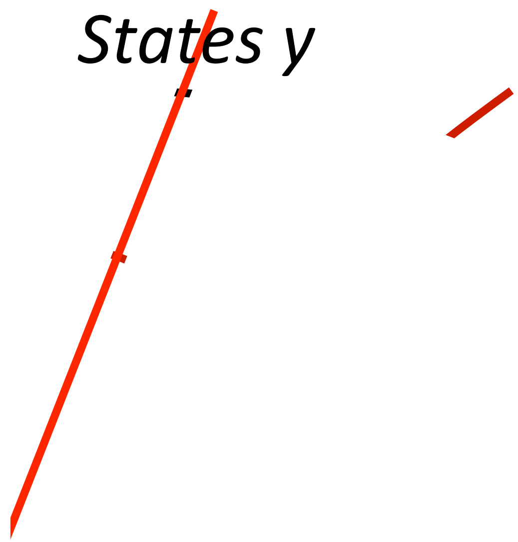
Can we generalize submodularity for sequential decision making?

Adaptive selection in diagnosis

- Prior over diseases P(Y)
- Deterministic test outcomes P($\mathbf{X}_V$ | Y)



- Each test eliminates hypotheses y



Problem Statement

Given:

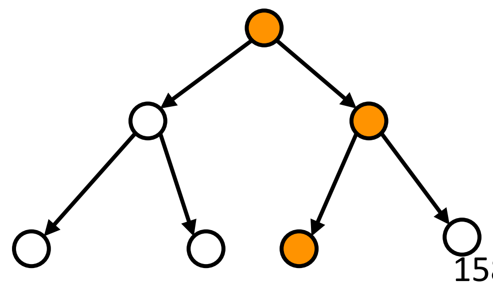
- Items (tests, experiments, actions, …) V={1,…,n}
- Associated with random variables $X_1,\ldots,X_n$ taking values in O
- Objective: $f : 2^V \times O^V \to \mathbb{R}$
- Policy π maps observation $\mathbf{x}_A$ to next item

Value of policy π: $$F(\pi) = \sum_{\mathbf{x}_V} P(\mathbf{x}_V) f(\pi(\mathbf{x}_V), \mathbf{x}_V)$$

Tests run by π
if world in state $\mathbf{x}_V$

Want $\pi^* \in \operatorname{argmax}_{|\pi| \leq k} F(\pi)$

NP-hard (also hard to approximate!)



Adaptive greedy algorithm

- Suppose we've seen $X_A = x_A$.
- Conditional expected benefit of adding item s:

$$\Delta(s \mid \mathbf{x}_A) = \mathbb{E}\Big[\underbrace{f(A \cup \{s\}, \mathbf{x}_V) - f(A, \mathbf{x}_V)}_{\text{Benefit if world in state } \mathbf{x}_V} \mid \mathbf{x}_A\Big]$$

Conditional on observations $\mathbf{x}_A$

<u>Adaptive Greedy algorithm</u>:

Start with $A = \emptyset$

For i = 1:k

- Pick $s_k \in \operatorname{argmax}_s \Delta(s \mid \mathbf{x}_A)$
- Observe $X_{s_k} = x_{s_k}$
- Set $A \leftarrow A \cup \{s_k\}$

When does this adaptive greedy algorithm work??

Adaptive submodularity
[Golovin & Krause, JAIR 2011]

Adaptive monotonicity:

$$\Delta(s \mid \mathbf{x}_A) \geq 0$$

x_B observes more than x_A

Adaptive submodularity:

$$\Delta(s \mid \mathbf{x}_A) \geq \Delta(s \mid \mathbf{x}_B) \quad \text{whenever} \quad \mathbf{x}_A \preceq \mathbf{x}_B$$

Theorem: If f is adaptive submodular and adaptive monotone w.r.t. to distribution P, then

$F(\pi_{greedy}) \geq (1\text{-}1/e)\ F(\pi_{opt})$

Many other results about submodular set functions can also be "lifted" to the adaptive setting!

From sets to policies

Submodularity ⇨ **Adaptive submodularity**

Applies to: set functions | policies, value functions

$$\Delta_F(s \mid A) = F(A \cup \{s\}) - F(A)$$

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

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

$$\max_A F(A)$$

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

$$\Delta_F(s \mid \mathbf{x}_A) = \mathbb{E}\Big[f(A \cup \{s\}, \mathbf{x}_V) - f(A, \mathbf{x}_V) \mid \mathbf{x}_A\Big]$$

$$\Delta_F(s \mid \mathbf{x}_A) \geq 0$$

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

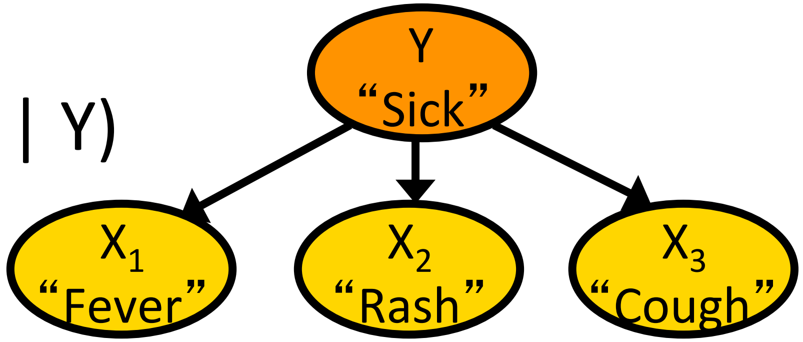
$$\max_\pi F(\pi)$$

Greedy 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

Optimal Diagnosis

- Prior over diseases P(Y)
- Deterministic test outcomes P($\mathbf{X}_V$ | Y)

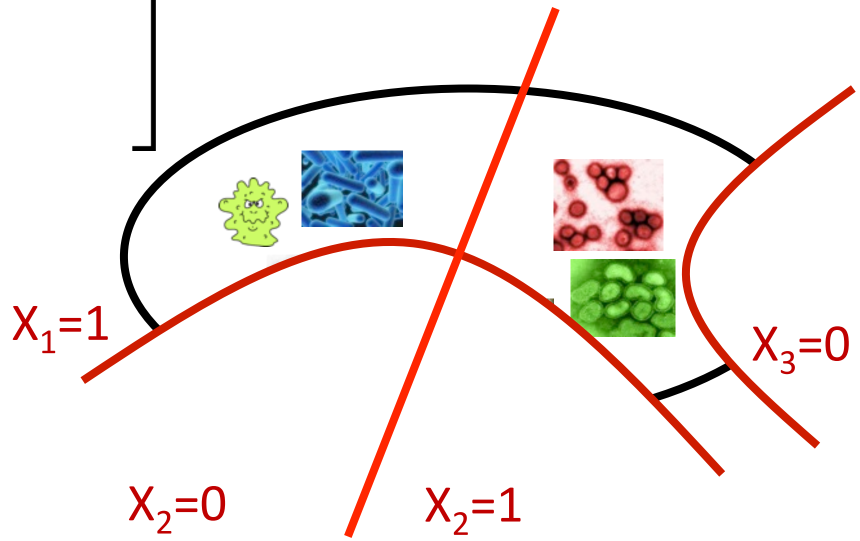


- How should we test to eliminate all incorrect hypotheses?

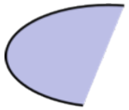
$$\Delta(t \mid x_A) = \mathbb{E}\begin{bmatrix}\text{mass ruled out} \\ \text{by } t \text{ if we} \\ \text{know } x_A\end{bmatrix}$$

“Generalized binary search”

Equivalent to max. infogain



OD is Adaptive Submodular

$b_0 := \mathbb{P}($  $)$

$g_0 := \mathbb{P}($  $)$

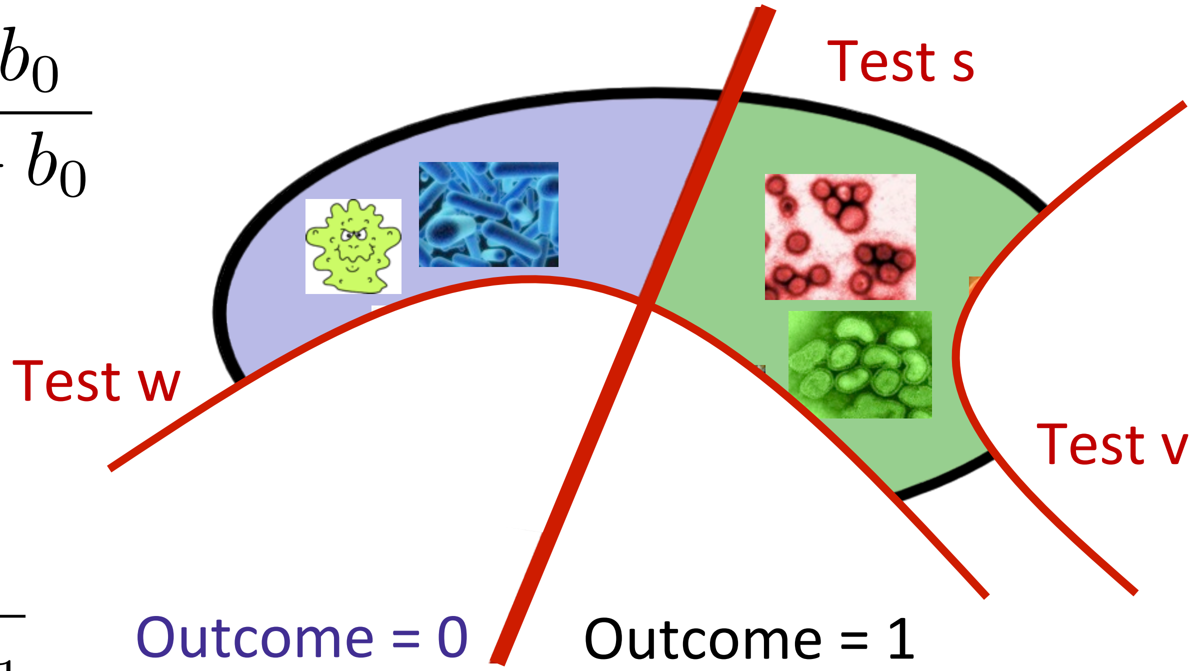
$$\Delta(s \mid \{\}) = \frac{2 g_0 b_0}{g_0 + b_0}$$

$b_1 := \mathbb{P}($  $)$

$g_1 := \mathbb{P}($  $)$

$$\Delta(s \mid \mathbf{x}_{v,w}) = \frac{2 g_1 b_1}{g_1 + b_1}$$

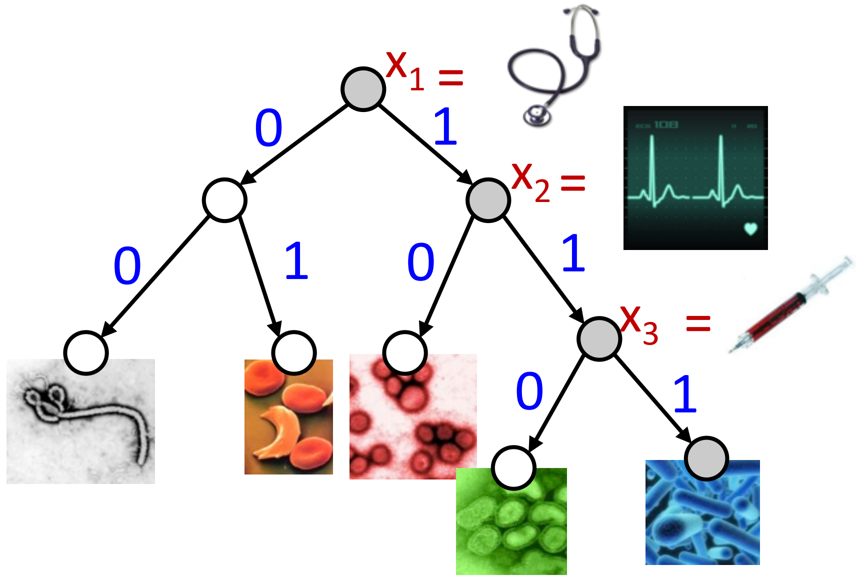
Objective = probability mass of hypotheses you have ruled out.



$$b_0 \geq b_1, \quad g_0 \geq g_1$$

Not hard to show that $\Delta(s \mid \{\}) \geq \Delta(s \mid \mathbf{x}_{v,w})$

Theoretical guarantees



Garey & Graham, 1974;
Loveland, 1985;
Arkin et al., 1993;
Kosaraju et al., 1999;
Dasgupta, 2004;
Guillory & Bilmes, 2009;
Nowak, 2009;
Gupta et al., 2010

Adaptive-Greedy is a $\boxed{(\ln(1/p_{\min}) + 1)}$ approximation.

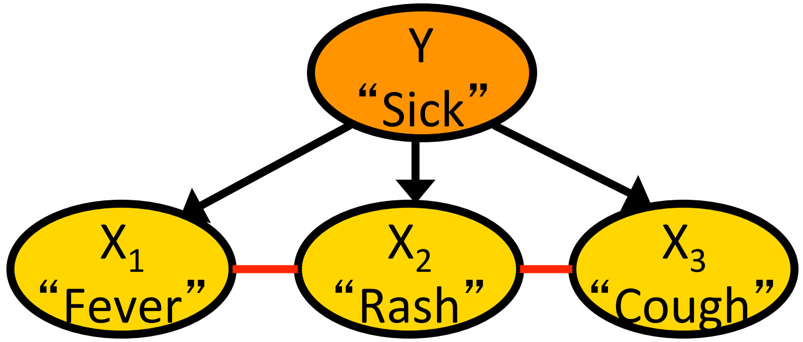
With adaptive submodular analysis!

Result requires that tests are *exact* (no noise)!

What if there is noise?

[w Daniel Golovin, Deb Ray, NIPS '10]

- Prior over diseases P(Y)
- Noisy test outcomes $P(\mathbf{X}_V \mid Y)$
- How should we test to learn about y (infer MAP)?



- Existing approaches:
  - Generalized binary search?
  - Maximize information gain?
  - Maximize value of information?

Not adaptive submodular!

Theorem: All these approaches can have cost **more than n/log n** times the optimal cost!

➔ Is there an adaptive submodular criterion??

Theoretical guarantees
[with Daniel Golovin, Deb Ray, NIPS '10]

Theorem: Equivalence class edge-cutting (EC^2) is adaptive monotone and adaptive submodular.
Suppose $P(\mathbf{x}_V, h) \in \{0\} \cup [\delta, 1]$ for all $\mathbf{x}_V, h$
Then it holds that

$$\mathrm{Cost}(\pi_{\mathrm{Greedy}}) \leq \mathcal{O}\left(\log \frac{1}{\delta}\right) \mathrm{Cost}(\pi^*)$$

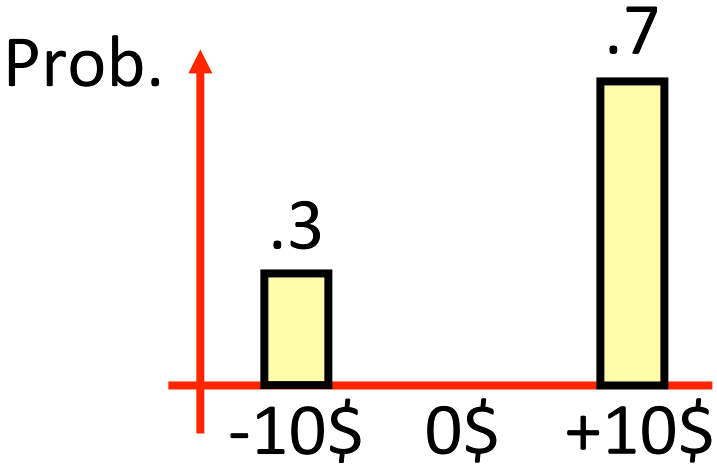
First approximation guarantees for **nonmyopic VOI** in general graphical models!

Example: The Iowa Gambling Task

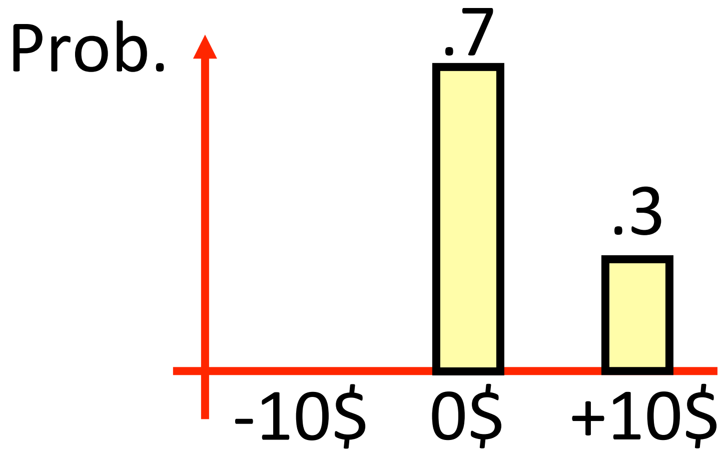
[with Colin Camerer, Deb Ray]

What would you prefer?

A)



B)



Various competing theories on how people make decisions under uncertainty

- Maximize expected utility? [von Neumann & Morgenstern '47]
- Constant relative risk aversion? [Pratt '64]
- Portfolio optimization? [Hanoch & Levy '70]
- (Normalized) Prospect theory? [Kahnemann & Tversky '79]

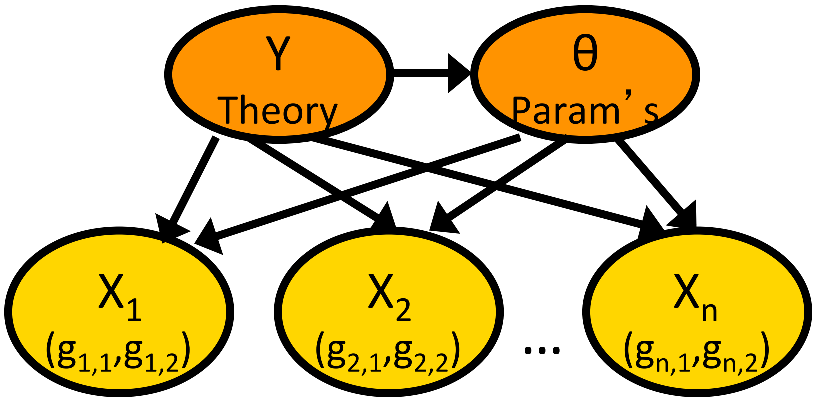
How should we design tests to distinguish theories?

Iowa Gambling as BED

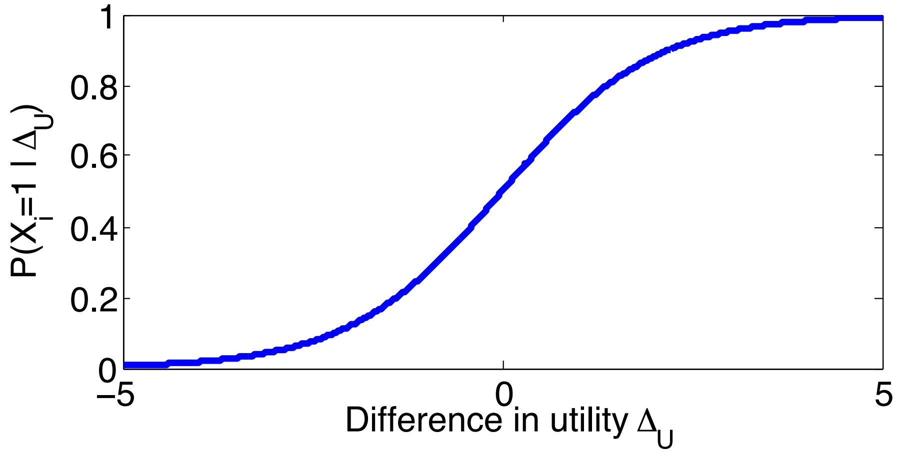
Every possible test $X_s = (g_{s,1}, g_{s,2})$ is a pair of gambles

Theories parameterized by θ

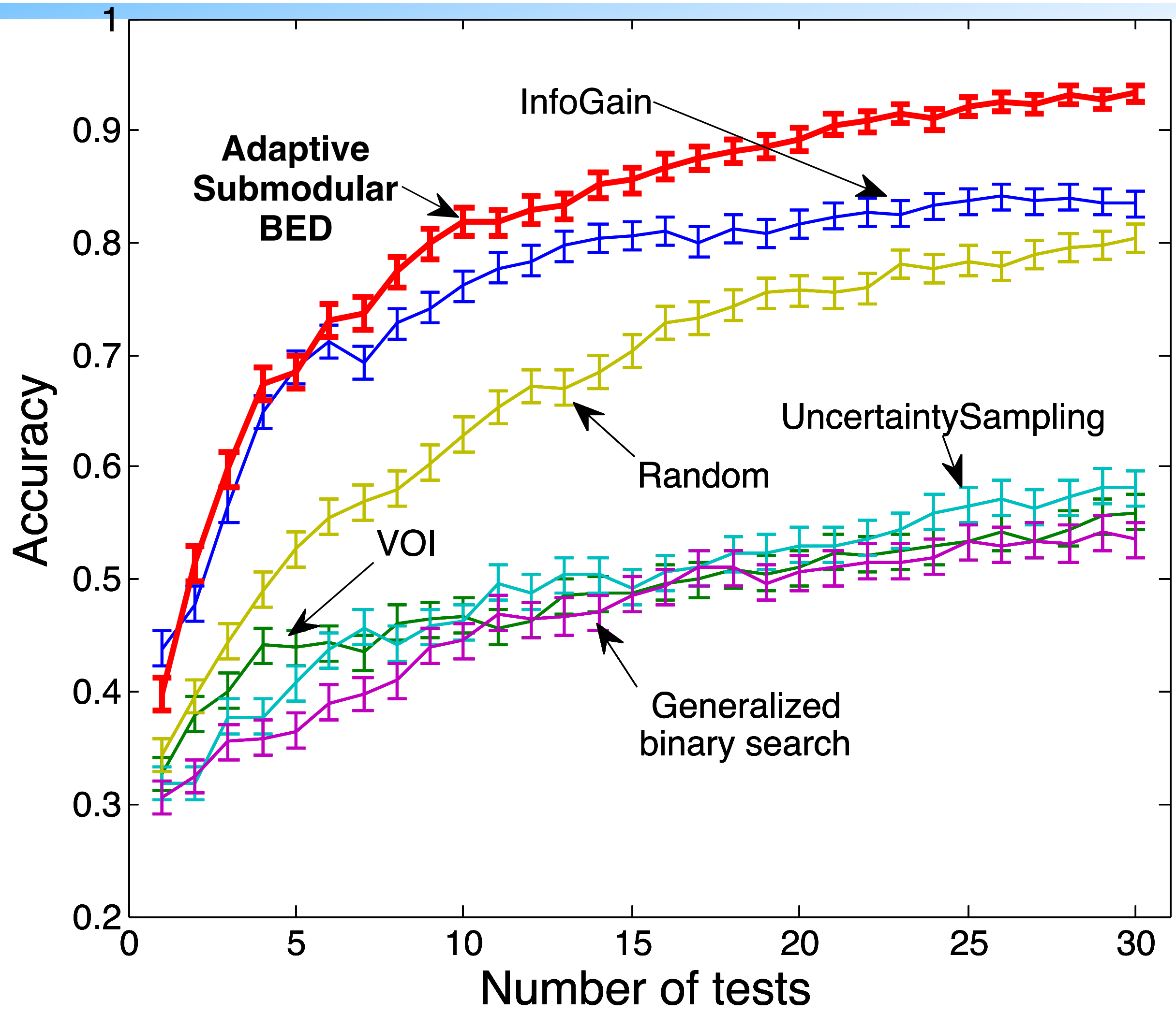
Each theory predicts utility for every gamble $U(g,y,\theta)$



$$P(X_s = 1 \mid y, \theta) = \frac{1}{1 + \exp(U(g_{s,1}, y, \theta) - U(g_{s,2}, y, \theta))}$$

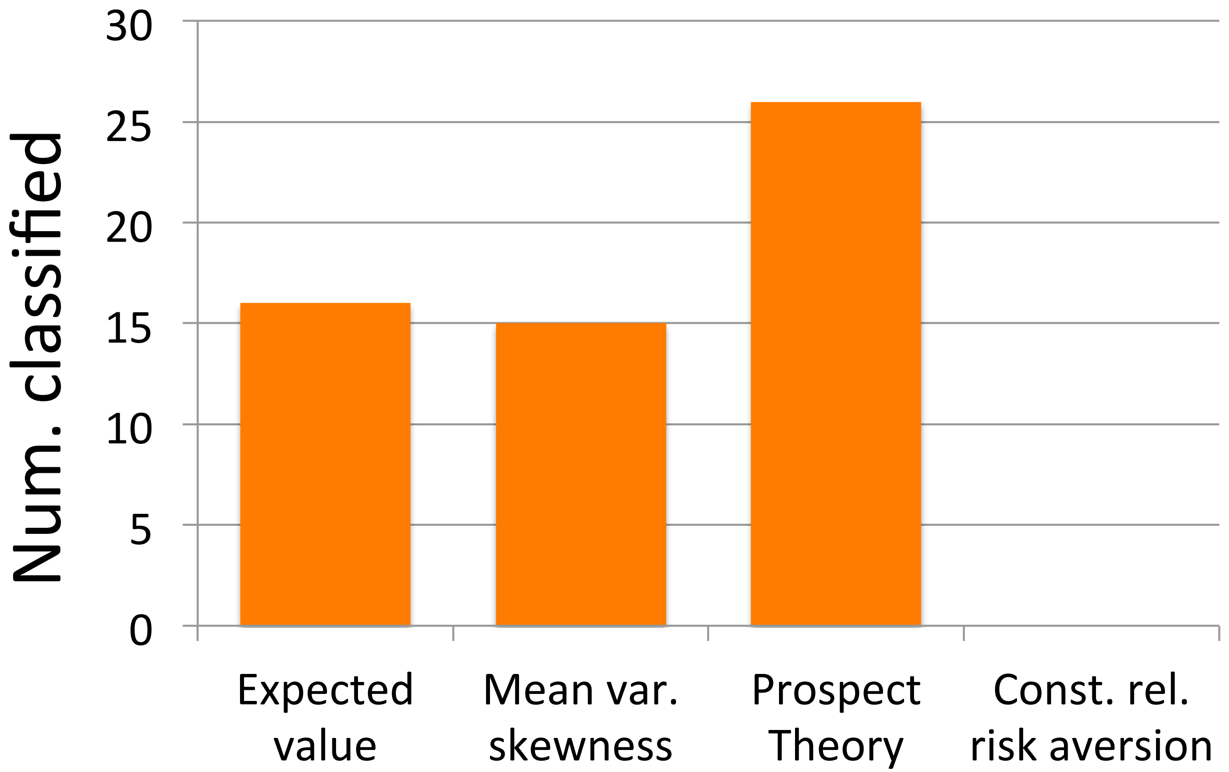


Simulation Results



Adaptive submodular criterion (EC^2) outperforms existing approaches

Experimental Study
[with Colin Camerer, Deb Ray]



Study with 57 naïve subjects

32,000 designs

40s per test ☹

Exploiting submodularity:

<5s per test ☺

- Strongest support for PT, with some heterogeneity
- Unexpectedly **no support** for CRRA
- Submodularity enables real-time performance!

Interactive submodular coverage

- Alternative formalization of adaptive optimization [Guillory & Bilmes, ICML '10]
  - Addresses the worst case setting
- Applications to (noisy) active learning, viral marketing [Guillory & Bilmes, ICML '11]

Other directions

- Game theory
  - Equilibria in cooperative (supermodular) games / fair allocations
  - Price of anarchy in non-cooperative games
  - Incentive compatible submodular optimization
- New algorithms for submodular maximization
  - Robust submodular optimization
- Generalizations of submodular functions
  - L#-convex / discrete convex analysis
  - XOS/Subadditive functions
- Efficient minimization of subclasses

Further resources

- submodularity.org
  - References
  - Matlab Toolbox for Submodular Optimization
- discml.cc
  - NIPS Workshops on Discrete Optimization in Machine Learning
  - Videos of invited talks on videolectures.net

...

- **Submit to DISCML 2012! ☺**

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
- Can handle complex constraints
- Can learn to optimize (online, adaptive, …)