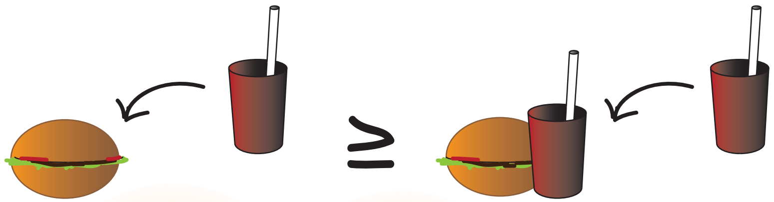
# Outline

- What is submodularity?
- Optimization
  - Minimize costs



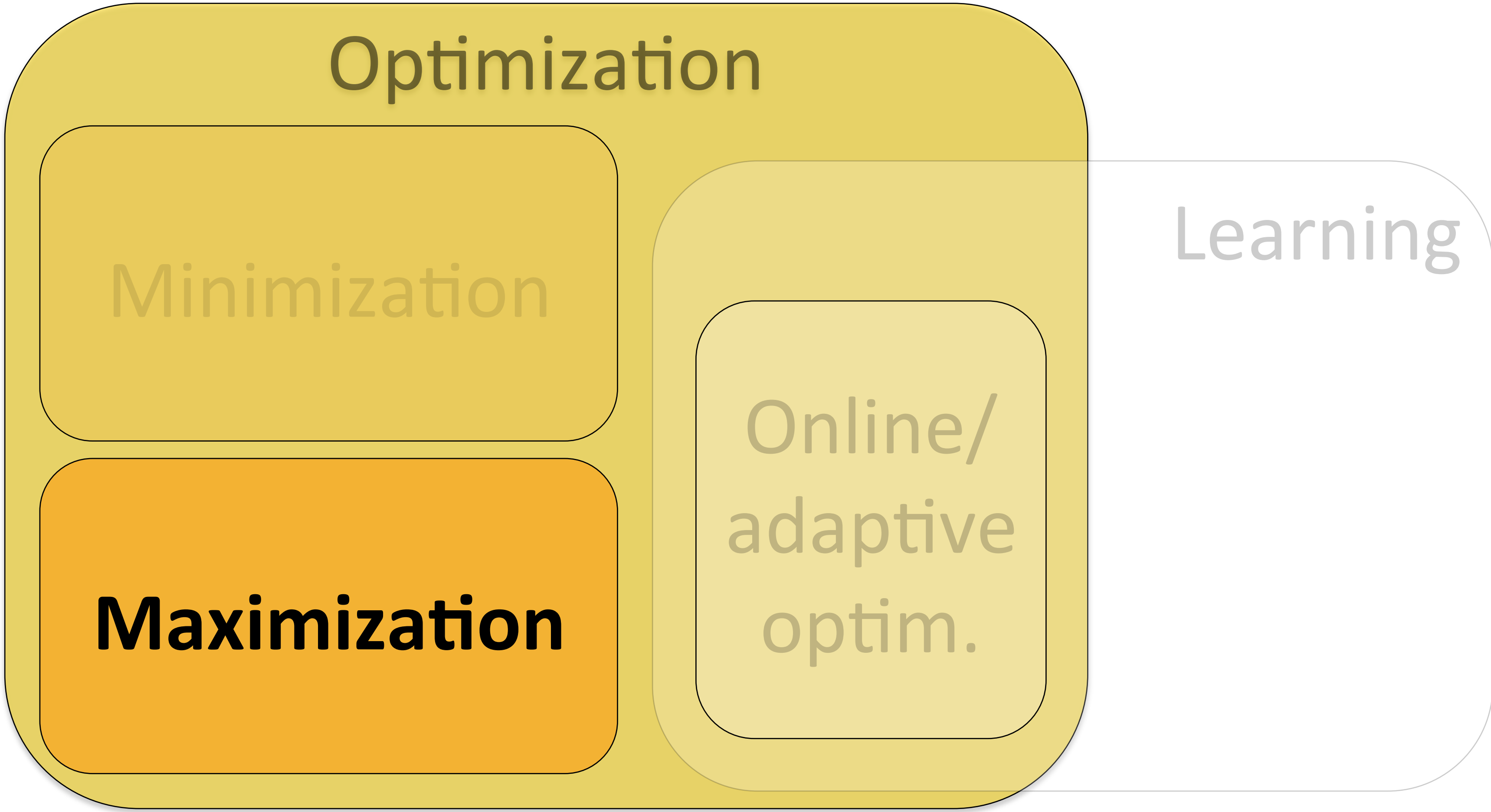
  - Maximize utility

- Learning
- Learning for Optimization: new settings

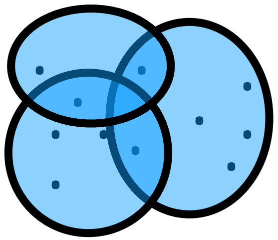
Part I

Part II

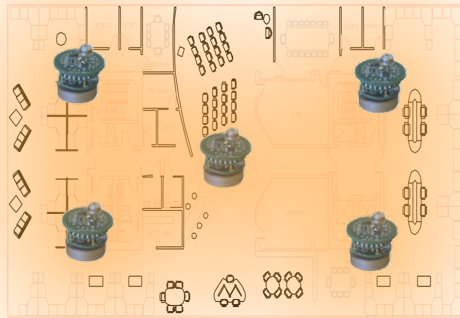
# Optimization



# Submodular maximization



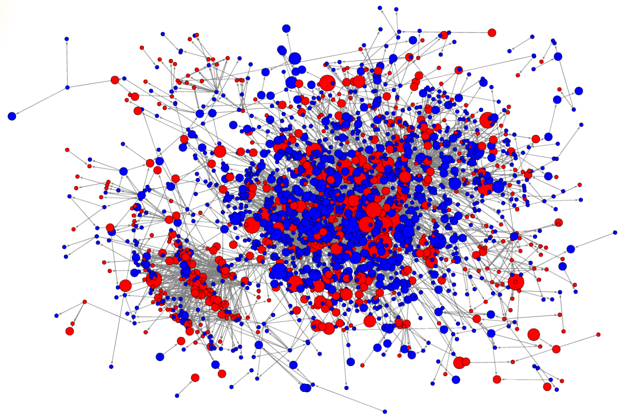
covering



sensing

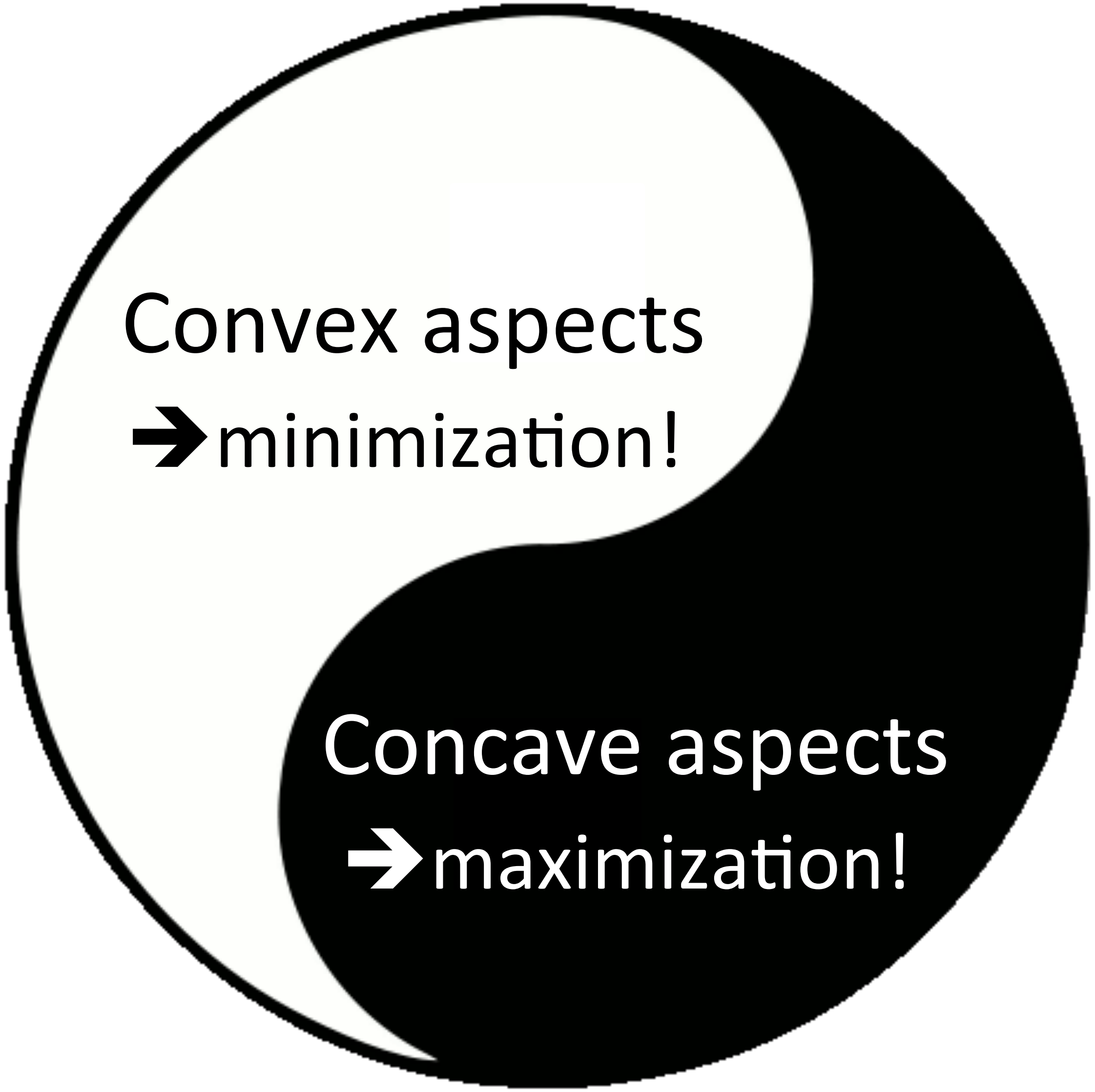
$$\max_{S\subseteq V} F(S)$$

summarization



network inference

# Two faces of submodular functions



# Submodular maximization

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

➔ submodularity and concavity

# Concave aspects

- submodularity:

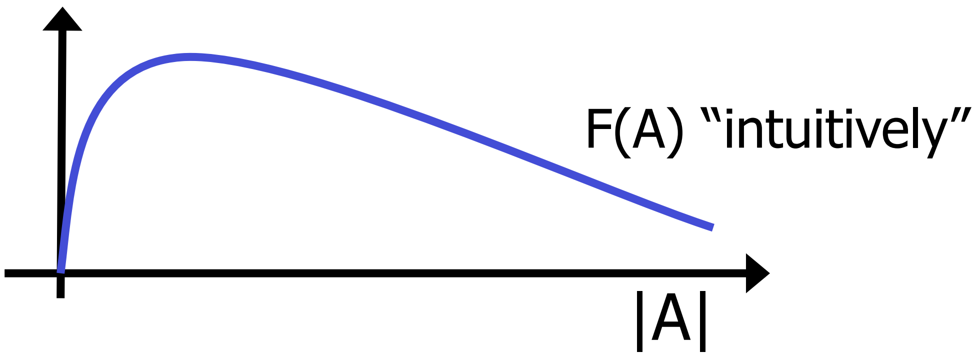
$$A \subseteq B, \;\; s \notin B :$$

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

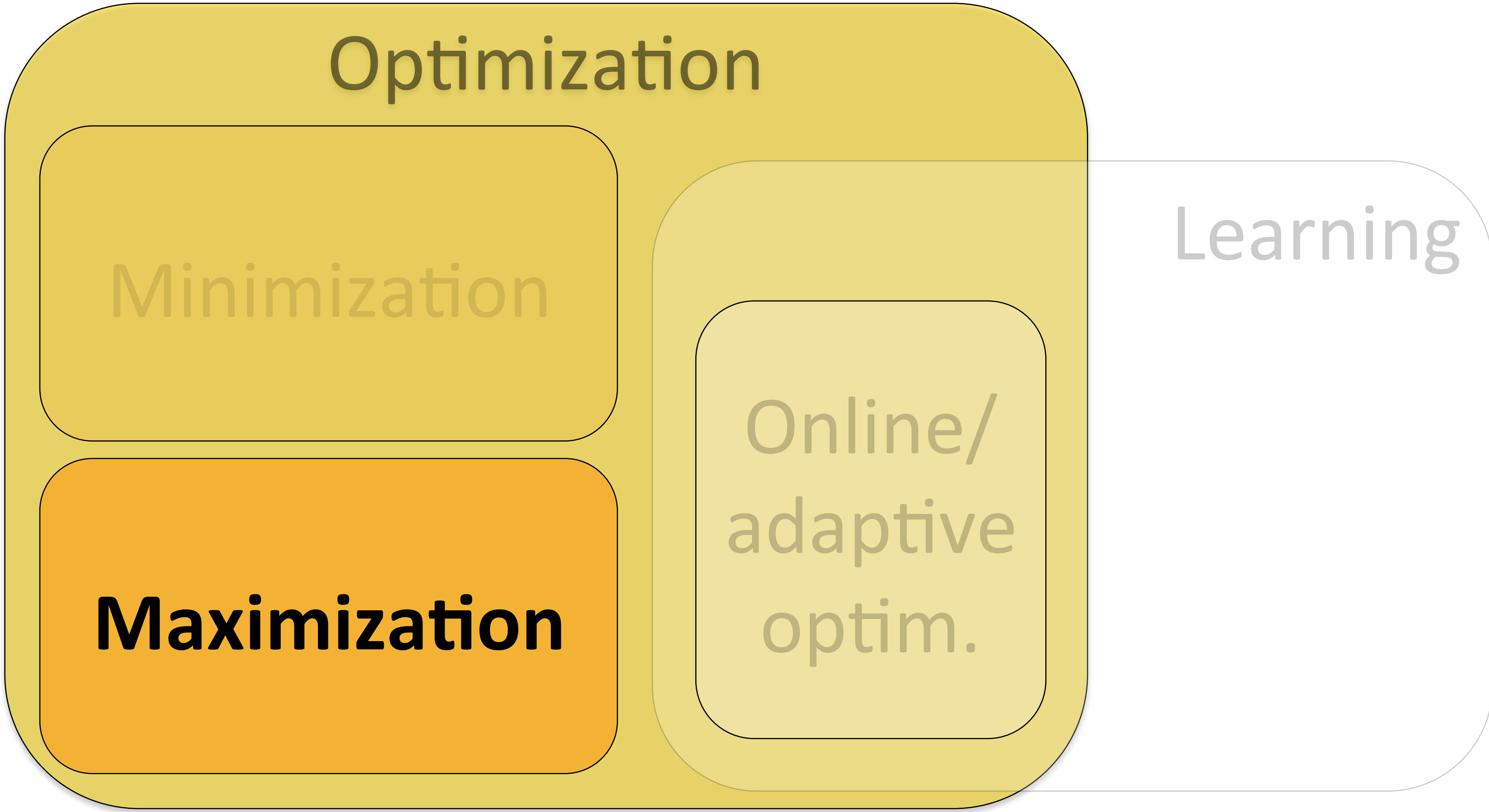
- concavity:

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

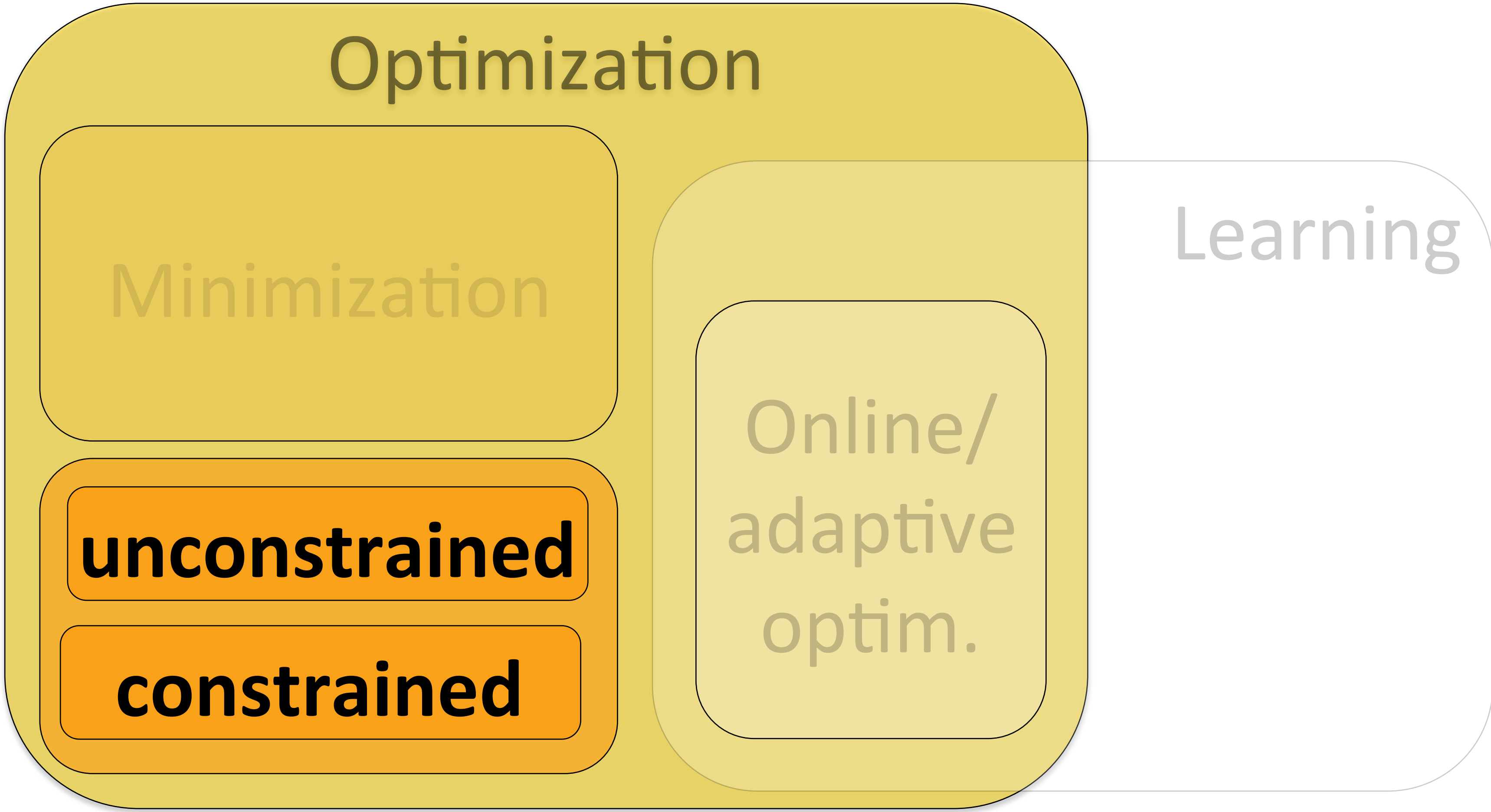
$$f(a+s) - f(a) \quad \geq \quad f(b+s) - f(b)$$



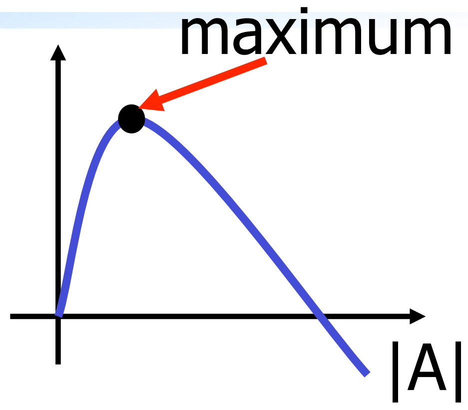
# Optimization



# 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)

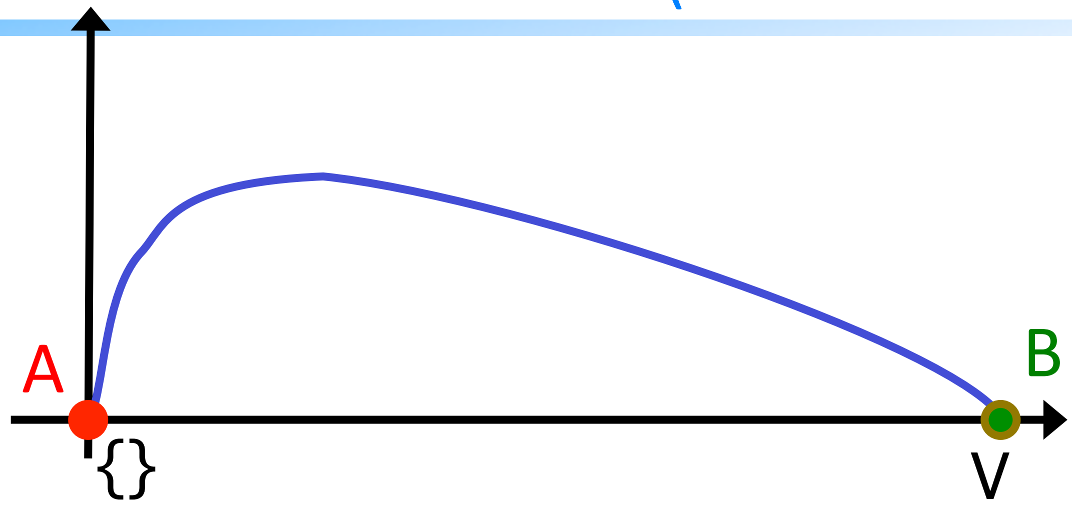
# Exact maximization of SFs

- 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!

# Randomized USM (Buchbinder et al '12)




Start with A={}, B=V

For i=1 to n

$$v_+ = \max\left(F(A \cup \{s_i\}) - F(A), 0\right)$$
$$v_- = \max\left(F(B \setminus \{s_i\}) - F(B), 0\right)$$

Pick $U \sim \text{Unif}([0,1])$

If $U \leq v_+/(v_+ + v_-)$ set $A \leftarrow A \cup \{s_i\}$

Else $B \leftarrow B \setminus \{s_i\}$

Return $A \ (= B)$

# 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:

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

"Scalarization"

Option 2:

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

"Constrained maximization"

Can get 1/2 approx...

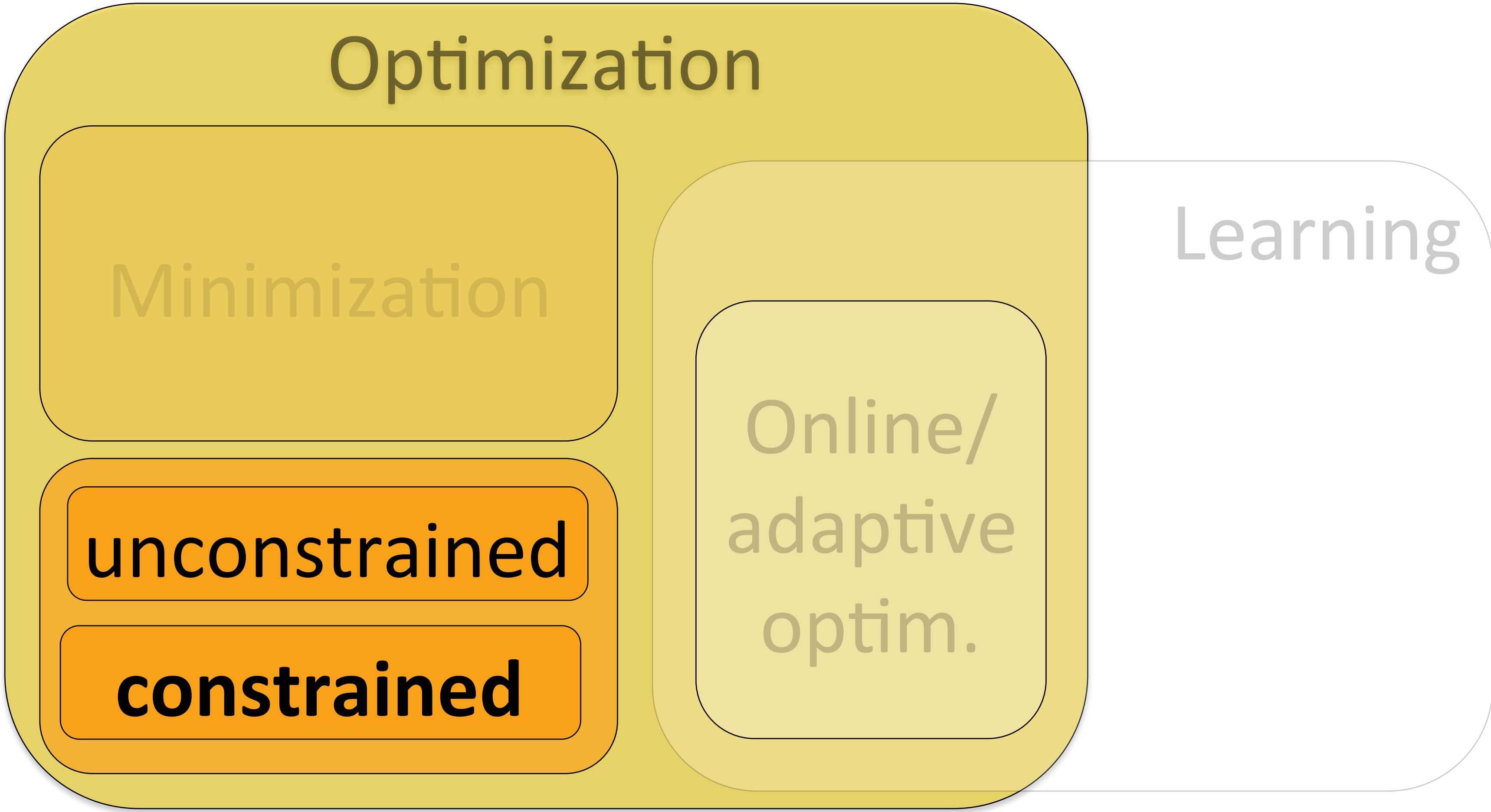
if F(A)-C(A) ≥ 0

for all sets A

What is possible?

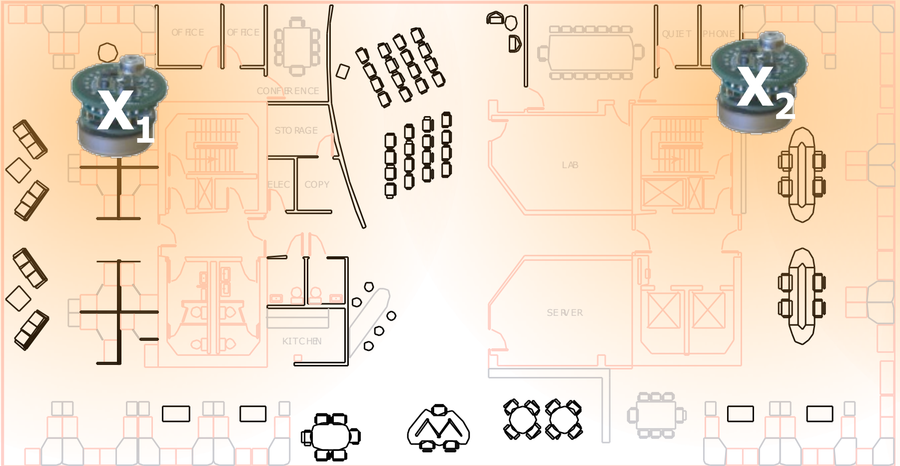
Positiveness is a strong requirement ☹

# Optimization

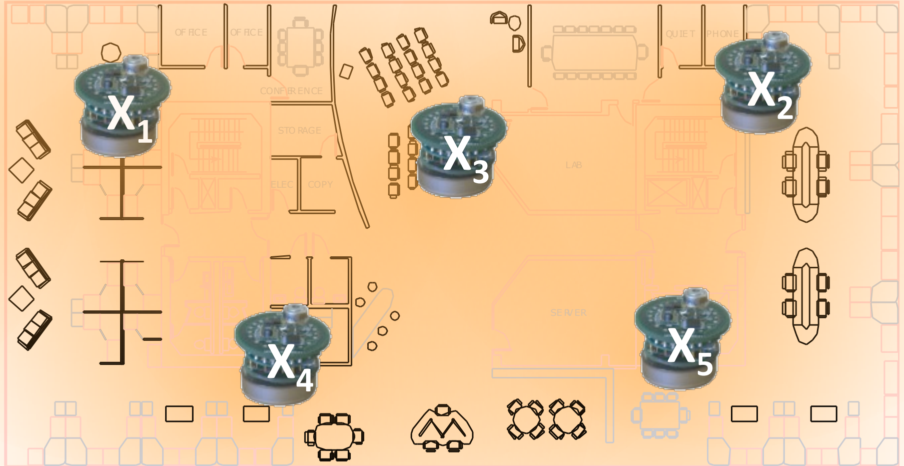


# Monotonicity

Placement A = {1,2}




Placement B = {1,…,5}



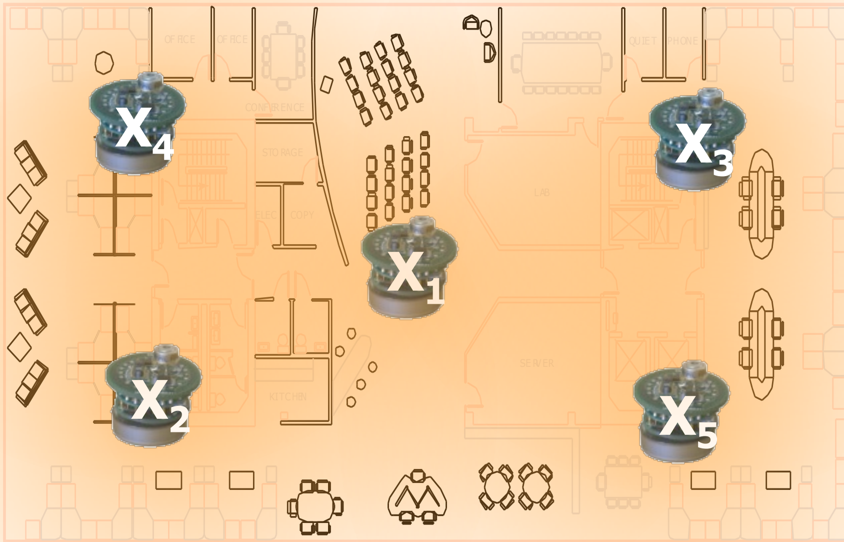

F is monotonic: $\forall A, s : \underbrace{F(A \cup \{s\}) - F(A)}_{\Delta(s \mid A) \geq 0} \geq 0$

*Adding sensors can only help*

# *Cardinality* constrained maximization

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

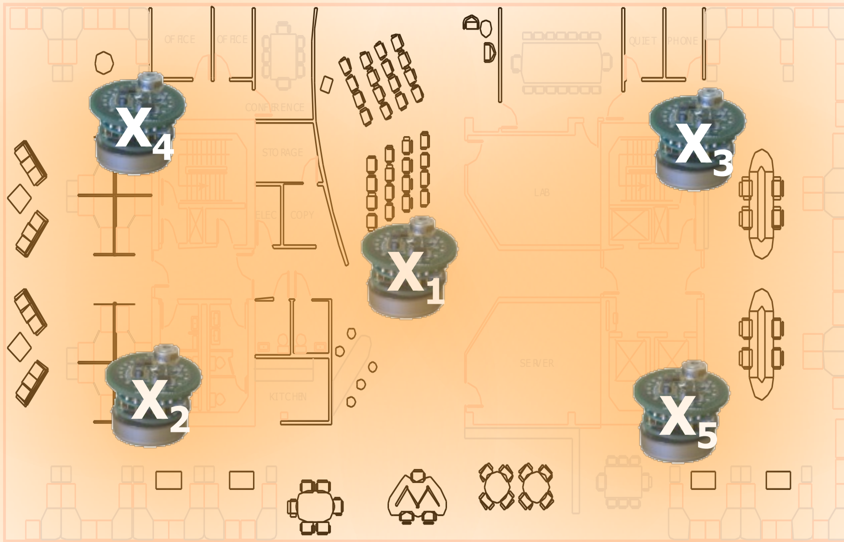
**NP-hard!**



# Greedy algorithm

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

**NP-hard!**




<u>Greedy algorithm:</u>

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

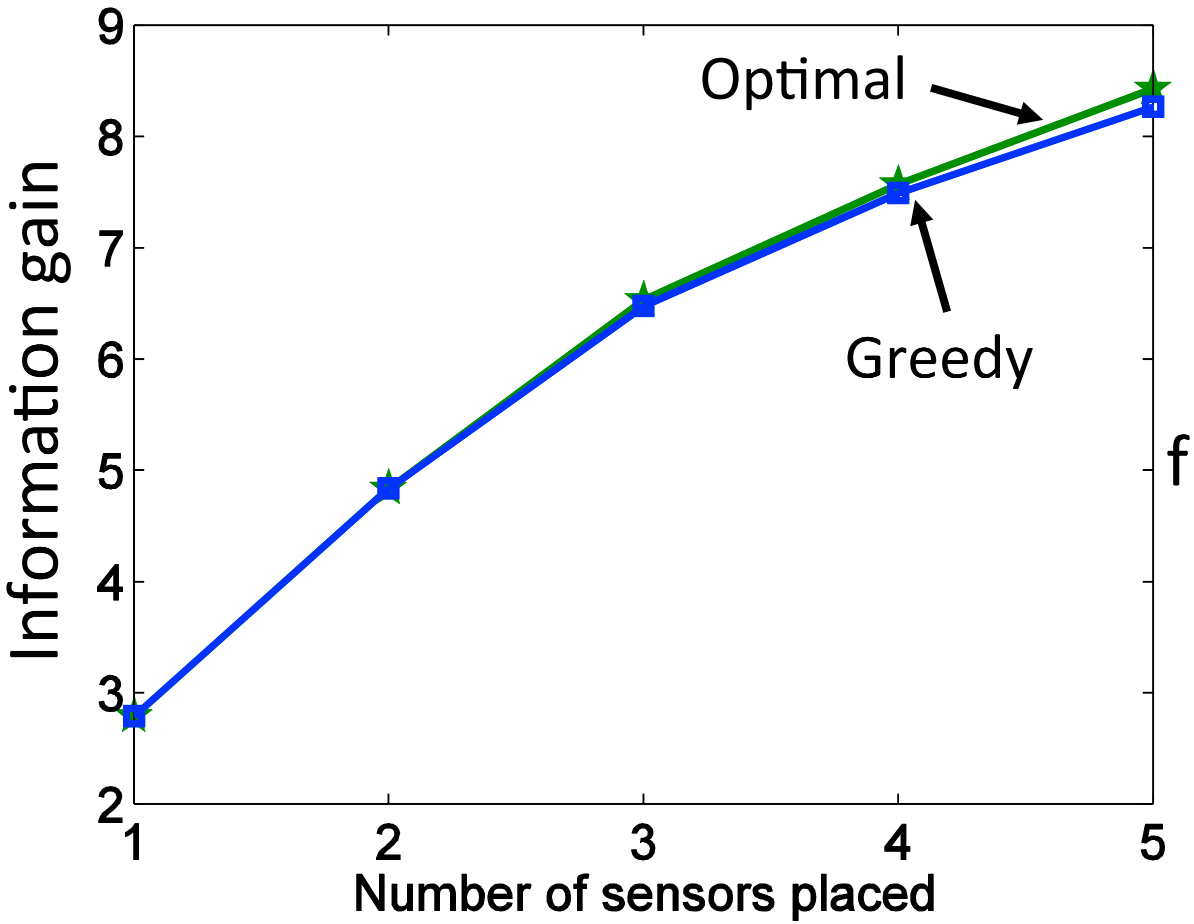
For i = 1 to k

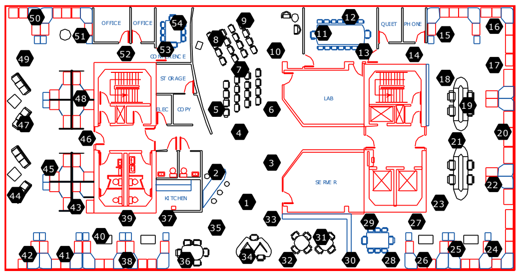
$$s^* \leftarrow \arg\max_s F(\mathcal{A} \cup \{s\})$$

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

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

# Scaling up the greedy algorithm [Minoux ’78]

In round i+1,

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

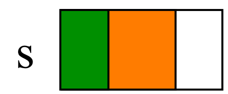
I.e., maximize “marginal benefit” $\Delta(s \mid A_i)$

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

**Key observation:** Submodularity implies

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

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

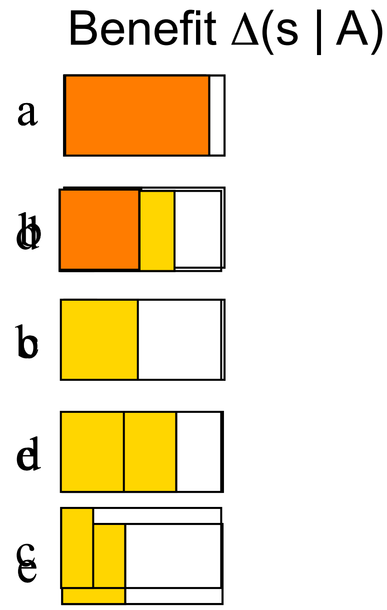



Marginal benefits can never increase!

# "Lazy" greedy algorithm [Minoux '78]

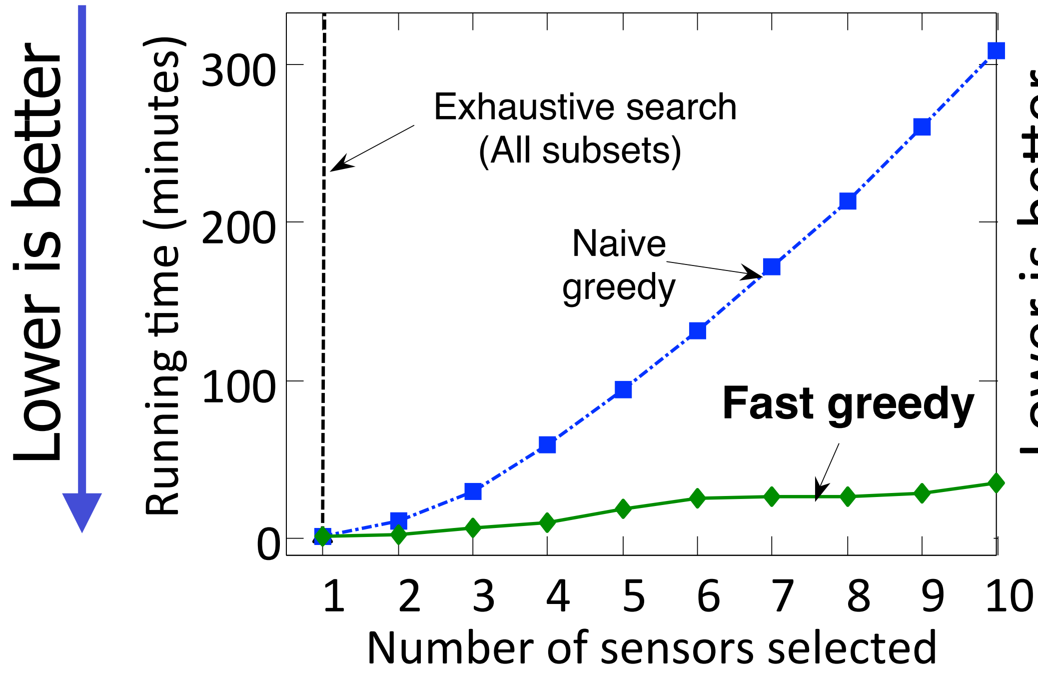
**Lazy greedy algorithm:**

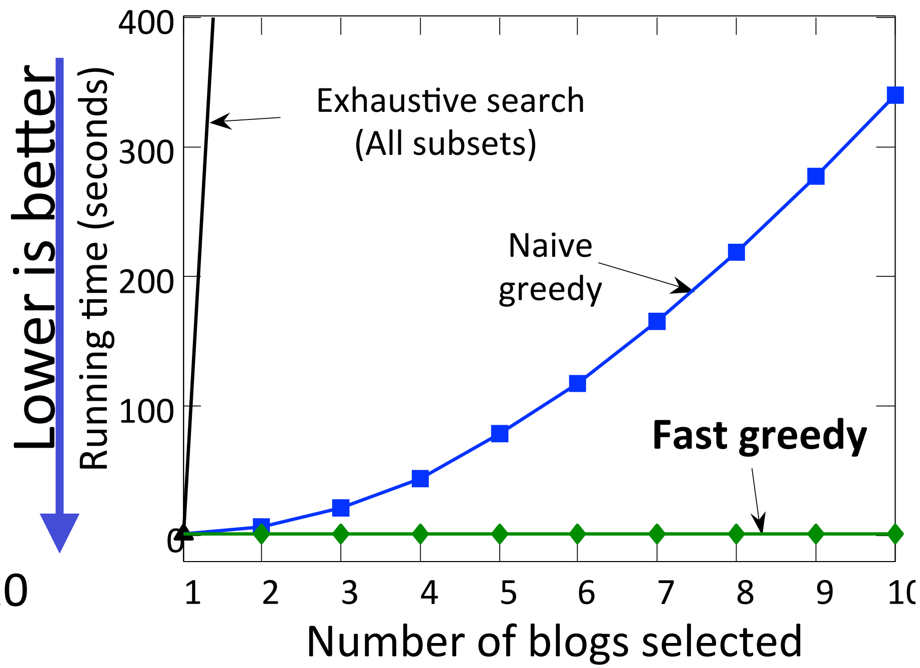
- First iteration as usual
- Keep an ordered list of marginal benefits $\Delta_i$ from previous iteration
- Re-evaluate $\Delta_i$ only for top element
- If $\Delta_i$ stays on top, use it, otherwise re-sort



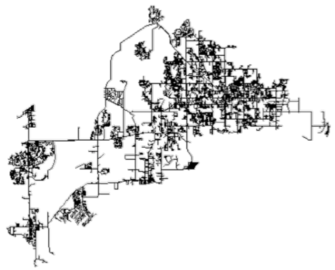

Note: Very easy to compute online bounds, lazy evaluations, etc. [Leskovec, Krause et al. '07]

# Empirical improvements [Leskovec, Krause et al'06]




Sensor placement

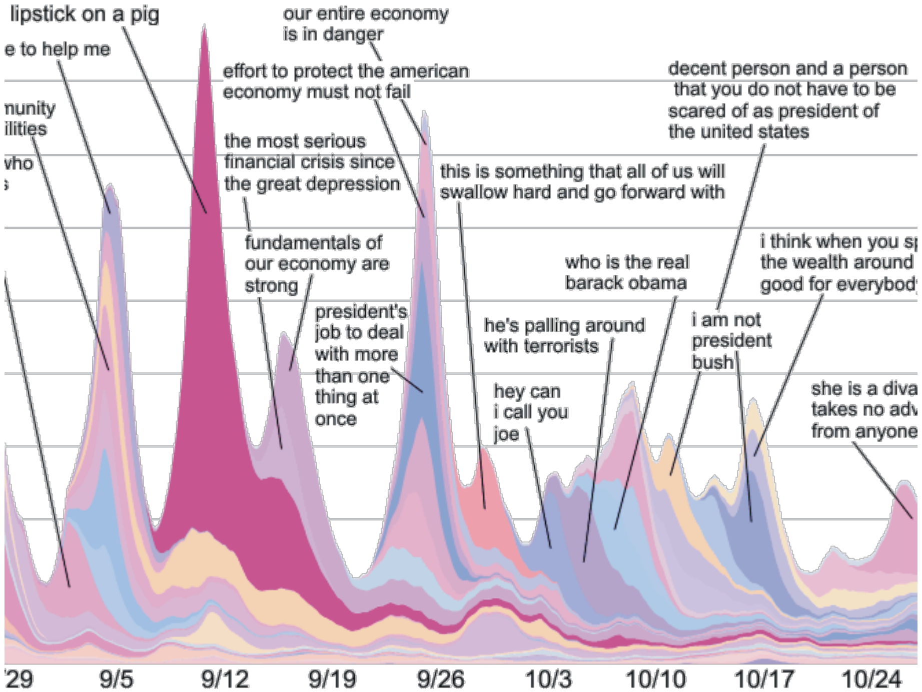


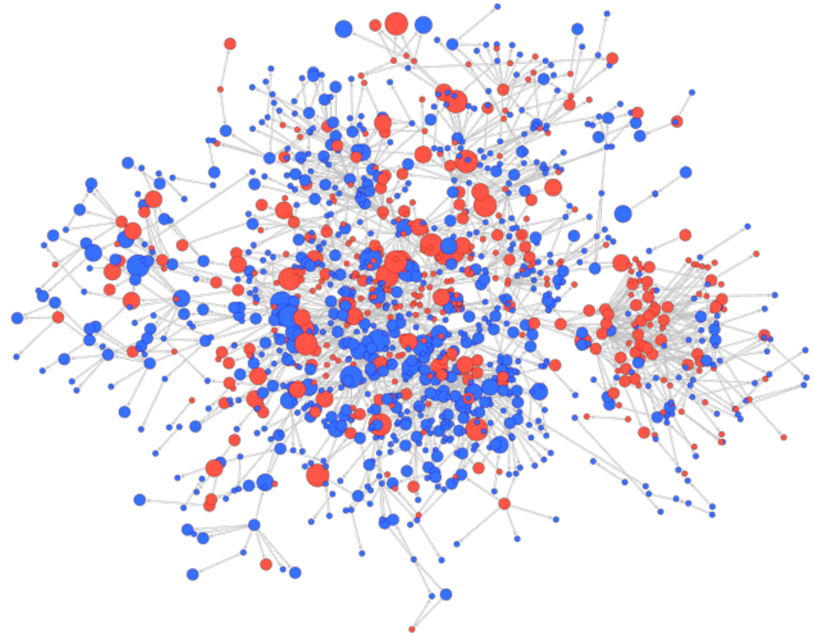
Blog selection

30x speedup

700x speedup

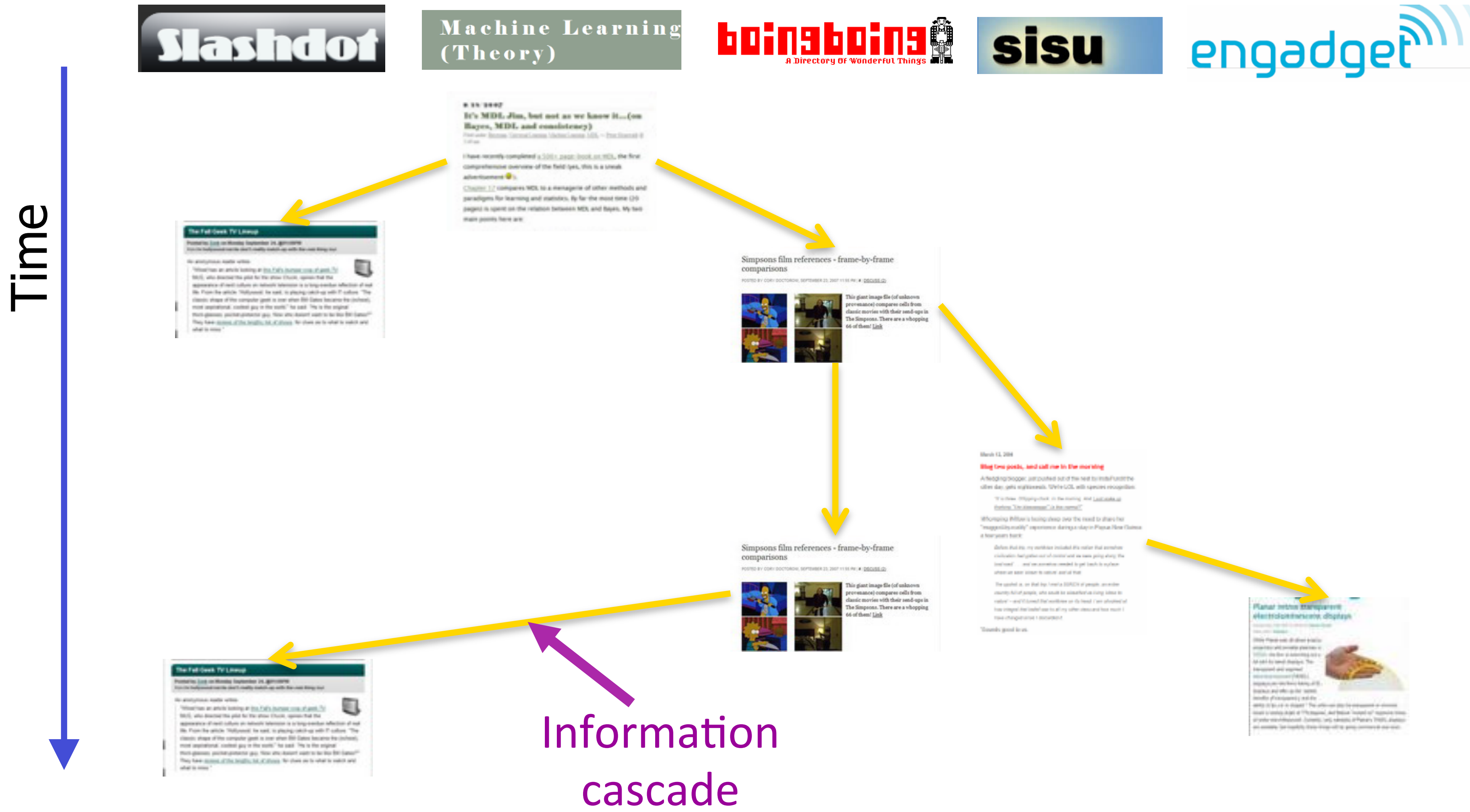
# Network inference



How can we learn who influences whom?

# Cascades in the Blogosphere



# Inferring diffusion networks

[Gomez Rodriguez, Leskovec, Krause ACM TKDE 2012]

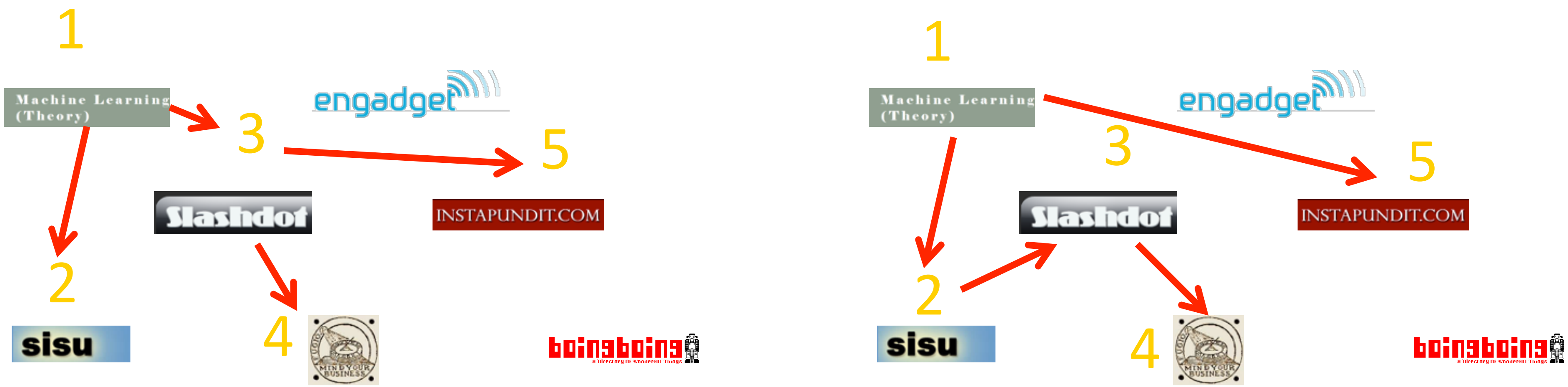
Given: Want:


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

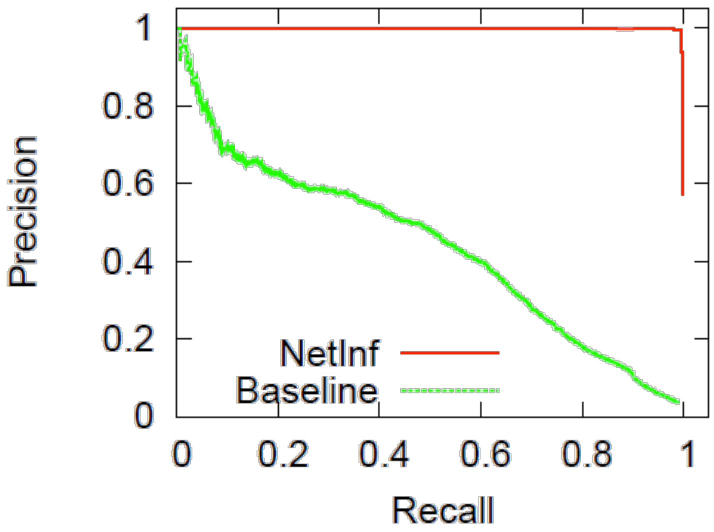



- Many influence trees T consistent with data
- For cascade $C_i$, model $P(C_i \mid T)$
- Find sparse graph that maximizes likelihood for all observed cascades

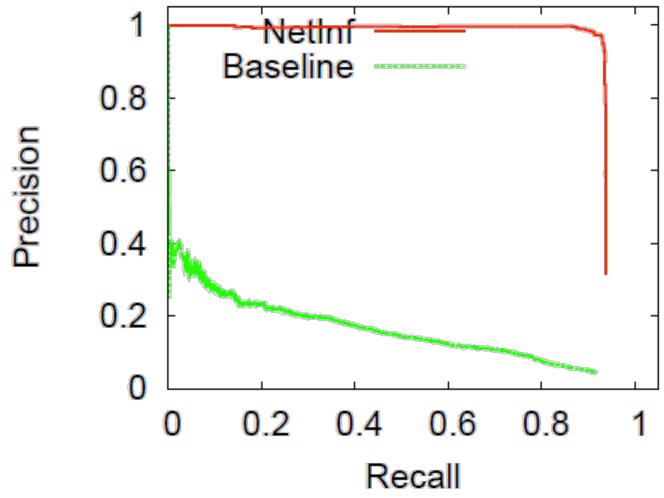
➔ Log likelihood monotonic submodular in selected edges

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

# Evaluation: Synthetic networks




1024 node hierarchical Kronecker exponential transmission model

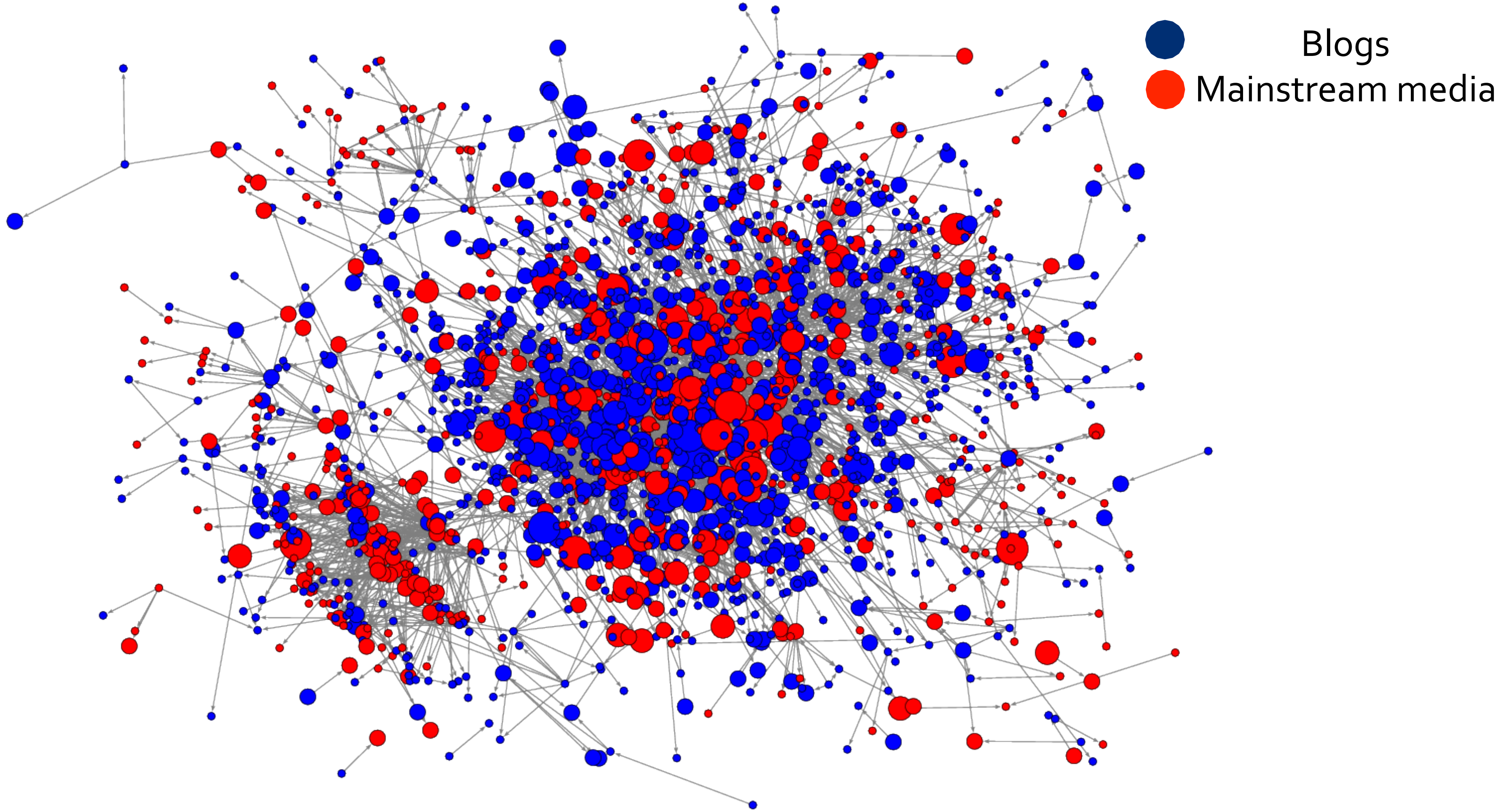



1000 node Forest Fire (α = 1.1) power law transmission model

- 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

# 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]

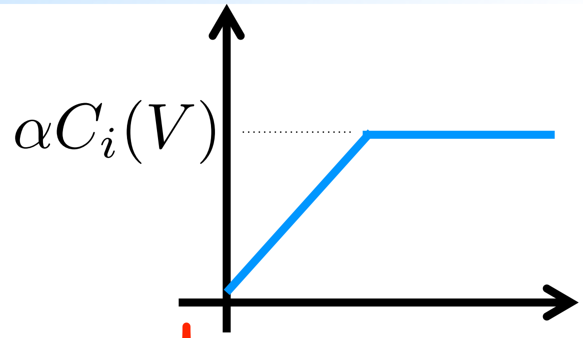
# Document summarization

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

Relevance — Diversity

# Relevance of a summary

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




$$R(S) = \sum_i C_i(S)$$

How well is sentence i „covered“ by S

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

Similarity between i and j

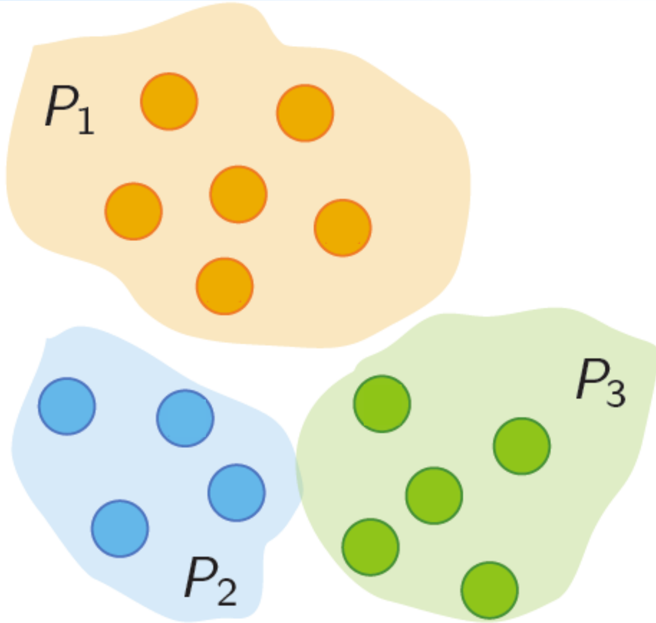
# Diversity of a summary

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

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

# 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** | **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 |

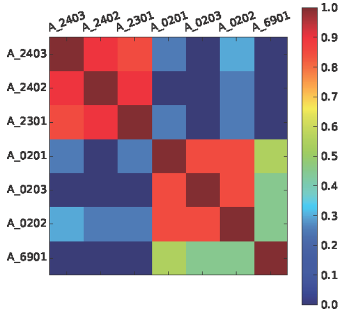
Best F1 score on benchmark corpus DUC-07!

Can do even better using submodular structured prediction! [Lin & Bilmes '12]

# 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]

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

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?

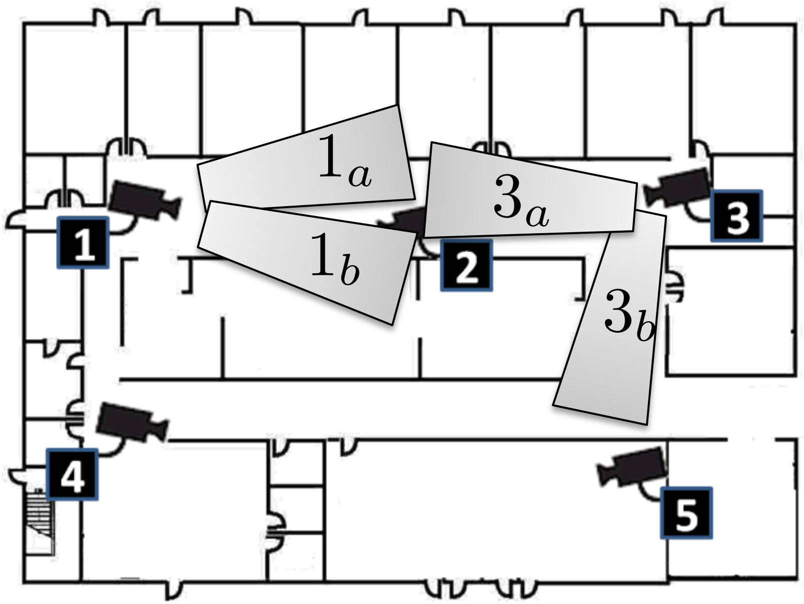
# Example: Camera network

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



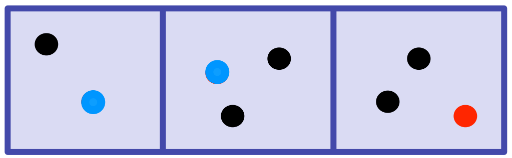
# Matroids

- Abstract notion of feasibility: independence

S is independent if …

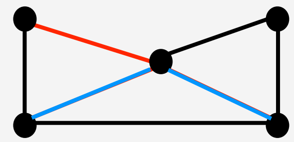
… |S| ≤ k

Uniform matroid



… S contains at most one element from each square

Partition matroid



… S contains no cycles

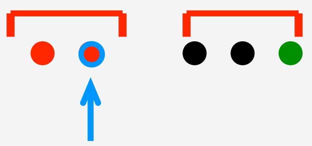
Graphic matroid

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

# Matroids

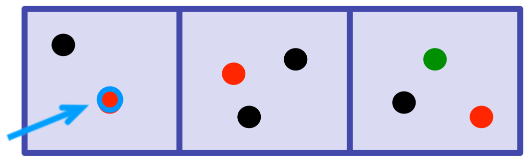
- Abstract notion of feasibility: independence

S is independent if …



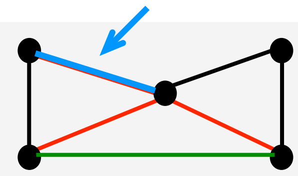
… |S| ≤ k

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

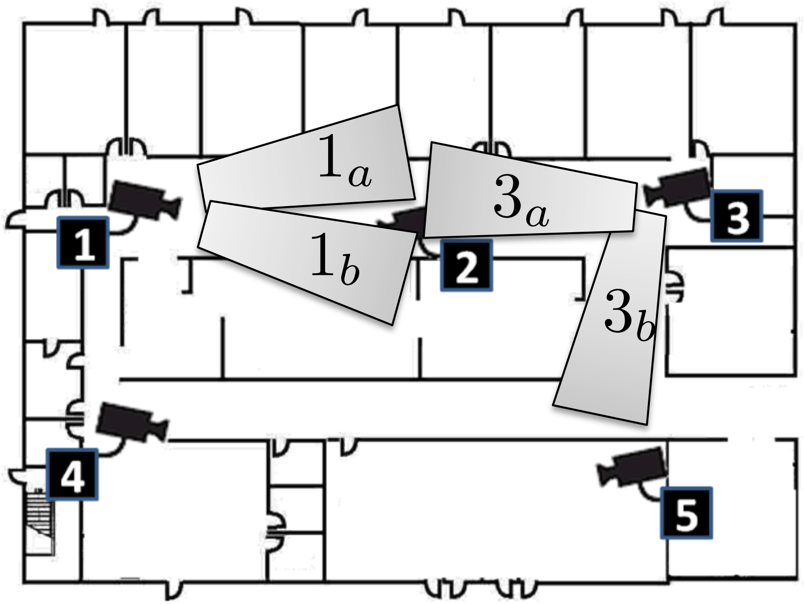
# Example: Camera network

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

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

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

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

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

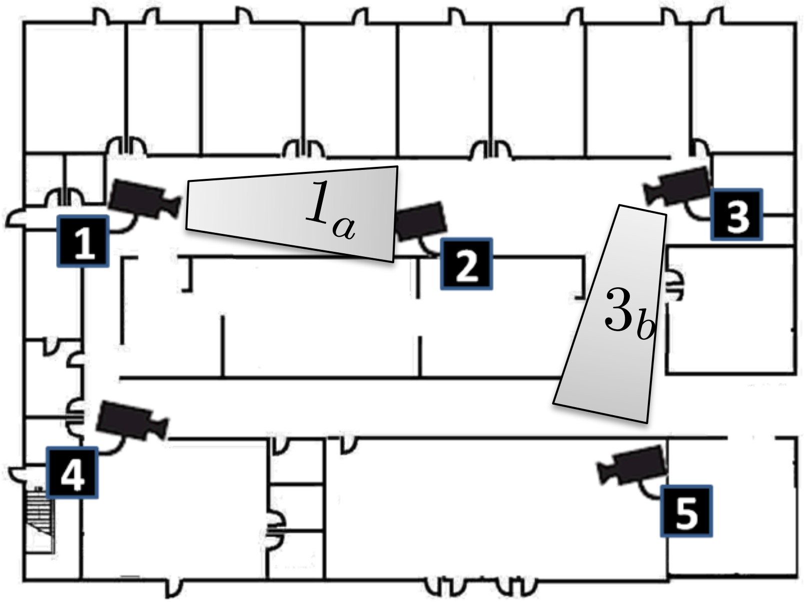
**Greedy algorithm:**

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 matchings / rankings with p=2:
    Each item can be assigned ≤ 1 rank, each rank can take ≤ 1 item
- Can get also obtain (1-1/e) for arbitrary matroids [Vondrak et al ’08] using continuous greedy algorithm

# Maximization: More complex constraints

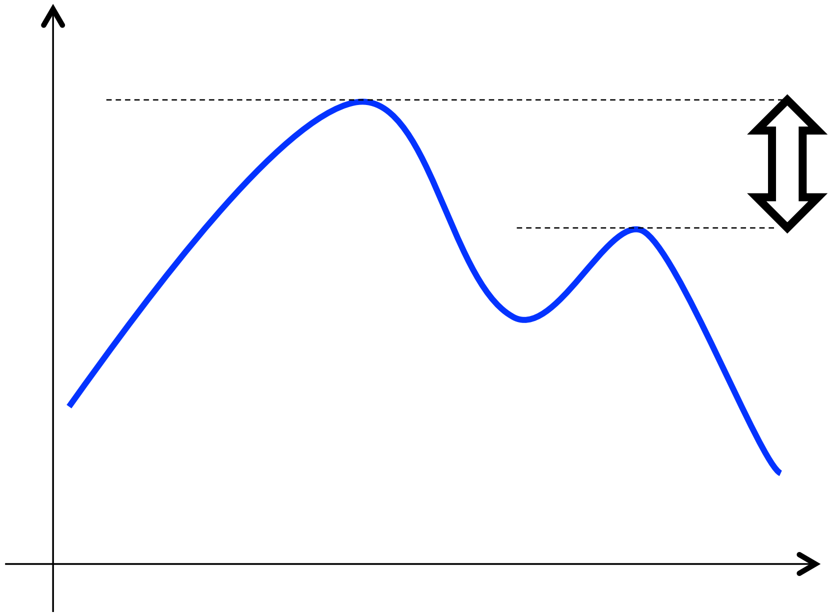
- Approximate submodular maximization possible under a variety of constraints:
  - (Multiple) matroid constraints
  - Knapsack (non-constant cost functions)
  - Multiple matroid and knapsack constraints
  - Path constraints (Submodular orienteering)
  - Connectedness (Submodular Steiner)
  - Robustness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

Greedy works well (first two items)

Need non-greedy algorithms (remaining items)

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

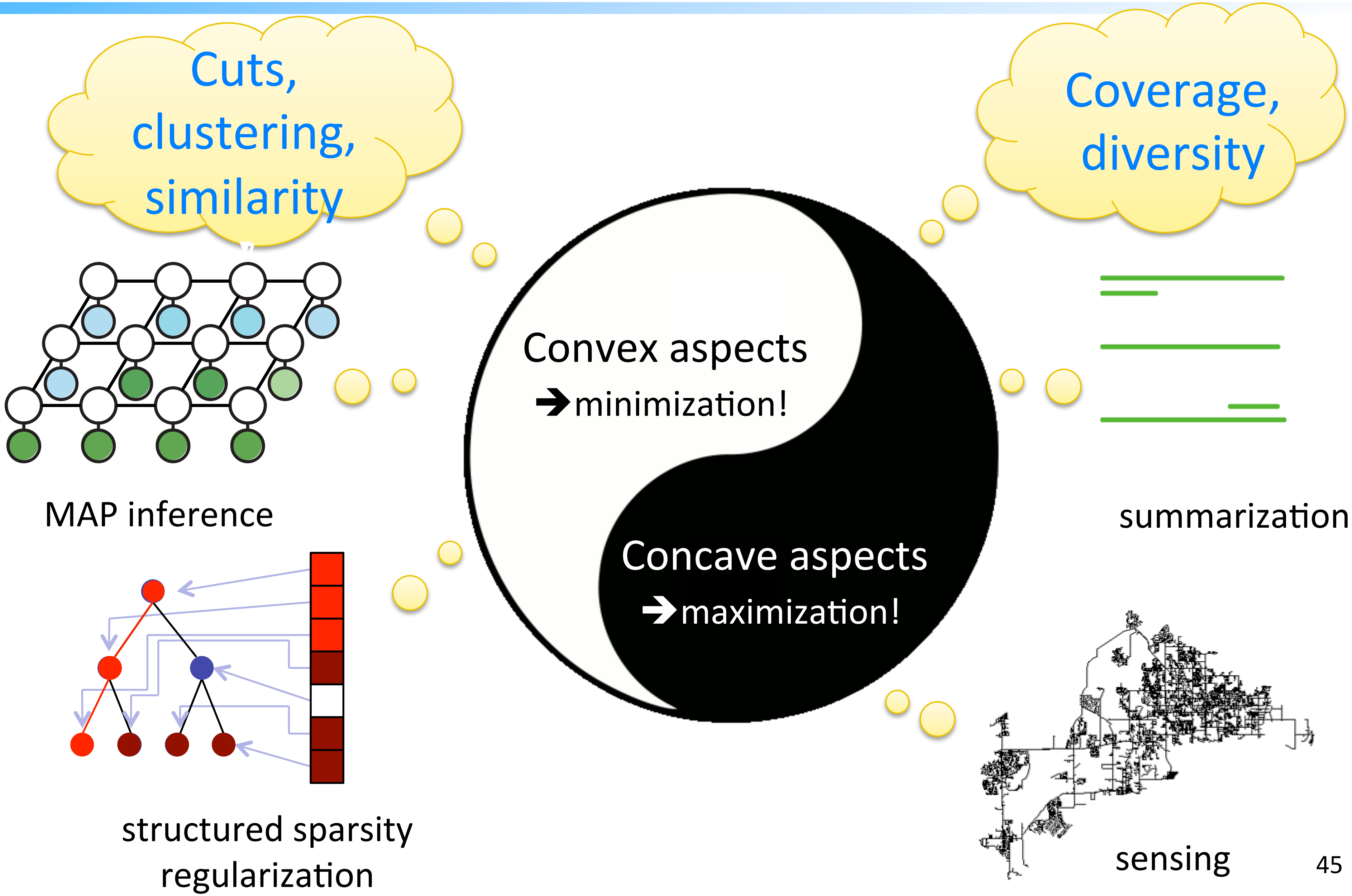
# Key intuition for approx. maximization



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

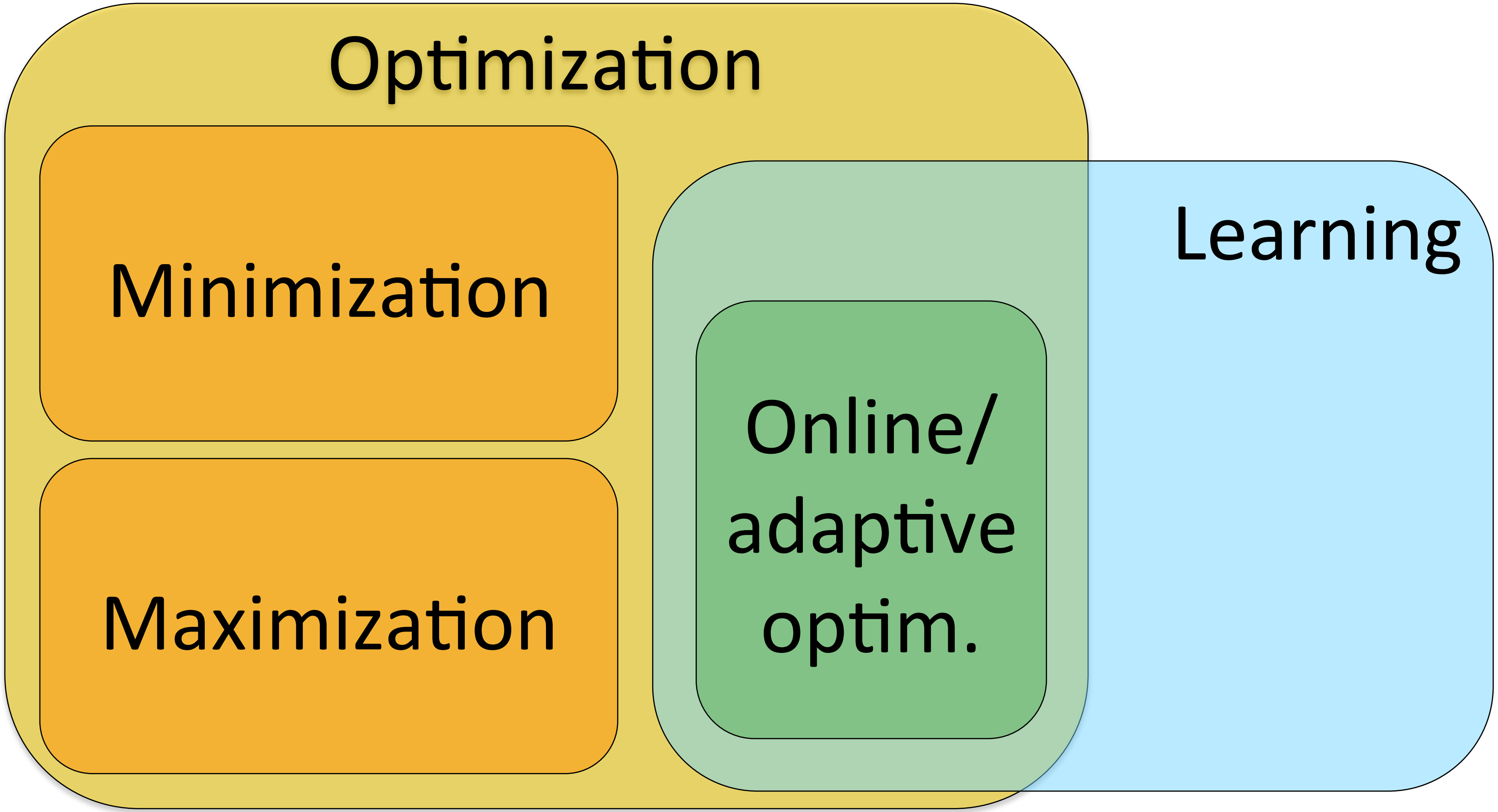
- E.g., all local maxima under cardinality constraints are within factor 2 of global maximum
- Key insight for more complex maximization
  ➔ Greedy, local search, simulated annealing
  for (non-monotone, constrained, ...)

# Two-faces of submodular functions

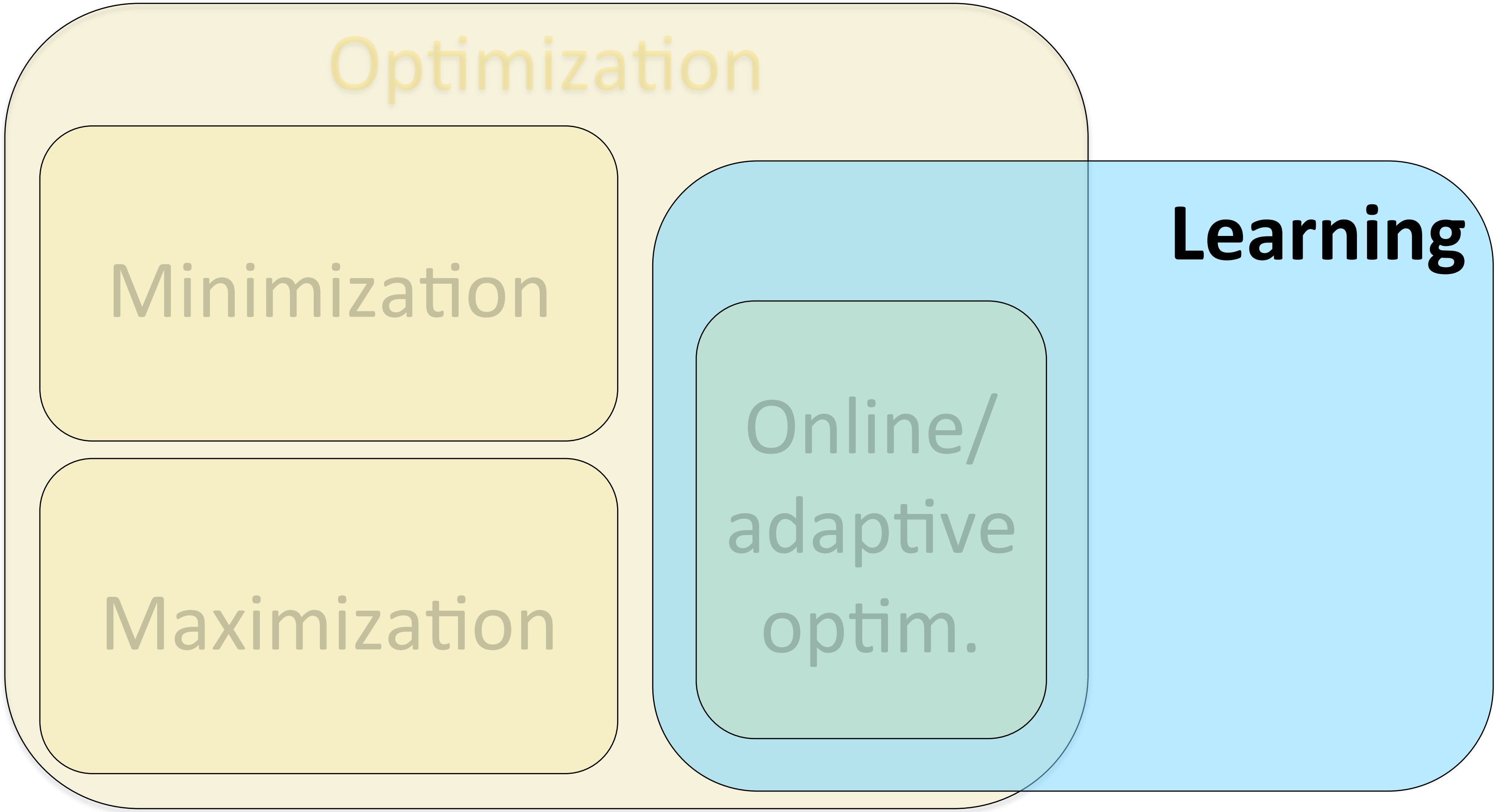


| | Maximization | Minimization |
|---|---|---|
| Unconstrained | NP-hard, but well-approximable (if nonnegative) | Polynomial time! Generally inefficent (n^6), but can exploit special cases (cuts; symmetry; decomposable; ...) |
| Constrained | NP-hard but well-approximable „Greedy-(like)" for cardinality,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

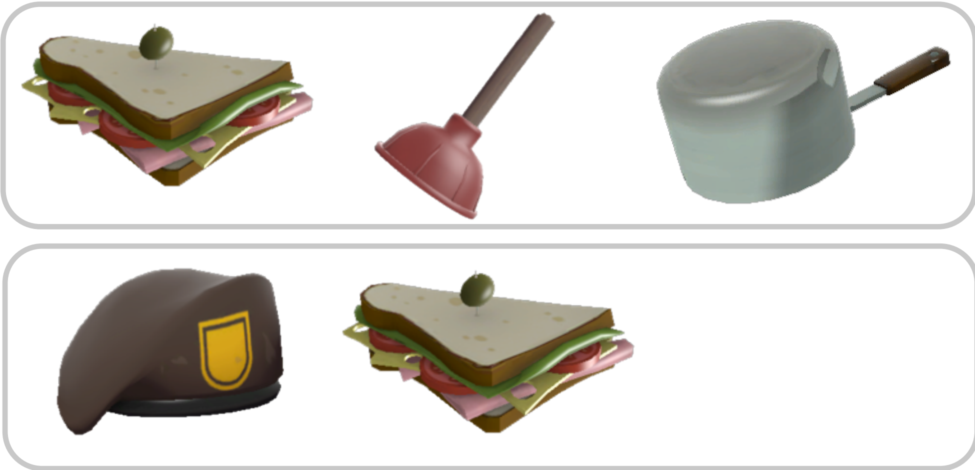
# What to do with submodular functions



# What to do with submodular functions



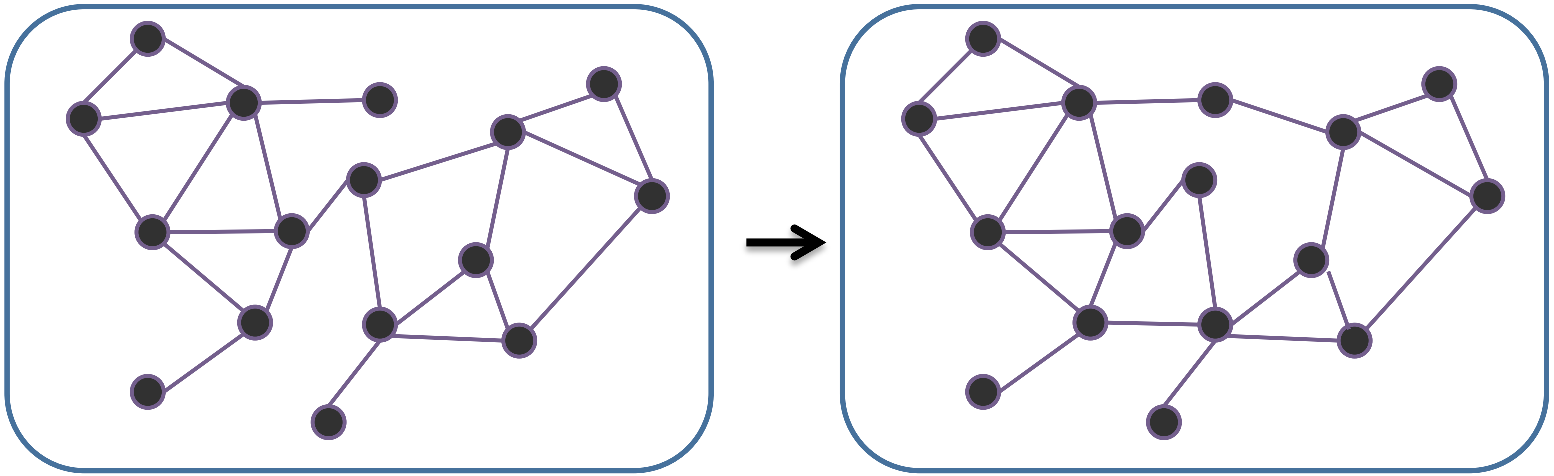
# Example 1: Valuation Functions



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

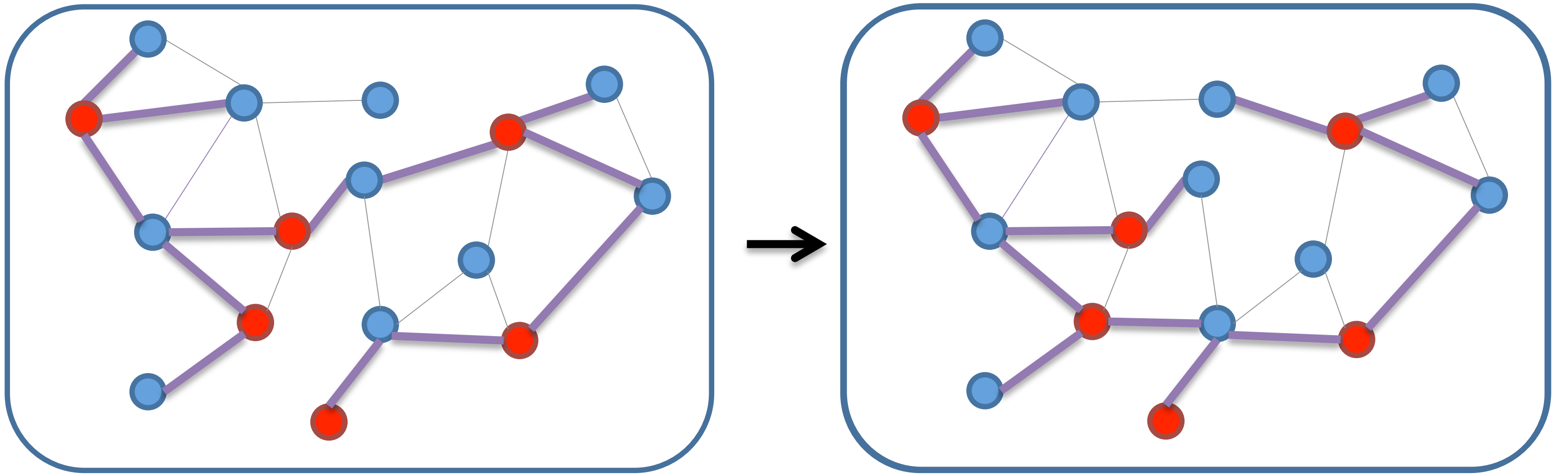
**Can we learn a bidder's utility function from few bids?**

# Example 2: Graph Evolution



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

# Random Graph Cut #1

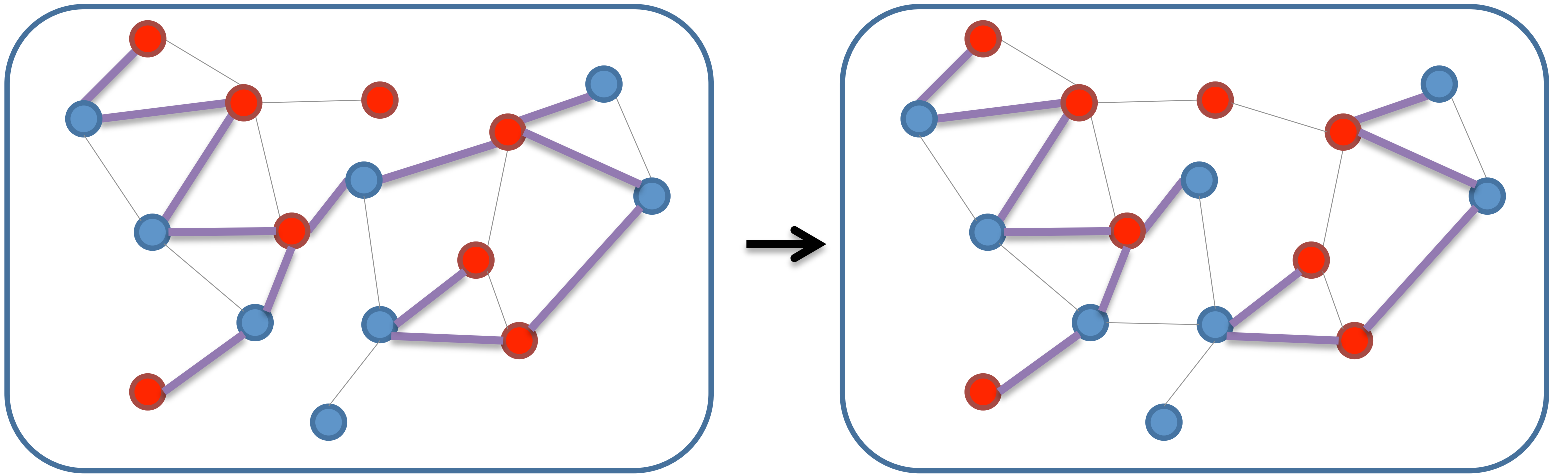


Cut value = 13

Cut value = 14

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

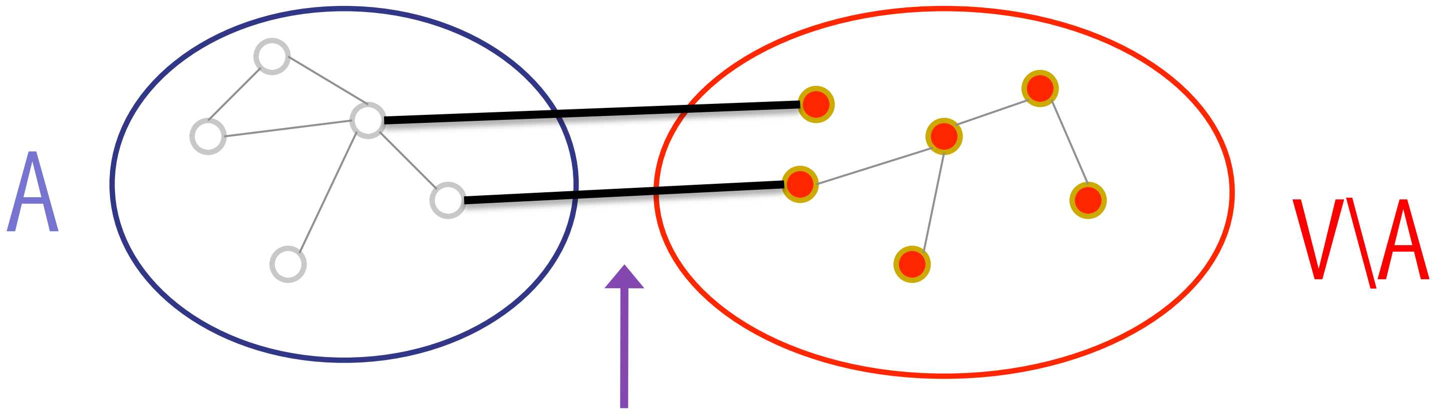
# Random Graph Cut #2



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

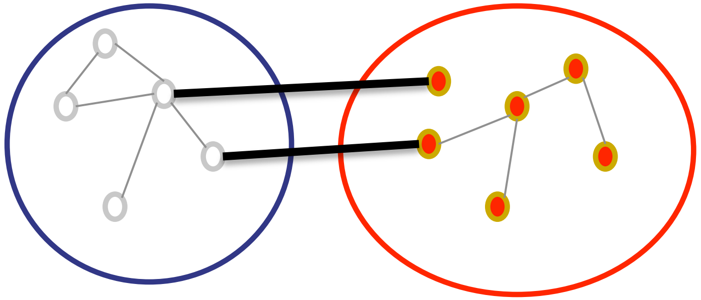
- V = set of vertices
- One-to-one correspondence of graphs and cut functions

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



Can we learn F from few measurements / data?

$$\{(A_1, F(A_1)), \ldots, (A_m, F(A_m))\}$$

# "Regressing" submodular functions
## [Balcan, Harvey STOC '11]

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

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

**Theorem**: cannot approximate better than

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

unless one looks at exponentially many samples $A_i$

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

# 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 $A_i$ adaptively,

cannot approximate better than $\alpha = n^{½} / \log(n)$
unless one looks at exponentially many sets $A_i$

But can efficiently obtain $\alpha = n^{½} \log(n)$

# What if we have structure?

- To learn effectively, need additional assumptions beyond submodularity.

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

$$F(A) = \sum_{B \subseteq V} (-1)^{|A \cap B|} \hat{F}(B)$$

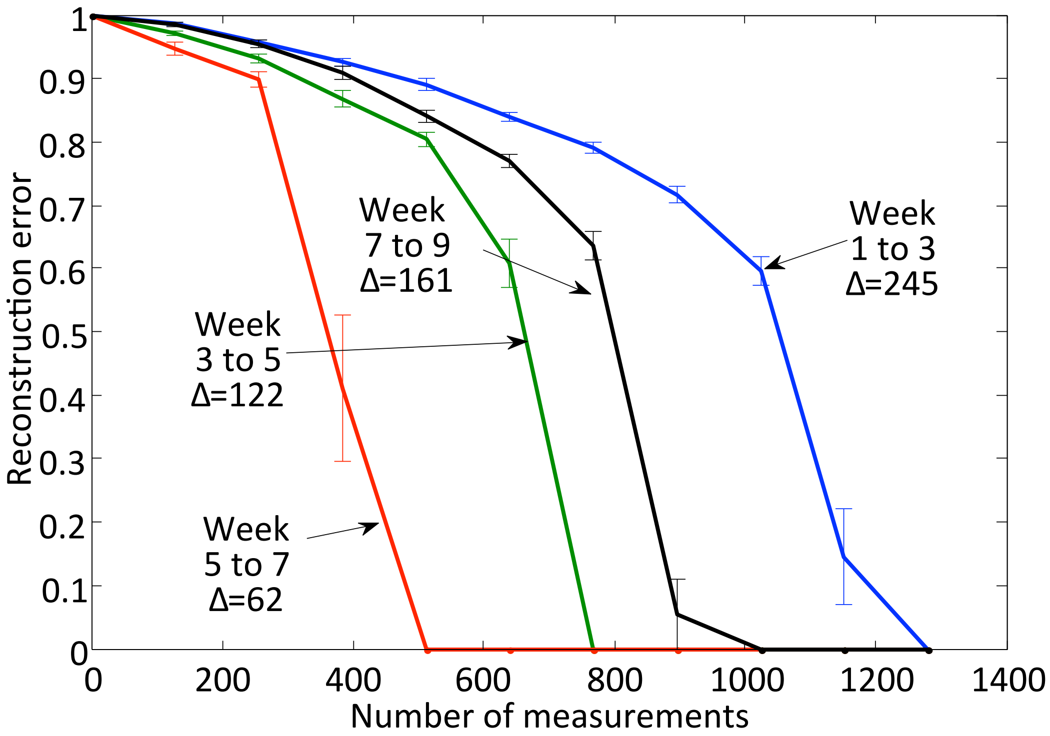
Sparsity: Most coefficients ≈0

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

# Results: Sketching Graph Evolution

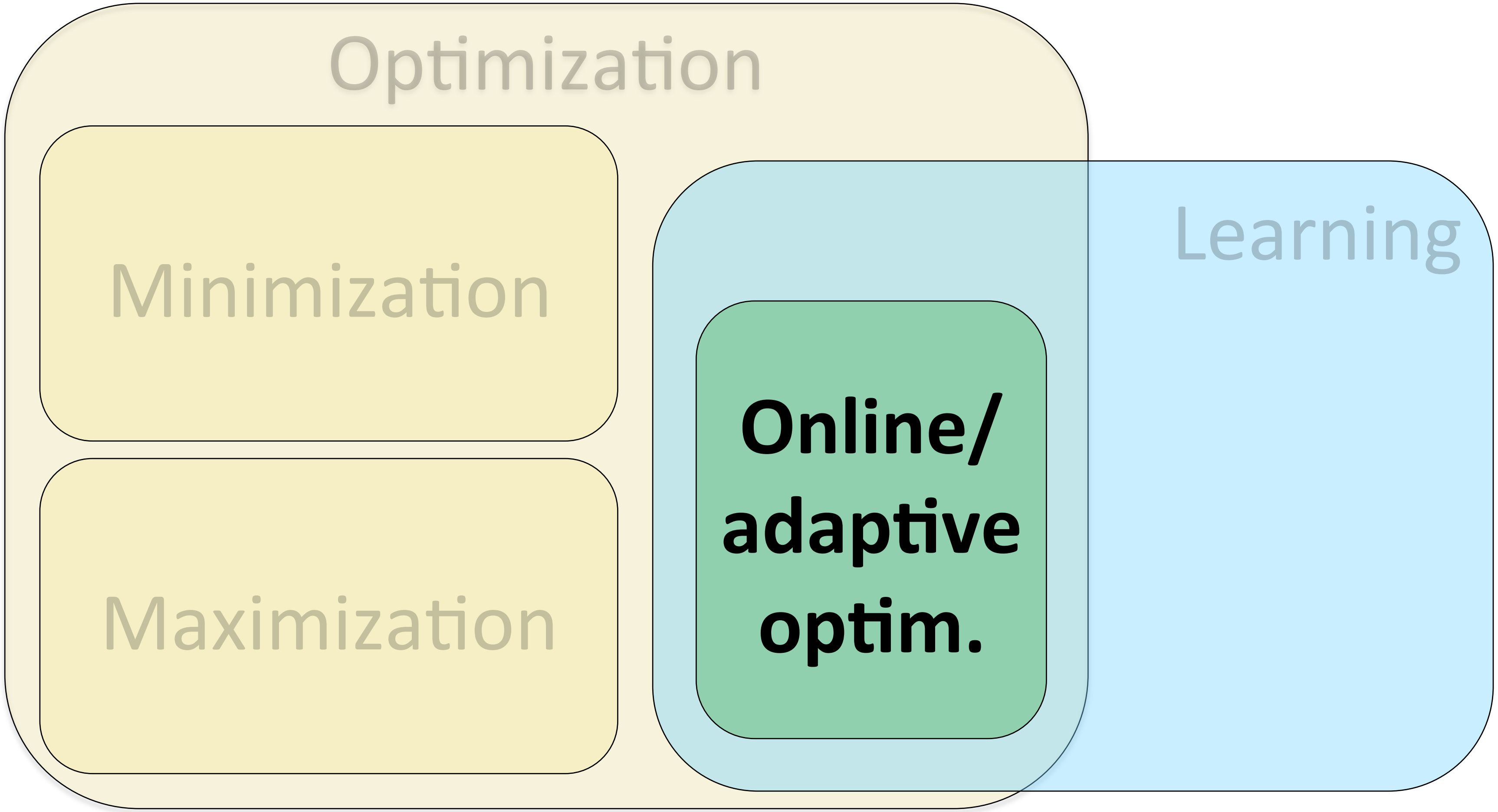
[Stobbe & Krause '12]

- 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

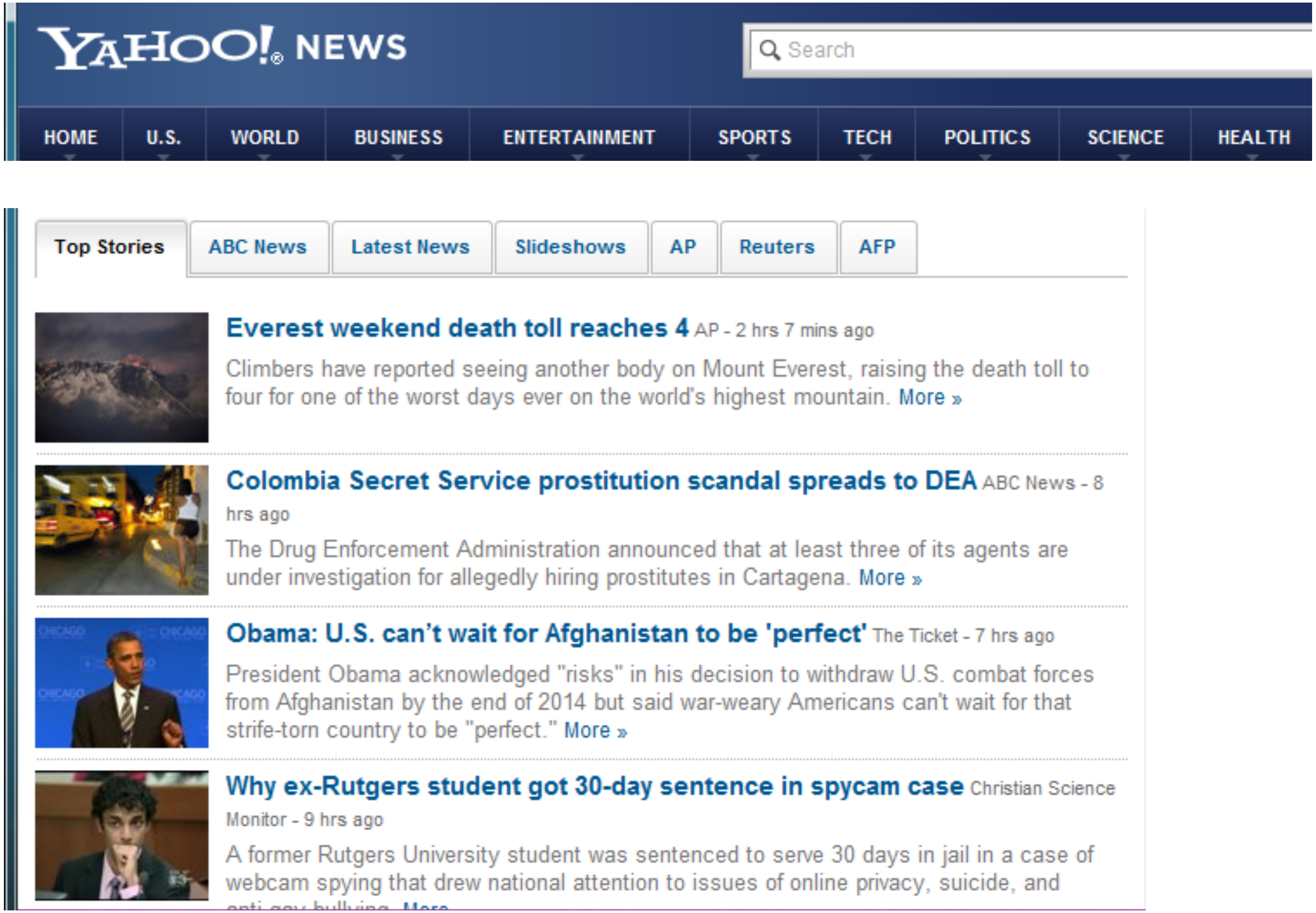
- Have seen how to
  - optimize submodular functions
  - learn submodular functions

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

# Learning to optimize submodular functions

- Online submodular optimization
  - Learn to pick a sequence of sets to maximize a sequence of (unknown) submodular functions
  - *Application*: Making diverse recommendations
- Adaptive submodular optimization
  - Gradually build up a set, taking into account feedback
  - *Application*: Experimental design / Active learning

# News recommendation



# Application: Diverse Recommendations

"Google to DOJ: Let us prove to users that NSA isn't snooping on them"
"US tech firms push for govt transparency on securityReuters"
"Internet Companies Call For More Disclosure of Surveillance"
"NSA scandal: Twitter and Microsoft join calls to disclose data requests"
"NSA Secrecy Prompts a Pushback"

"Google to DOJ: Let us prove to users that NSA isn't snooping on them"
"Storms Capable of Producing Derecho Possible in Midwest Today"
"Ohio kidnap suspect pleads not guilty"
"Five takeaways from Spurs-Heat in Game 3 of the NBA Finals"
"Samsung Unveils Galaxy S4 Zoom With 16MP Camera"

Prefer recommendations that are both *relevant* and *diverse*

# Simple model

- We‘re given a set of articles $V$
- Each round:
  - A user appears, interested in a subset of the articles $S_t$
  - We recommend a set of articles $A_t$
  - The user clicks on any displayed article that she is interested in

$$F_t(A_t) = \min(|A_t \cap S_t|, 1)$$

- **Goal**: Maximize the total #of clicks $\sum_t F_t(A_t)$
- **Challenge**:
  - We don‘t know which articles the user is interested in!

# 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 $r_1=F_1(A_1)$ $r_2$ $r_3$ … $r_T$

Time

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

Total: $\sum_t r_t \rightarrow$ max

**Goal**: Want to choose $A_1,\ldots A_t$ s.t. the regret

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

grows sublinearly, i.e., $R_T/T \rightarrow 0$

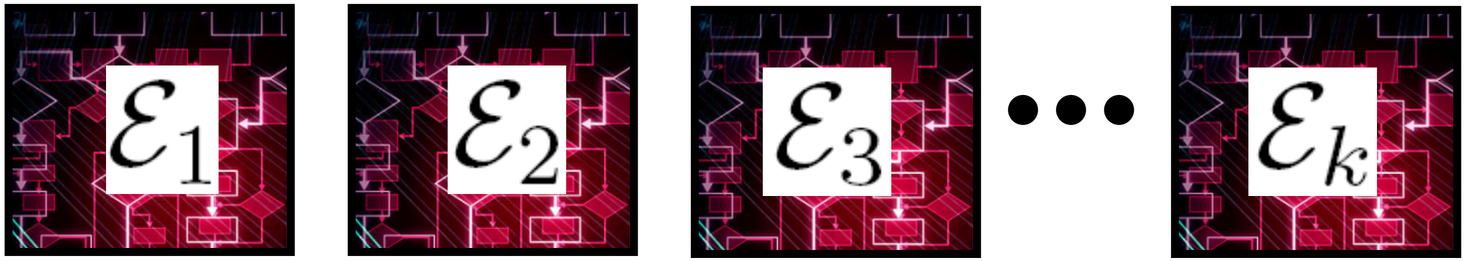
**For k=1, many good algorithms known! ☺**

**But what if k>1?**

# Online Greedy Algorithm

[Streeter & Golovin, NIPS `08]

Replace each stage of greedy algorithm with a multi-armed bandit algorithm.




Select $\{ a_1, \quad a_2, \quad a_3, \quad \ldots, \quad a_k \}$ .

Feedback to $\mathcal{E}_j$ for action $\mathbf{a_j}$ is (unbiased est. of)
$F_t(\{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_{j-1}}, \mathbf{a_j}\}) - F_t(\{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_{j-1}}\})$

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

**Theorem**
Online greedy algorithm chooses $A_1,\ldots,A_T$ s.t.
for any sequence $F_1,\ldots,F_T$

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

Can get 'no-regret' over greedy algorithm in hindsight
I.e., can learn ``enough'' about F to optimize greedily!

# Stochastic linear submodular bandits

[Yue & Guestrin '11]

- Basic submodular bandit algorithm has slow convergence
- Can do better if we make stronger assumptions
  - Submodular function is linear combination of m SFs

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

  - We evaluate it up to (stochastic) noise*

$$F_t(S) = F(S) + \text{noise}$$

➔ **LSBGreedy algorithm**

*some independence conditions

# User Study [Yue & Guestrin '11]

- Real data: >10k articles
- T=10 days, rec. 10 articles per day

- 27 users rate articles, aim to maximize #likes

| |
|---|
| "Google to DOJ: Let us prove to users that NSA isn't snooping on them" ✓ |
| "Storms Capable of Producing Derecho Possible in Midwest Today" ✓ |
| "Ohio kidnap suspect pleads not guilty" ✗ |
| "Five takeaways from Spurs-Heat in Game 3 of the NBA Finals" ✓ |
| "Samsung Unveils Galaxy S4 Zoom With 16MP Camera" ✗ |

- LSBGreedy outperforms baselines that fail to …
  - adapt weights (no personalization)
  - address the exploration—exploitation tradeoff
  - model diversity explicitly

# Other results on online submodular optimization

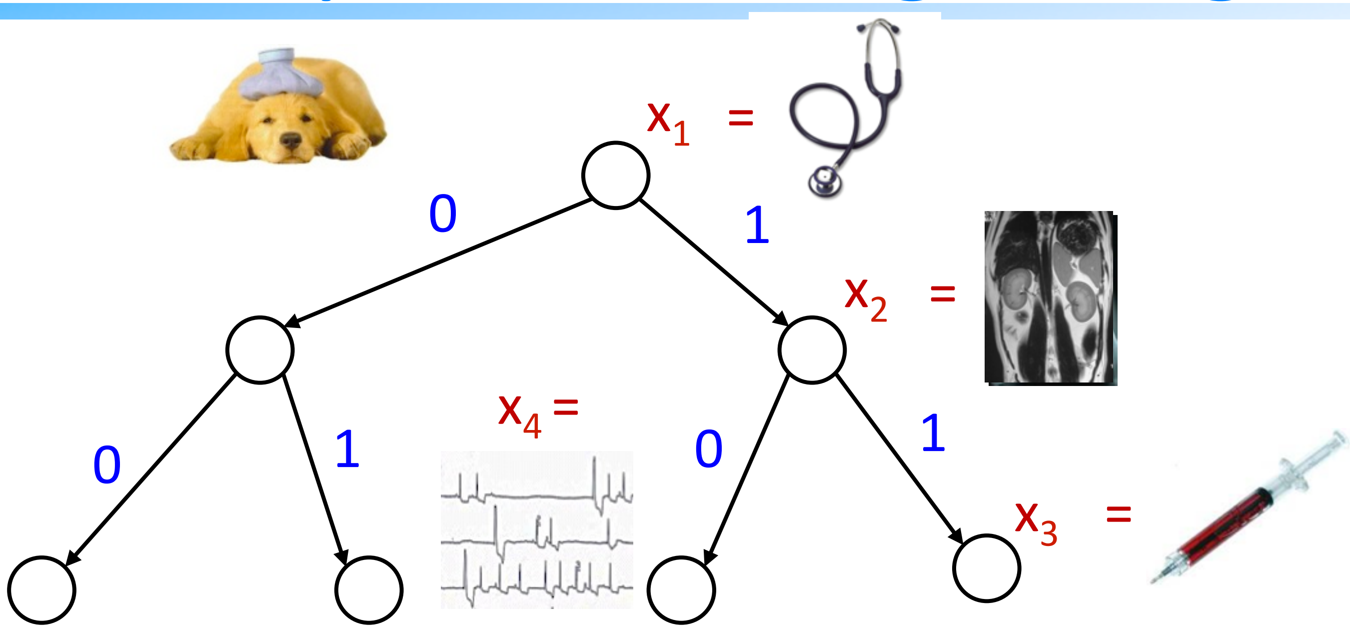
- Online submodular maximization
  - No (1-1/e) regret for ranking (partition matroids) [Streeter, Golovin, Krause 2009]
  - Distributed implementation [Golovin, Faulkner, Krause '2010]
- Online submodular coverage
  - Min-cost / Min-sum submodular cover [Streeter & Golovin NIPS 2008, Guillory & Bilmes NIPS 2011]
- Online Submodular Minimization
  - Unconstrained [Hazan & Kale NIPS 2009]
  - Constrained [Jegelka & Bilmes ICML 2011]
- See also the „submodular secretary problem"

# Learning to optimize submodular functions

- Online submodular optimization
  - Learn to pick a sequence of sets to maximize a sequence of (unknown) submodular functions
  - *Application*: Making diverse recommendations

- Adaptive submodular optimization
  - Gradually build up a set, taking into account feedback
  - *Application*: Experimental design / Active learning

# Adaptive Sensing / Diagnosis



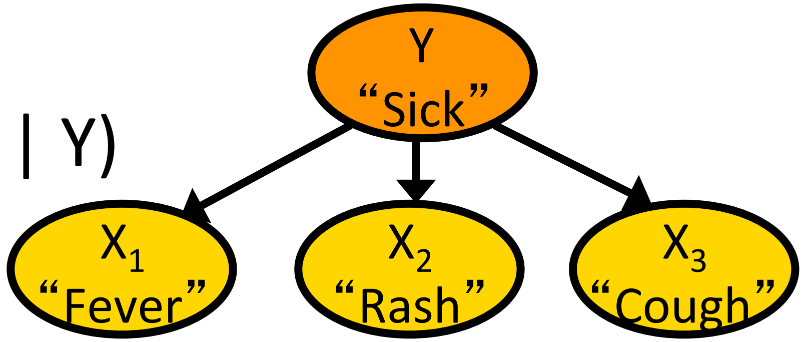

Want to effectively diagnose while minimizing cost of testing!

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

*States y*

# Problem Statement

Given:

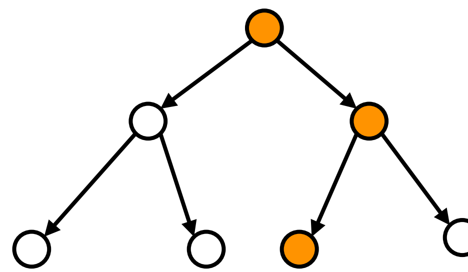
- Items (tests, experiments, actions, …) V={1,…,n}
- Associated with random variables $X_1,\ldots,X_n$ taking values in O
- Objective: $f : 2^V \times O^V \to \mathbb{R}$
- Policy π maps observation $\mathbf{x}_A$ to next item

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 $\mathbf{x}_A$

**Adaptive Greedy algorithm:**

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:*

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

**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 |
| $\Delta_F(s \mid A) = F(A \cup \{s\}) - F(A)$ | $\Delta_F(s \mid \mathbf{x}_A) = \mathbb{E}\Big[f(A \cup \{s\}, \mathbf{x}_V) - f(A, \mathbf{x}_V) \mid \mathbf{x}_A\Big]$ |
| $\Delta_F(s \mid A) \geq 0$ | $\Delta_F(s \mid \mathbf{x}_A) \geq 0$ |
| $A \subseteq B \Rightarrow \Delta_F(s \mid A) \geq \Delta_F(s \mid B)$ | $\mathbf{x}_A \preceq \mathbf{x}_B \Rightarrow \Delta_F(s \mid \mathbf{x}_A) \geq \Delta_F(s \mid \mathbf{x}_B)$ |
| $\max_A F(A)$ | $\max_\pi F(\pi)$ |
| Greedy algorithm provides | Greedy policy provides |

**Submodularity** — Greedy algorithm provides

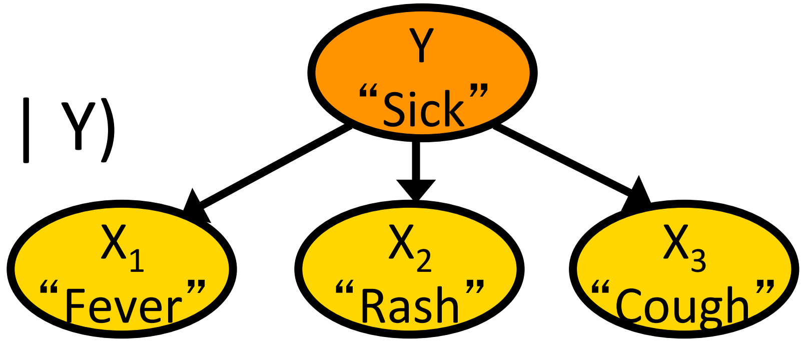
- *(1-1/e)* for max. w card. const.
- *1/(p+1)* for p-indep. systems
- *log Q* for min-cost-cover
- *4* for min-sum-cover

**Adaptive submodularity** — 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)

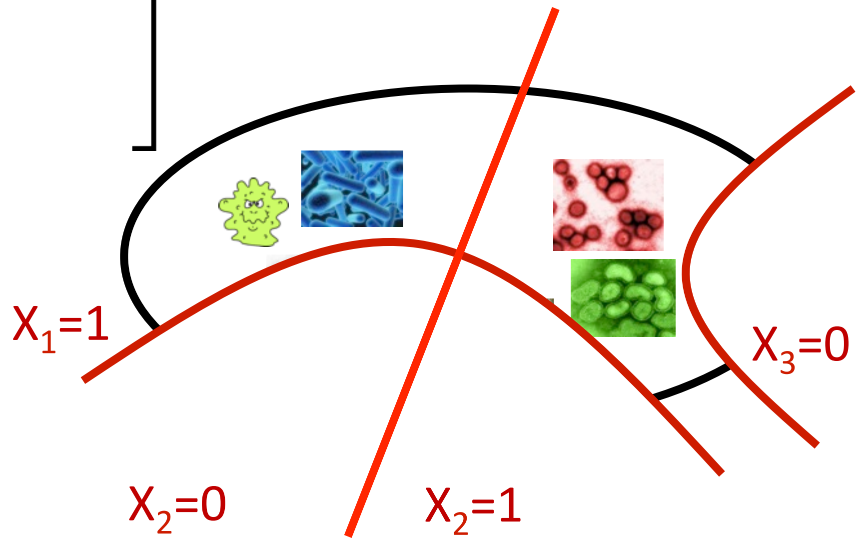



- How should we test to eliminate all incorrect hypotheses?

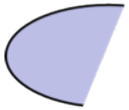
$$\Delta(t \mid x_A) = \mathbb{E}\left[\begin{array}{l}\text{mass ruled out}\\ \text{by } t \text{ if we}\\ \text{know } x_A\end{array}\right]$$

"Generalized binary search"

Equivalent to max. infogain



# OD is Adaptive Submodular

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

Objective = probability mass of hypotheses you have ruled out.

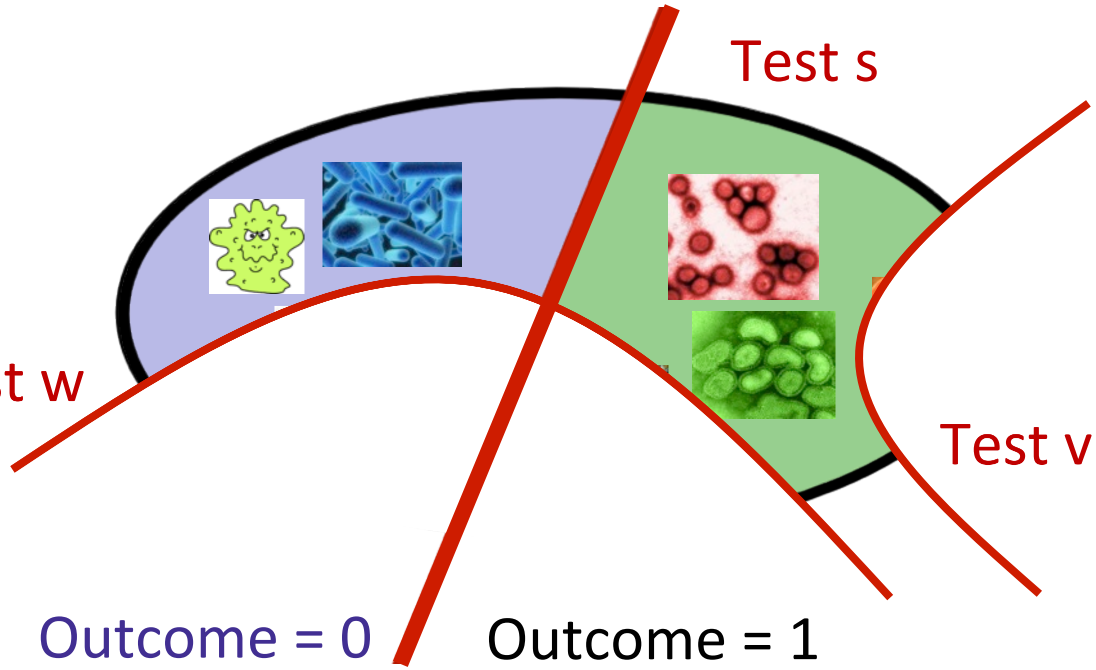
$g_0 := \mathbb{P}($  $)$

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

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

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

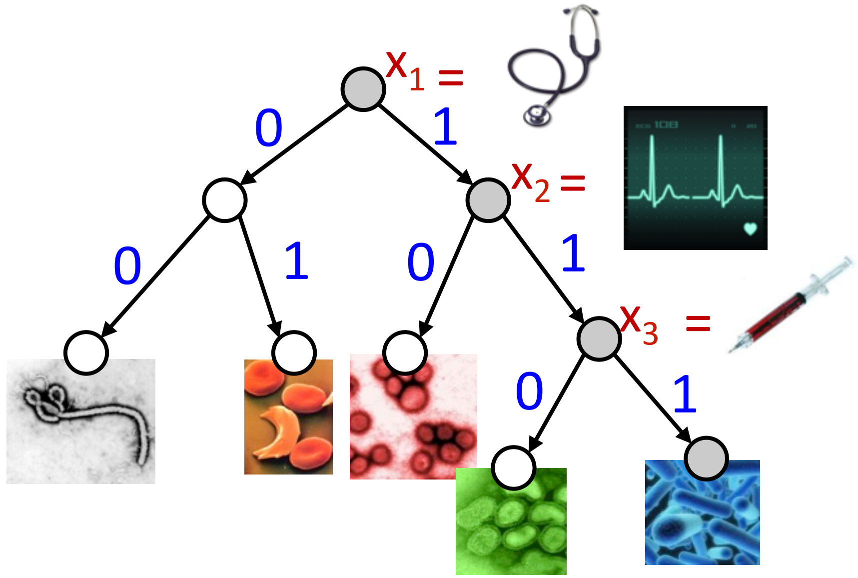
$$\Delta(s \mid \mathbf{x}_{v,w}) = \frac{2 g_1 b_1}{g_1 + b_1}$$




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

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

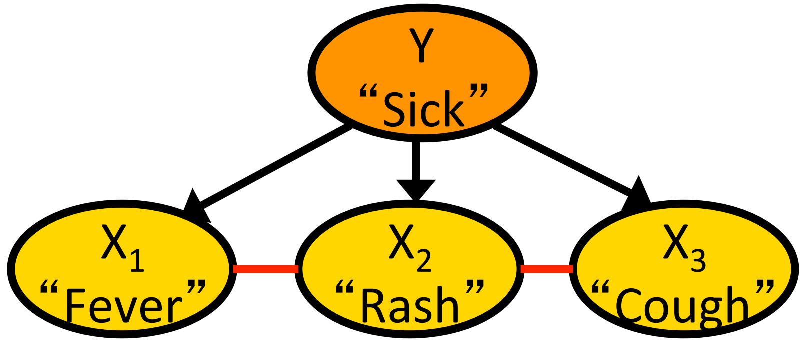
Adaptive-Greedy is a $(\ln(1/p_{\min}) + 1)$ approximation.

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($\mathbf{X}_V$ | 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

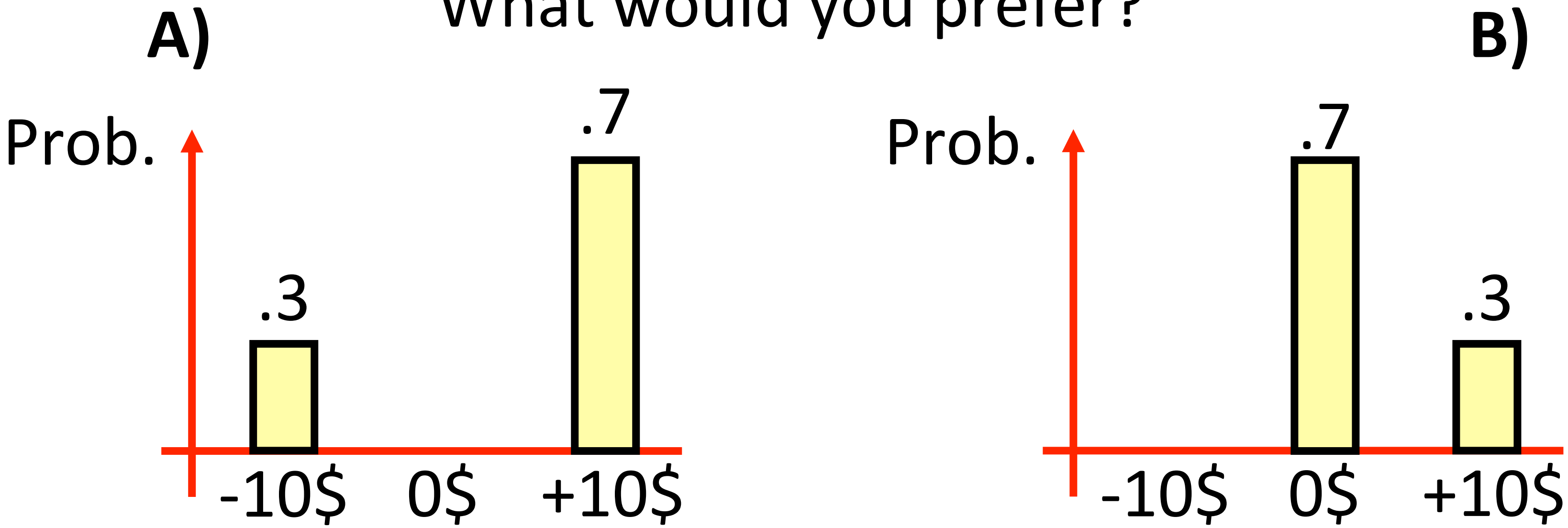
$$\mathrm{Cost}(\pi_{\mathrm{Greedy}}) \leq \mathcal{O}\left(\log \frac{1}{\delta}\right) \mathrm{Cost}(\pi^*)$$

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

# Example: The Iowa Gambling Task

## [with Colin Camerer, Deb Ray]

What would you prefer?




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]

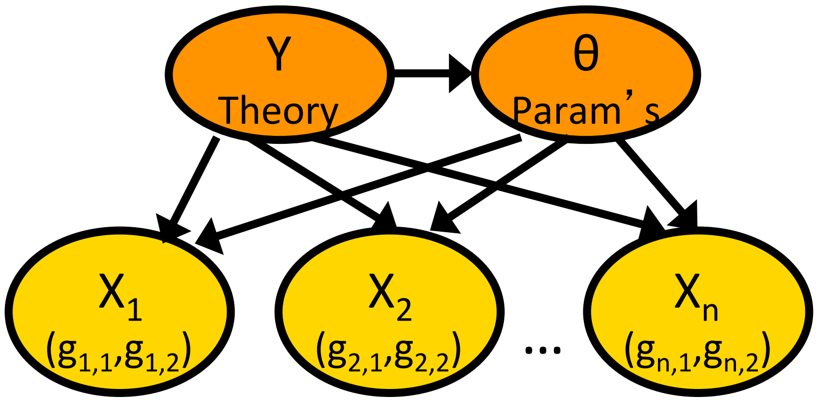
**How should we design tests to distinguish theories?**

# Iowa Gambling as BED

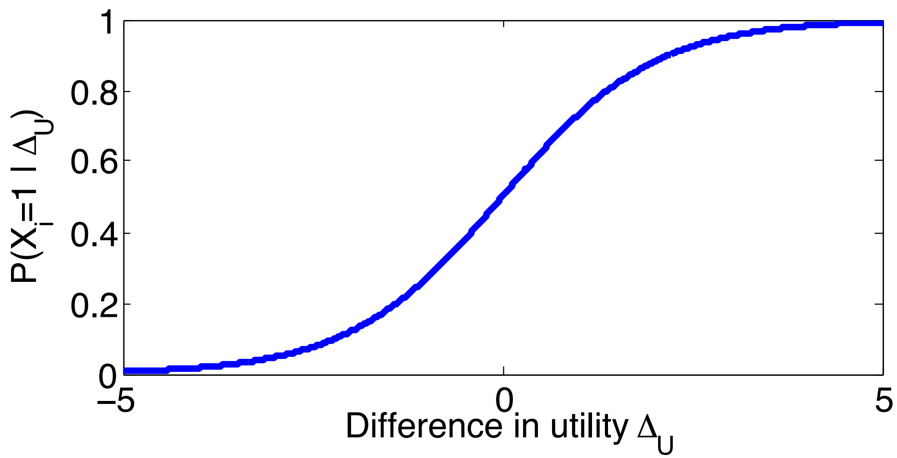
Every possible test $X_s = (g_{s,1}, g_{s,2})$ is a pair of gambles

Theories parameterized by θ

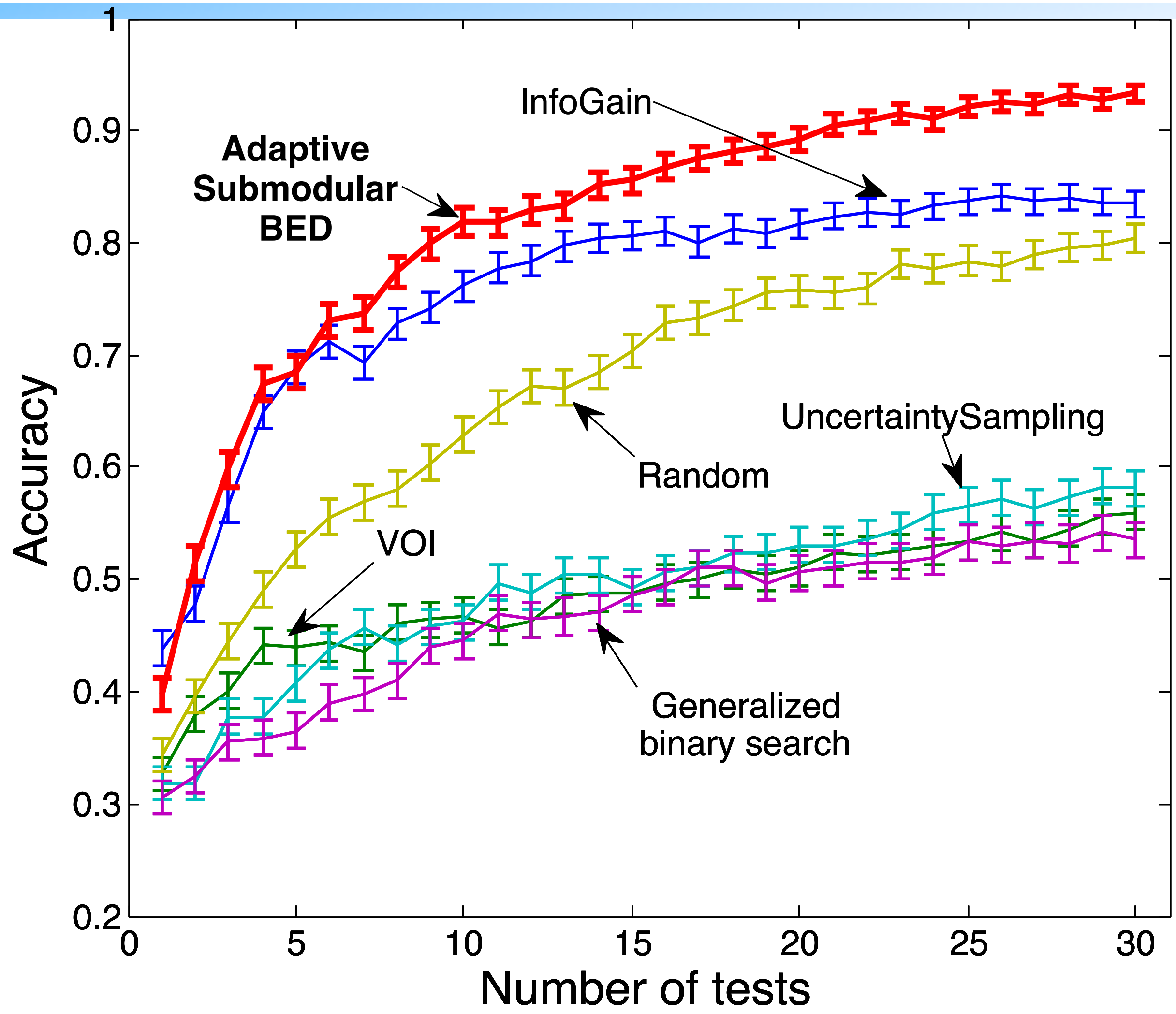
Each theory predicts utility for every gamble U(g,y,θ)




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

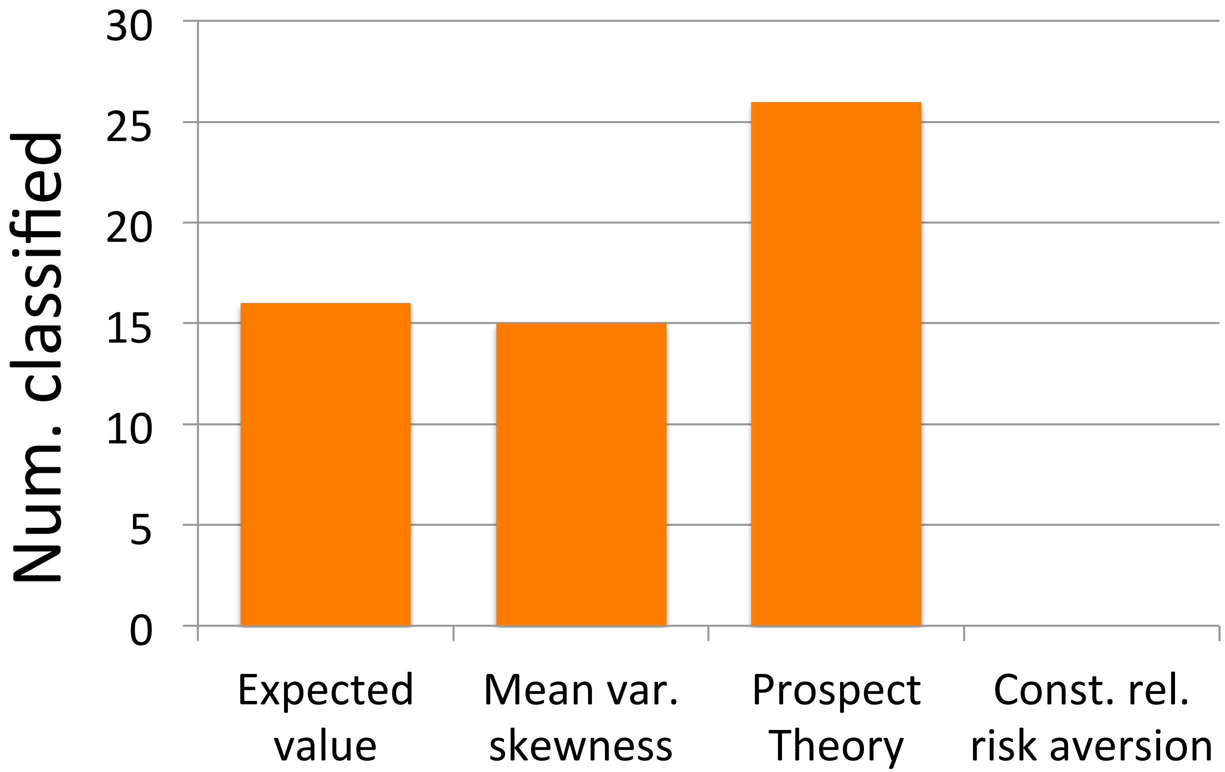


# Simulation Results




**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 ☹**

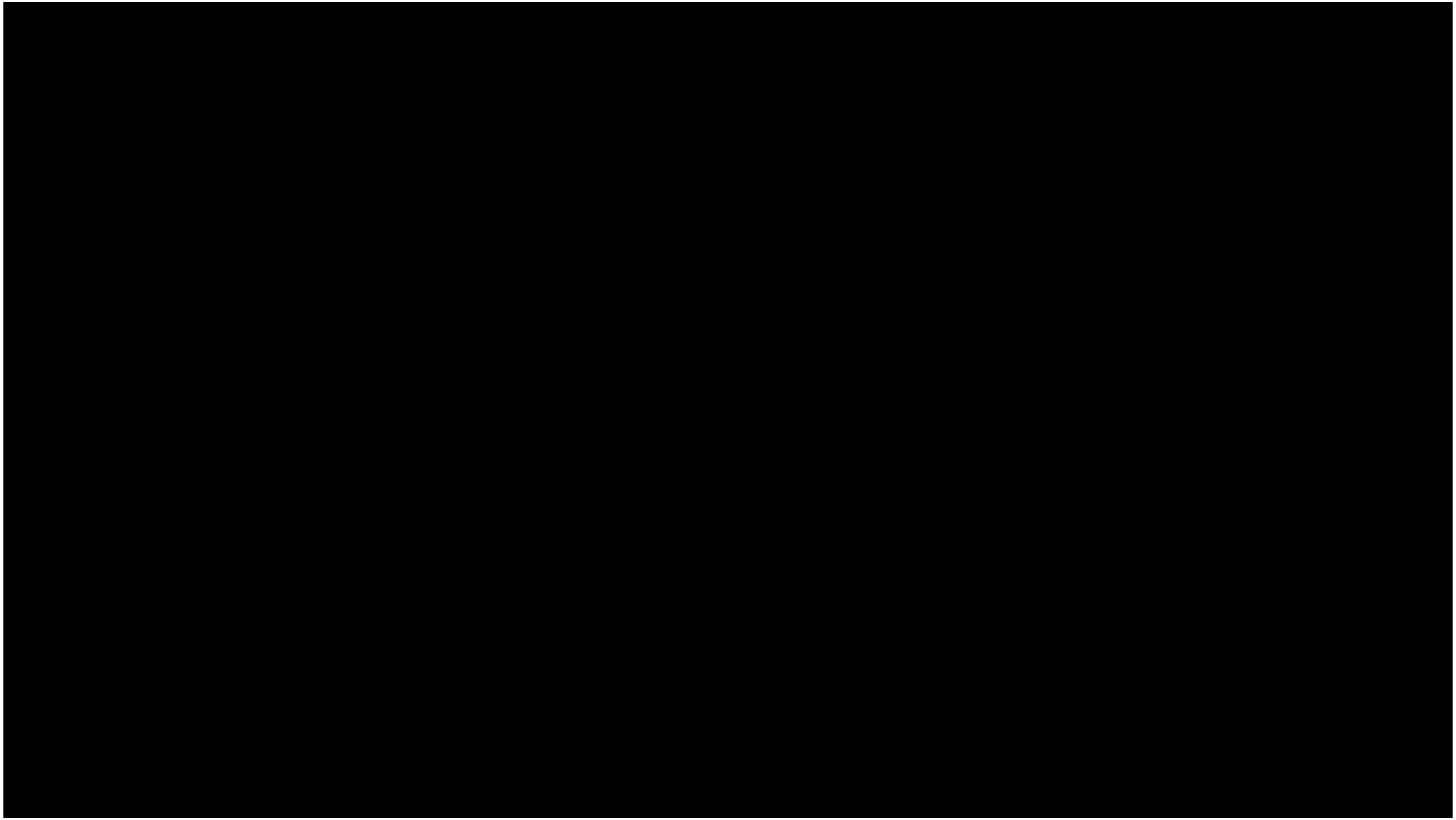
**Using lazy evaluations:**

**<5s per test ☺**

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

# Application: Touch-based localization

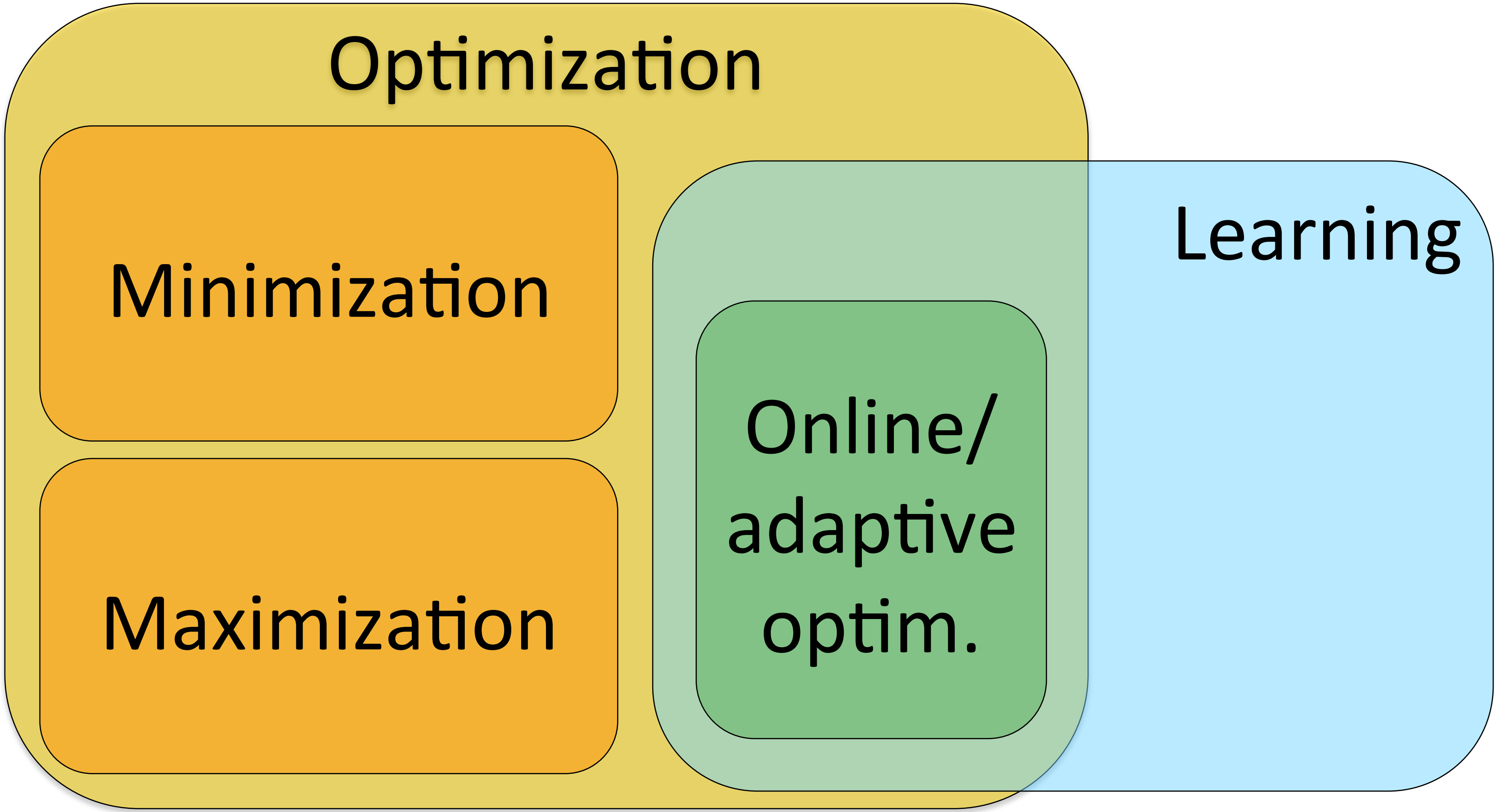
[Javdani, Klingensmith, Bagnell, Pollard, Srinivasa, ICRA 2013]



# 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]

# What to do with submodular functions



# 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
  - Submodular-supermodular procedure / semigradient methods
- Structured prediction with submodular functions

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