Massachusetts Institute of Technology

Submodular Functions and Machine Learning

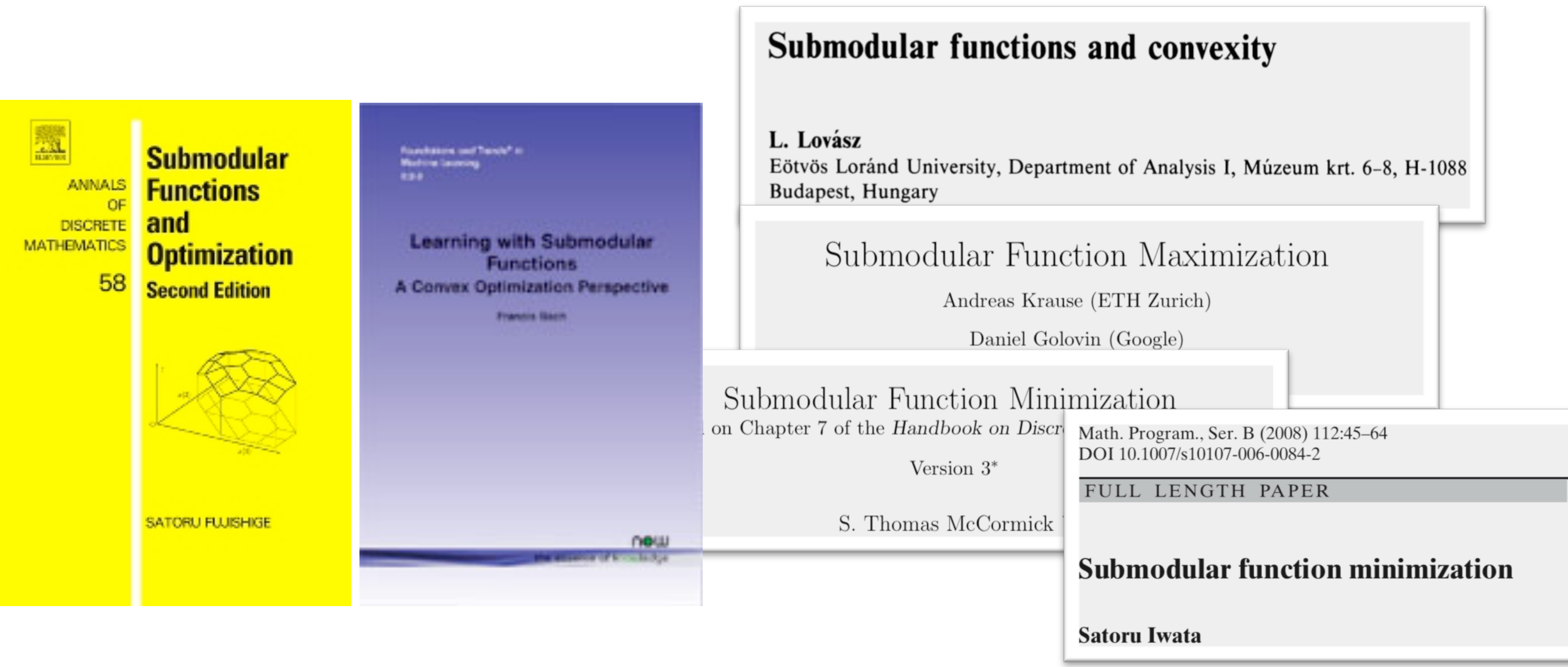
MLSS Kyoto

Stefanie Jegelka

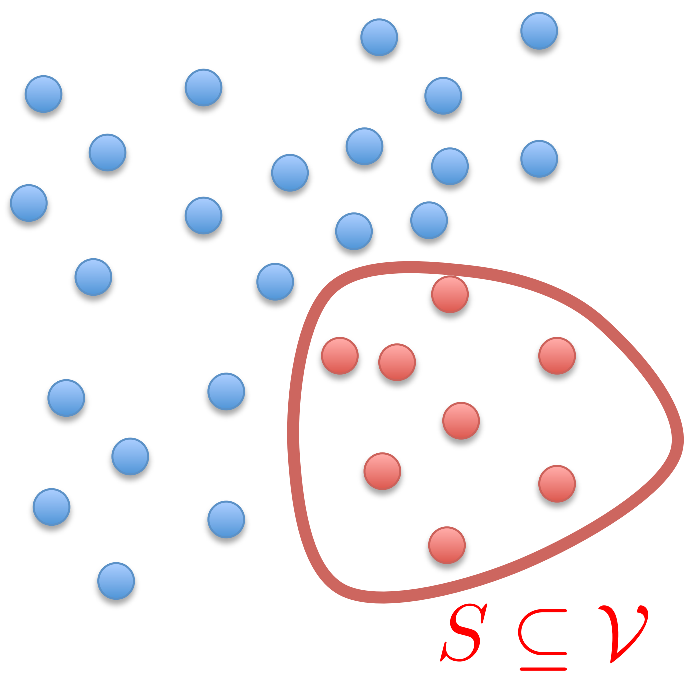
MIT

Resources

- submodularity.org
- people.csail.mit.edu/stefje/mlss/literature.pdf
  references for the lectures, pointers to surveys, papers, books
- discml.cc talks on submodularity in machine learning



Setup



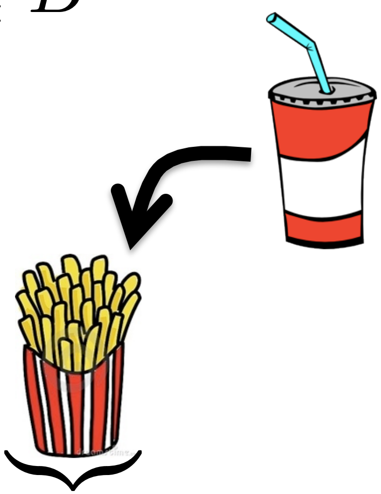
- ground set $\mathcal{V}$
- (scoring) function

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

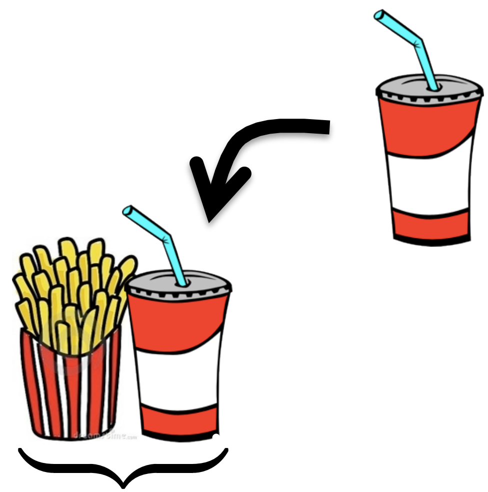
$$\max \quad F(S)$$

Submodularity

$A \subseteq B$



A



B

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

extra cost:
one drink

extra cost:
free refill ☺

diminishing marginal costs

Roadmap

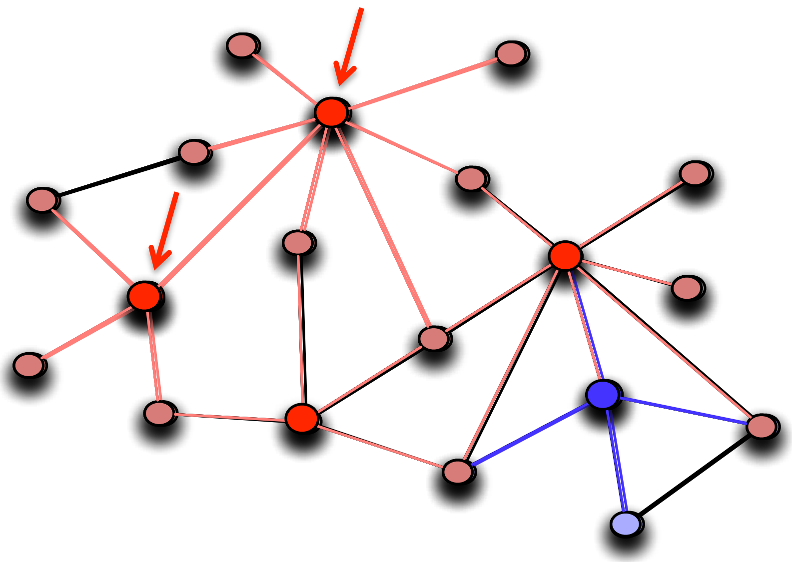
- Submodular set functions
  - links to convexity
  - special polyhedra

- Minimizing submodular functions
  - general and special cases
  - constraints

- Maximizing submodular functions
  - monotone & non-monotone
  - repulsive point processes

Maximizing Influence

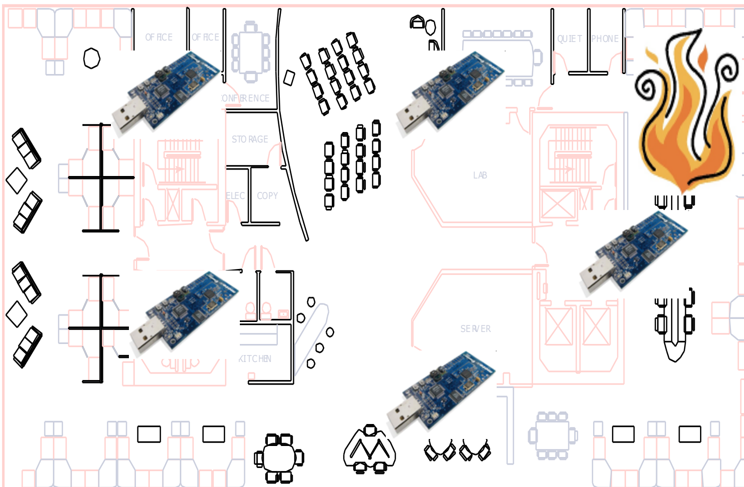
$$F(S) = \text{expected \# infected nodes}$$



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

Kempe, Kleinberg & Tardos 2003

Informative Subsets

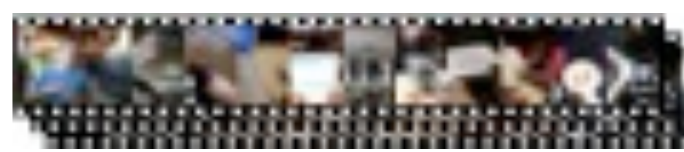


- where put sensors?
- which experiments?
- summarization

$$F(S) = \text{“information”}$$

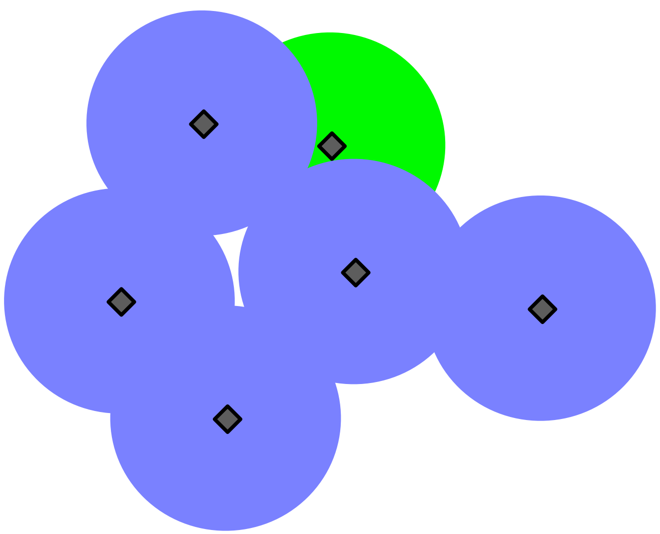
Summarization

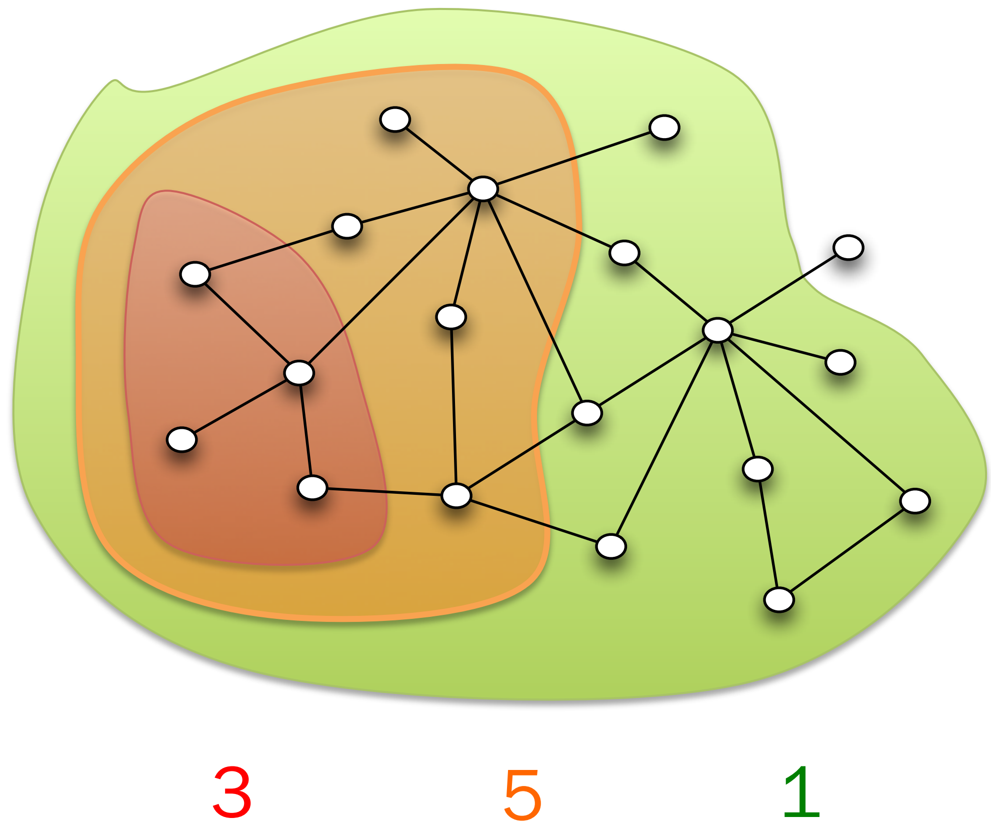
- videos, text, pictures …
- would like:
  relevance, reliability, diversity



Monotonicity

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





Maximizing monotone functions

$$\text{if } A \subseteq B \quad \text{then} \quad F(A) \leq F(B)$$

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

- NP-hard
- approximation: greedy algorithms

Maximizing monotone functions

$$\max_{S} \; F(S) \;\; \text{s.t.} \;\; |S| \leq k$$

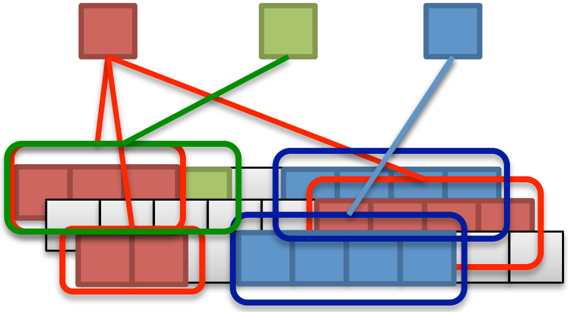
- greedy algorithm:

$S_0 = \emptyset$

for *i* = *0, …, k-1*

$$e^* = \arg \max_{e \in \mathcal{V} \setminus S_i} F(S_i \cup \{e\})$$

$$S_{i+1} = S_i \cup \{e^*\}$$



How good is greedy? ... in theory

$$\max_S \; F(S) \;\; \text{s.t.} \;\; |S| \leq k$$

Theorem *(Nemhauser, Fisher, Wolsey `78)*

F monotone submodular, S_k solution of greedy. Then

$$F(S_k) \;\geq\; \left(1 - \frac{1}{e}\right) F(S^*)$$

optimal solution

in general, no poly-time algorithm can do better than that!

Questions

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

More complex constraints: budget

$$\max \quad F(S) \ \text{ s.t. } \sum_{e \in S} c(e) \leq B$$

1. run greedy: S_{gr}
2. run a modified greedy: S_{mod}

$$e^* = \arg\max \frac{F(S_i \cup \{e\}) - F(S_i)}{c(e)}$$

3. pick better of S_{gr} , S_{mod}

➔ approximation factor: $\frac{1}{2}\left(1 - \frac{1}{e}\right)$

even better but less fast: partial enumeration *(Sviridenko, 2004)* or filtering *(Badanidiyuru & Vondrák 2014)*

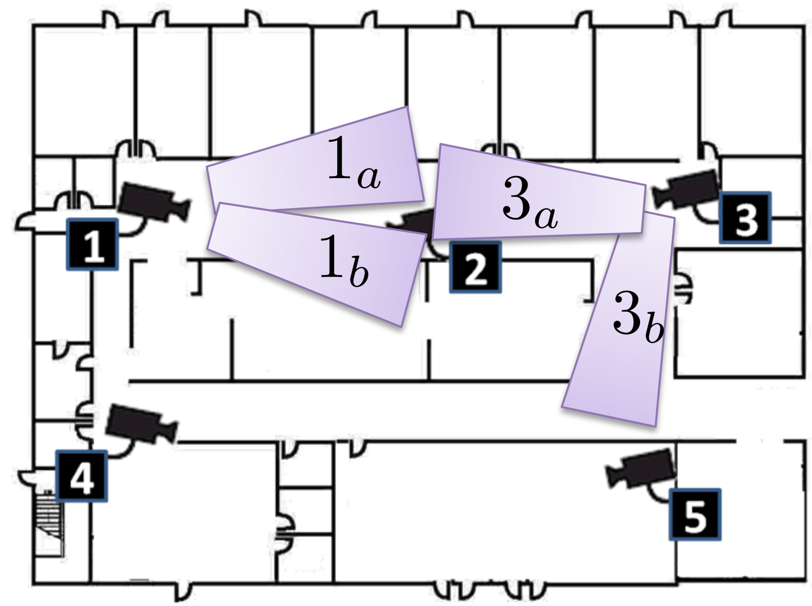
(Leskovec et al 2007)

Example: Camera network

- Ground set: $V = \{1_a, 1_b, \ldots, 5_a, 5_b\}$
- Sensing quality model: $F : 2^V \to \mathbb{R}$

- Configuration (subset) is feasible if no camera is pointed in two directions at once

(partition) matroid constraint!

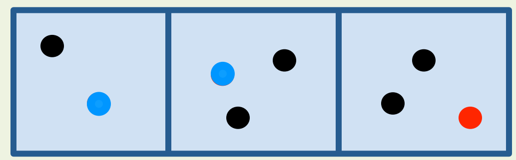


Matroids (semi-formally)

S is independent (= feasible) if ...

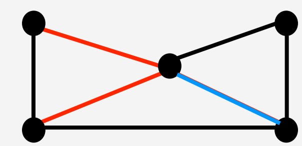
... $|S| \leq k$

Uniform matroid



... S contains at most one element from each square

Partition matroid



... S contains no cycles

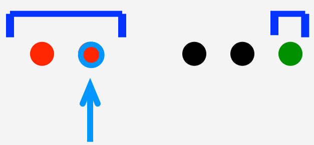
Graphic matroid

matroid properties:

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

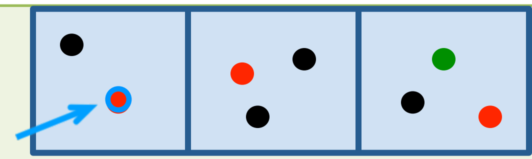
Matroids

S is independent (=feasible) if …



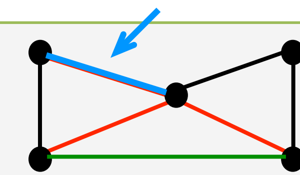
… |S| ≤ k

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

Example: Camera network

- Ground set: $V = \{1_a, 1_b, \ldots, 5_a, 5_b\}$
- Sensing quality model: $F : 2^V \to \mathbb{R}$

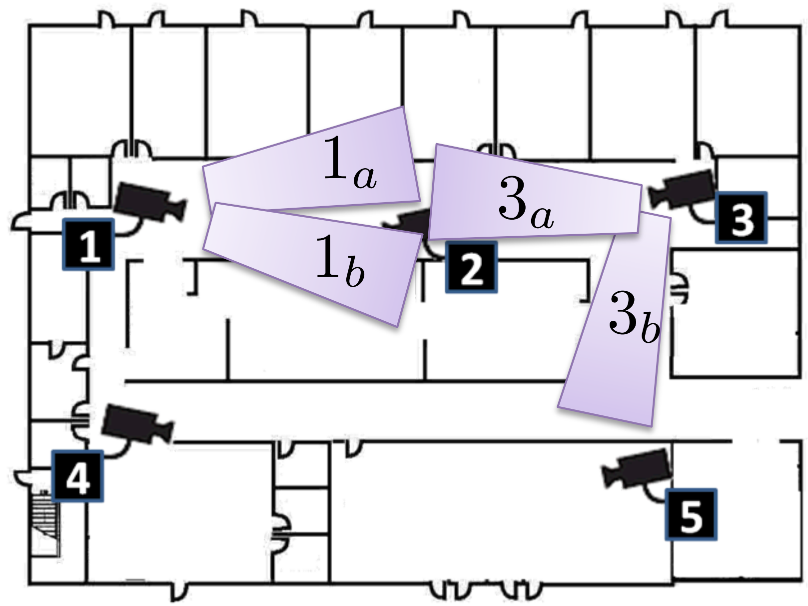
- Configuration (subset) is feasible if no camera is pointed in two directions at once

(partition) matroid constraint:

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

require:

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



Greedy algorithm for matroids

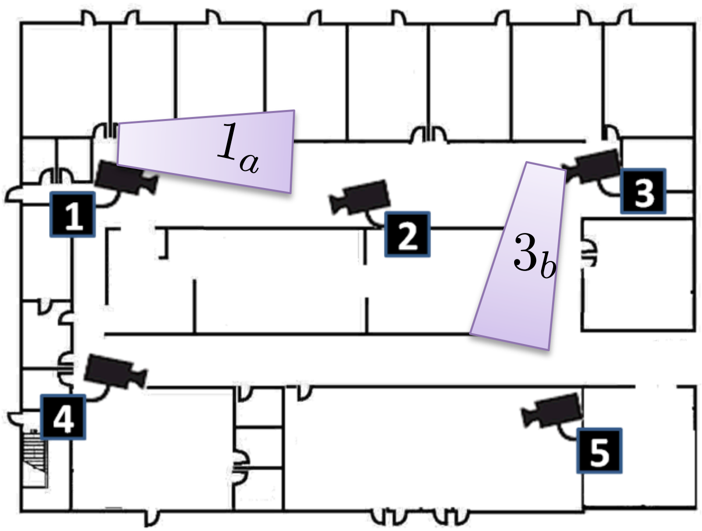
$S = \emptyset$

While $\exists e : S \cup e$ independent

$$S \leftarrow S \cup \underset{e: S \cup e \text{ indep.}}{\operatorname{argmax}} F(S \cup e)$$

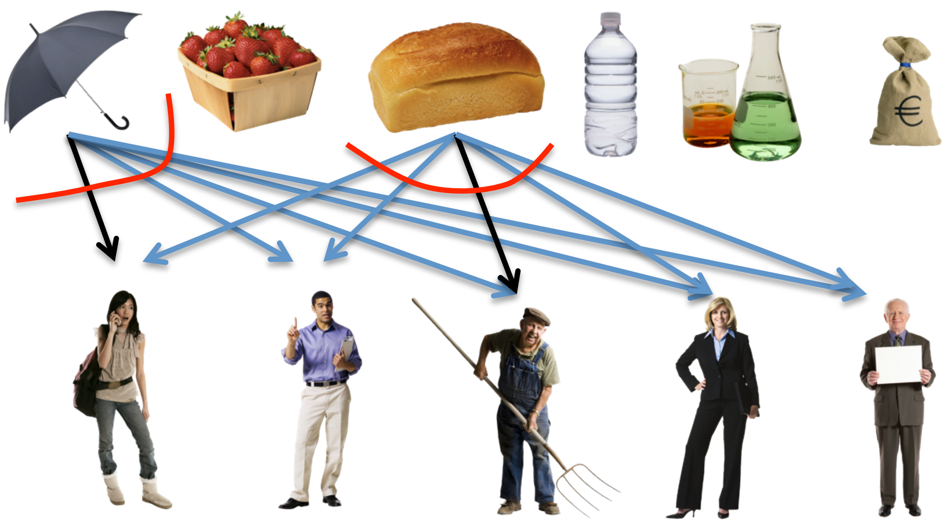
Theorem *(Nemhauser, Wolsey, Fisher 78)*
For monotone submodular functions:

$$F(S_{\text{greedy}}) \geq \frac{1}{2} F(S^*)$$



better approximation (1-1/e): relaxation

Submodular welfare



- assign set S_i to person i to maximize

$$\sum_{i=1}^{k} F_i(S_i)$$

- $\mathcal{V} =$ all possible assignments

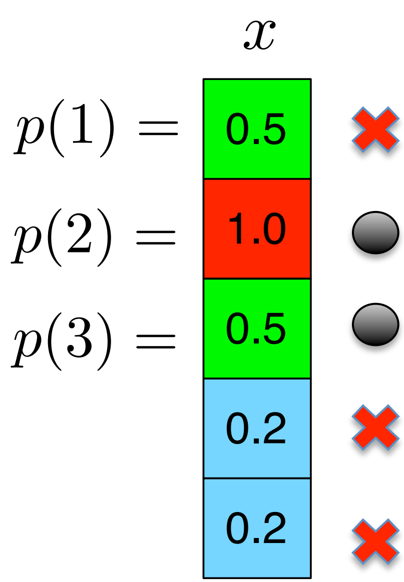
- partition matroid: assign each item only once

Relaxation?

- concave analog of Lovasz extension: not in polynomial time ☹
- multi-linear extension: probability distribution from *x*
  sample element e with probability x_e

$$f_M(x) = \sum_{S \subseteq \mathcal{V}} F(S) \prod_{e \in S} x_e \prod_{e \notin S} (1 - x_e)$$

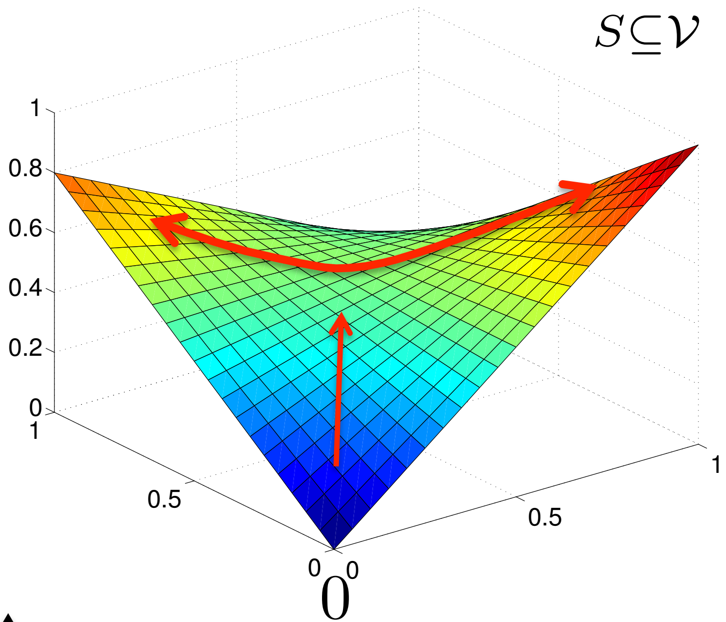
$$= \mathbb{E}_{S \sim x} \left[F(S) \right]$$



(Calinescu-Chekuri-Pal-Vondrak 2011)

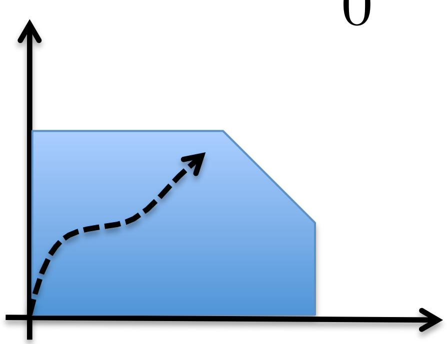
Multilinear extension

$$f_M(x) = \sum_{S \subseteq \mathcal{V}} F(S) \prod_{e \in S} x_e \prod_{e \notin S} (1 - x_e)$$



1. concave in positive directions: $f_M(x + \lambda d)$ concave function of λ if $d \succeq 0$.

2. convex in swap directions: $f_M(x + \lambda d)$ convex function of λ if $d = 1_i - 1_j$



➔ Optimization: *continuous greedy*
move in directions

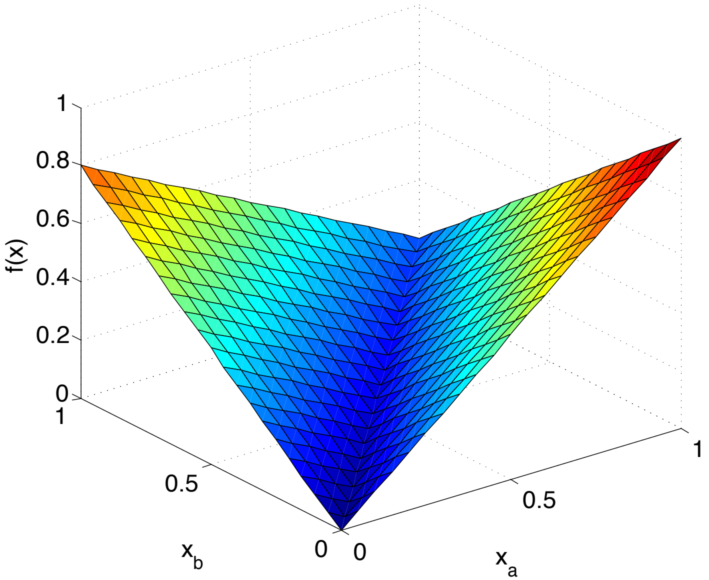
$$v = \arg\max_{v \in P} \; v^\top \nabla f_M(x^t)$$

Relaxation: algorithm

1. approximately maximize f_M (Frank-Wolfe like algorithm)
2. round (pipage rounding)

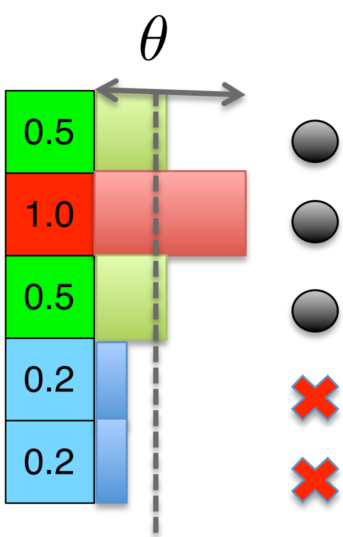
(Calinescu-Chekuri-Pal-Vondrak 2011)

Lovász extension as expectation



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

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



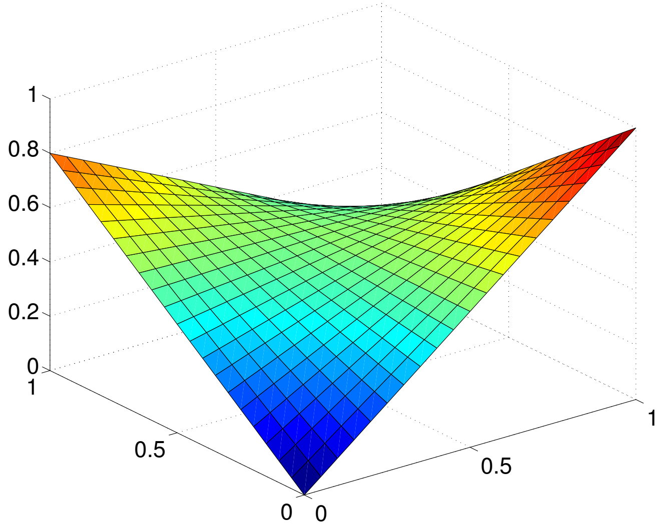
e.g. $\theta = 0.4$

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

$$= \alpha_i F(S_i)$$

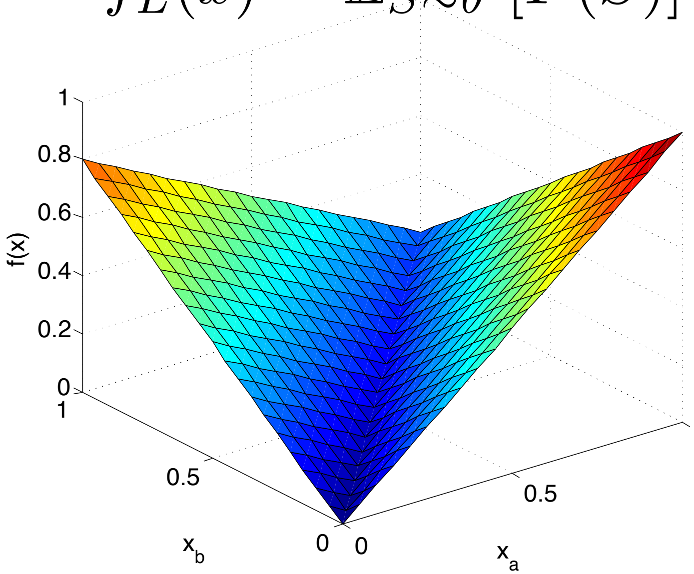
Multilinear relaxation vs. Lovász ext.

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



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

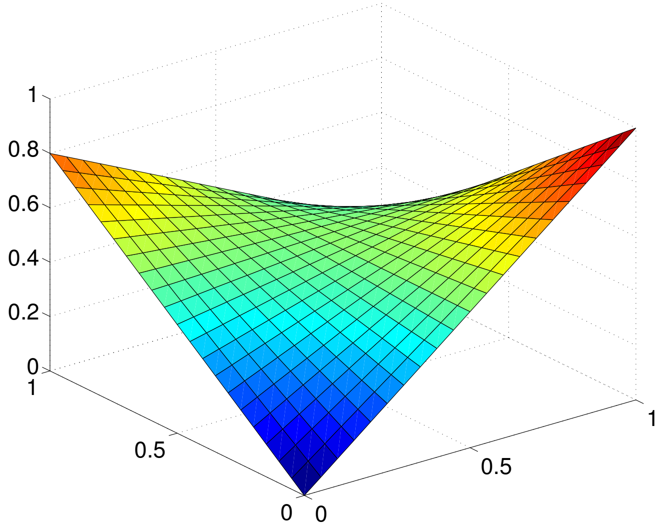
$$f_L(x) = \mathbb{E}_{S \sim \theta}\left[F(S)\right]$$



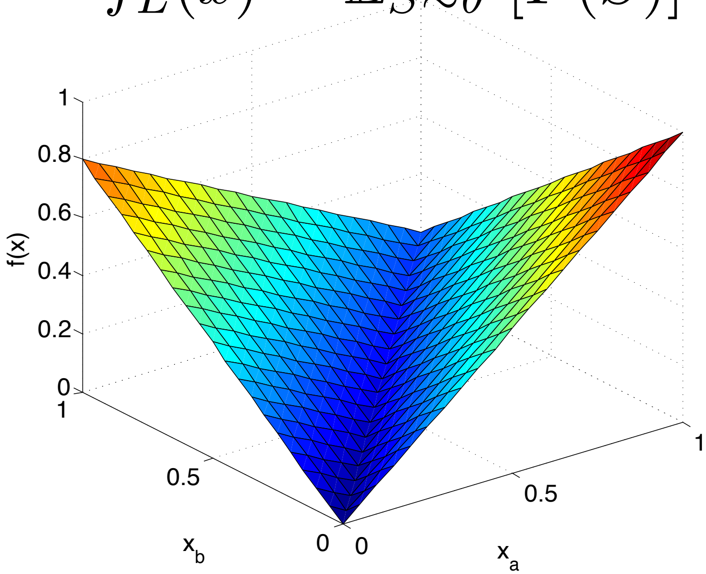
- convex
- computable in O(n log n)

Multilinear relaxation vs. Lovász ext.

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



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



example: cut function

$$f_M(x) = x_u + x_v - 2x_u x_v$$

$$f_L(x) = |x_u - x_v|$$

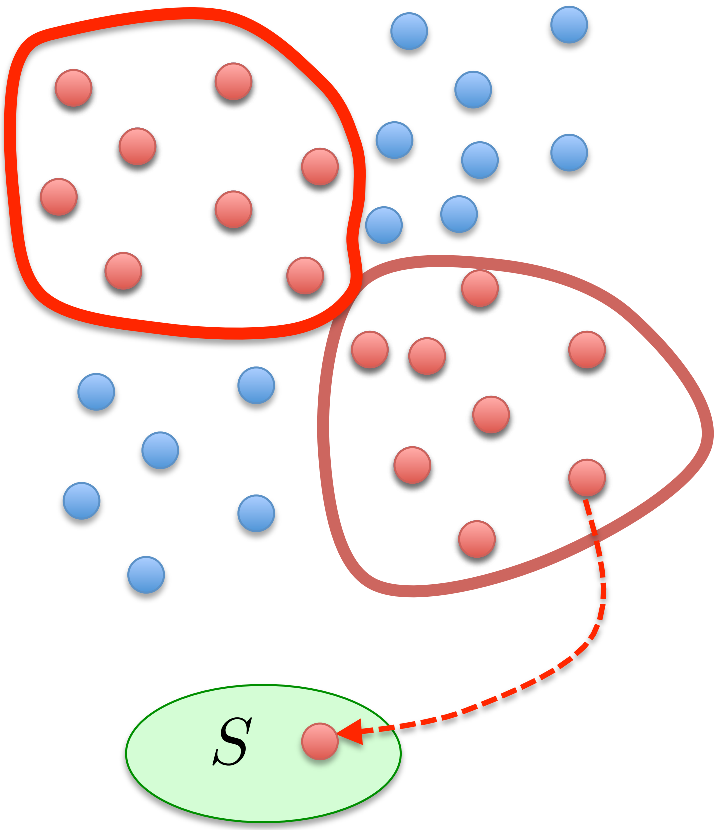
Questions

- What if I have more complex constraints?
  - budget constraints
  - matroid constraints

- Greedy takes *O(nk)* time. What if *n*, *k* are large?
  - faster sequential algorithms
  - filtering
  - parallel / distributed

- What if my function is not monotone?

Making greedy faster: stochastic



approximation factor:

$$F(S_k) \geq (1 - \frac{1}{e} - \epsilon) F(S^*)$$

$$\max_S \; F(S) \; \text{ s.t. } |S| \leq k$$

for i=1...k:

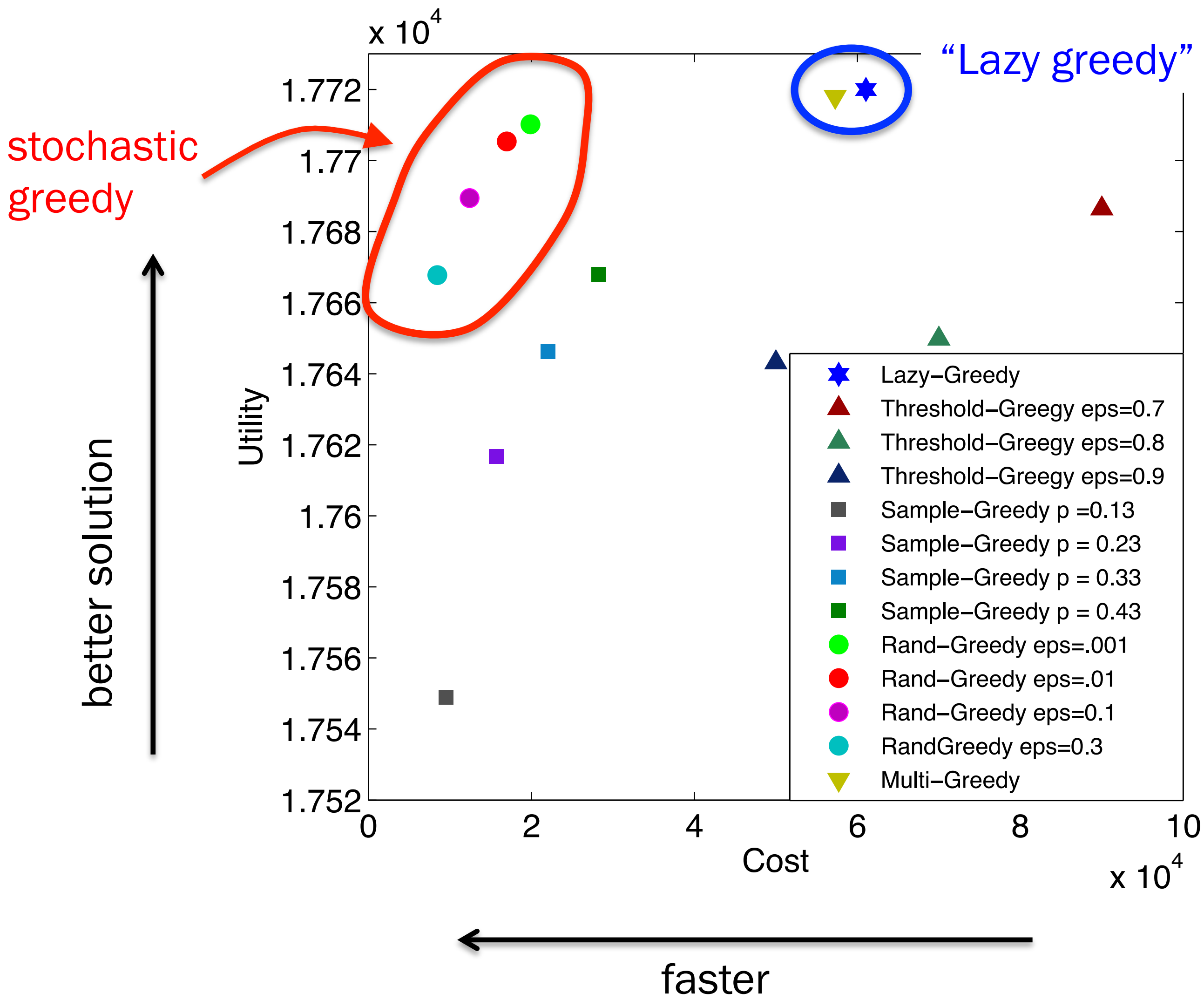
- randomly pick set *T* of size $\frac{n}{k} \log \frac{1}{\epsilon}$
- find best *a* element in *T* and add

$$a_i = \arg\max_{a \in T} F(a | S_{i-1})$$

$$S_i \leftarrow S_{i-1} \cup \{a_i\}$$

(Mirzasoleiman et al 2014)

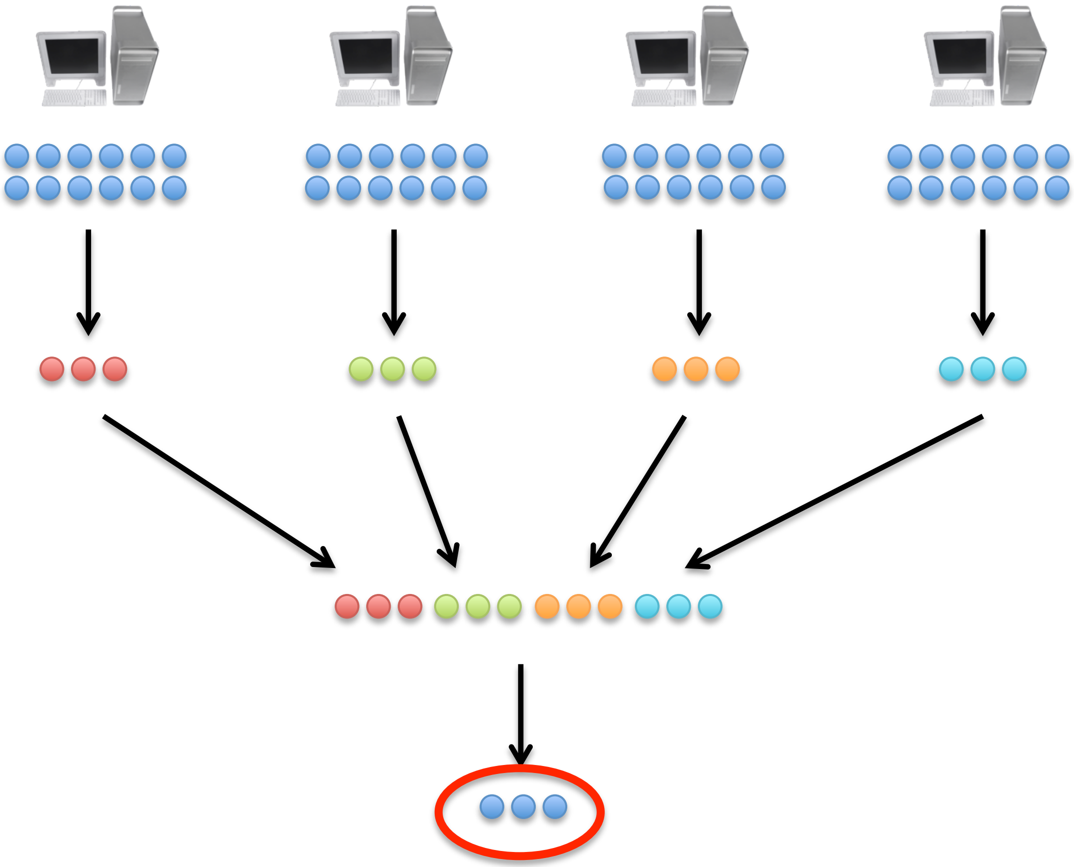
Performance



even more data ...

distributed greedy algorithm?

Distributed greedy algorithms



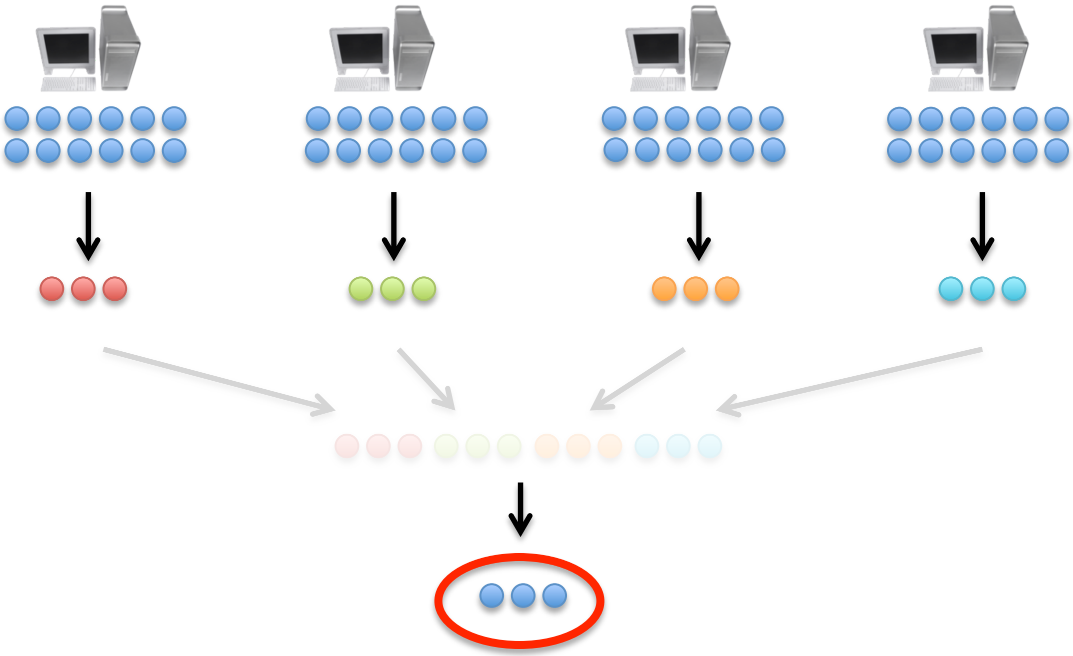
greedy is sequential.
pick in parallel??

pick *k* elements
on each machine.

combine and run
greedy again.

Is this useful?

Distributed greedy algorithms



pick in parallel
from *m* machines

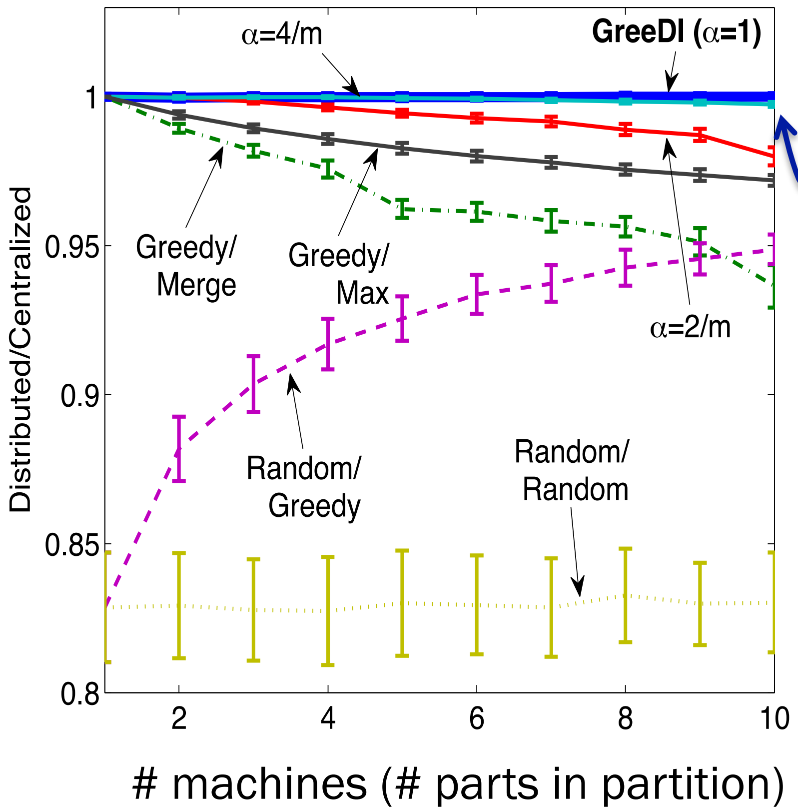
Is this useful?

Approximation factor:

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

(*Mirzasoleiman et al 2013*)

Distributed Greedy

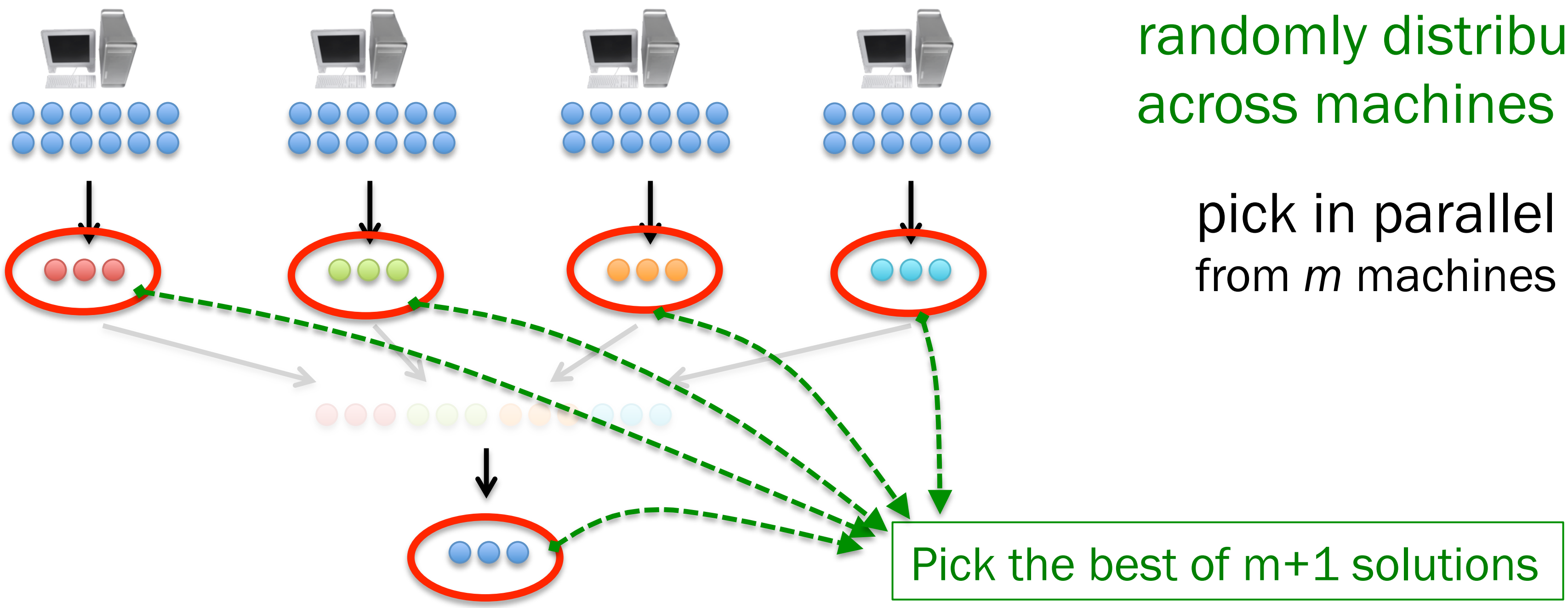


In practice, performs often quite well.

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

(Mirzasoleiman et al 2013)

Distributed greedy algorithms

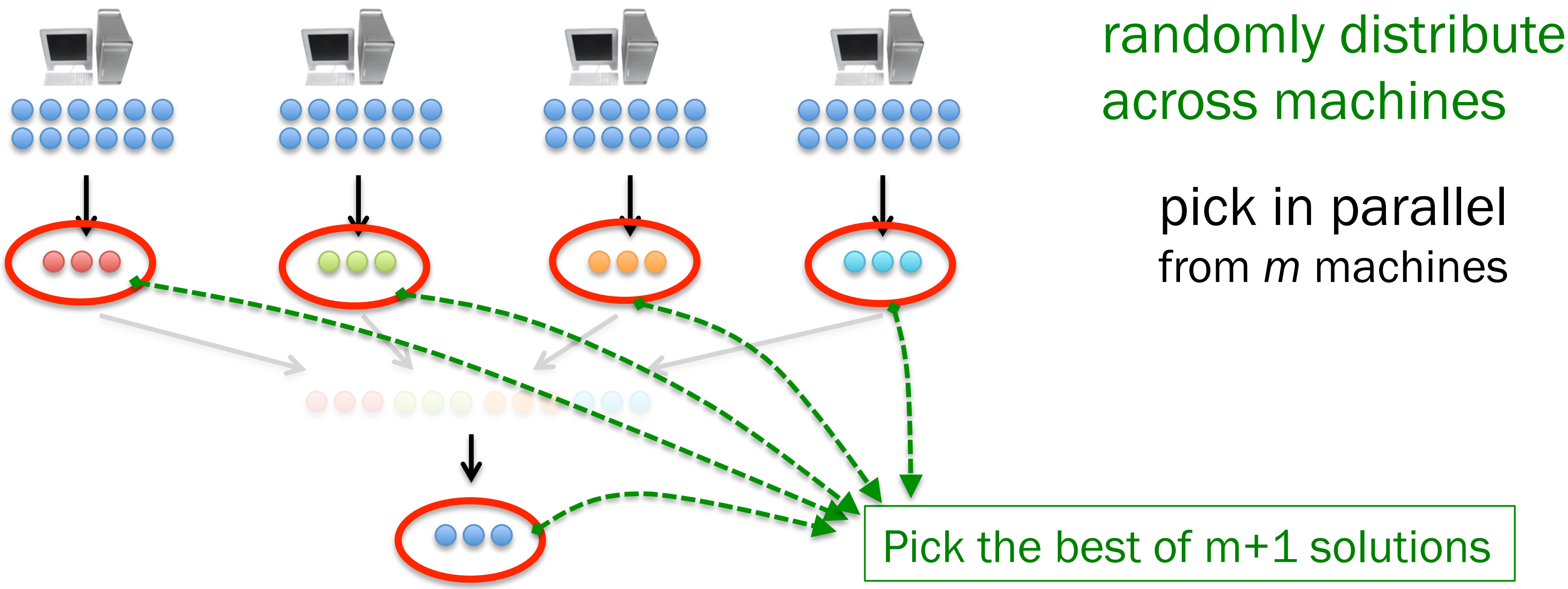


- each machine: $\alpha-$approximation algorithm
- level 2: $\beta-$ approximation algorithm

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

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

Distributed greedy algorithms



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

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

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

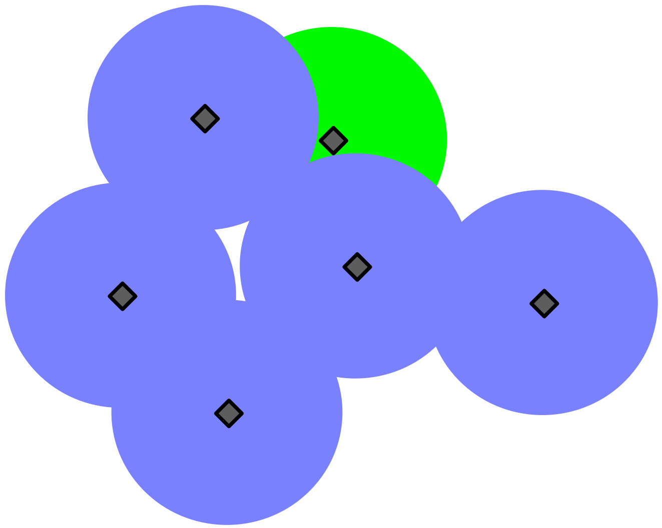
(Mirzasoleiman et al 2013, de Ponte Barbosa et al 2015, see also Mirrokni, Zadimoghaddam 2015)

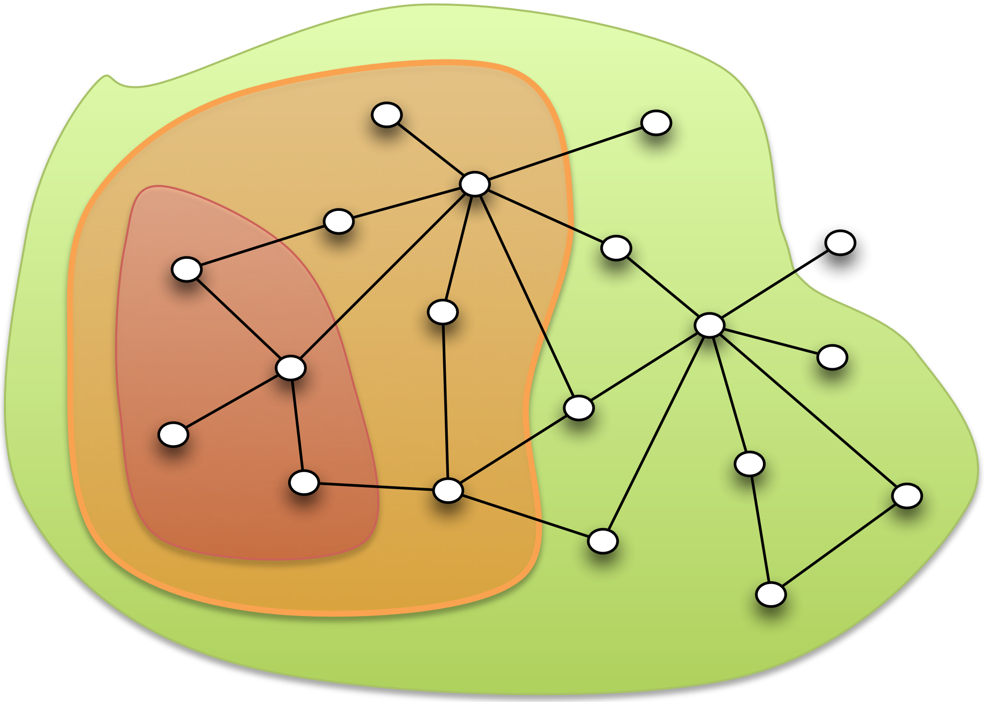
Questions

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

Non-monotone functions

~~if $S \subseteq T$ then $F(S) \leq F(T)$~~





3 5 1

still assume:

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

Picking at random

- Let F be a non-monotone nonnegative submodular function. Pick set S uniformly at random from $\mathcal{V}$

$$\Pr(\text{ include i}) = 1/2 \text{ for all i}$$

- Then

$$\mathbb{E}[F(S)] \geq \frac{1}{4} F(S^*)$$

- If F is symmetric:

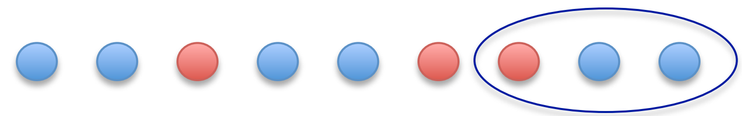
$$\mathbb{E}[F(S)] \;\geq\; \frac{1}{2} F(S^*)$$

Picking at random

- Can we do this for constrained (monotone) maximization?

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

- Example:



$$|R| = k$$

$$F(S) = |S \cap R| + \epsilon \cdot \min\{|S \cap B|, 1\}$$

$$F(S^*) = F(R) = k$$

- Pick k elements at random: will hit very few red ones

$$\mathbb{E}[F(S)] < \left(\frac{k+\epsilon}{n}\right) F(S^*)$$

Non-monotone maximization

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

Can we do better than completely random?

$$\mathbb{E}[F(S)] \geq \frac{1}{4} F(S^*)$$

Greedy can fail ...

greedy

$F(A)$

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

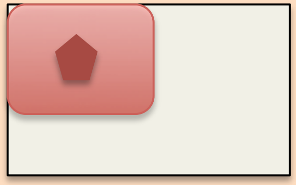
optimal solution

$$F(A) = 95$$

sensor 1

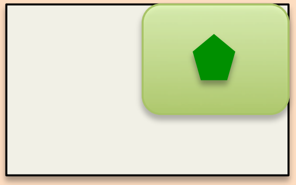
coverage: 100
cost: -60
gain 40

sensor 2



coverage: 30
cost: - 1
gain 29

sensor 3



coverage: 30
cost: - 1
gain 29

sensor 4

coverage: 40
cost: - 3
gain 37

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

Greedy can fail ...

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

greedy solution:

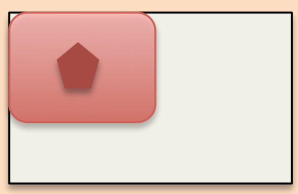
$$F(A) = 40$$

optimal solution: $F(A) = 95$

sensor 1

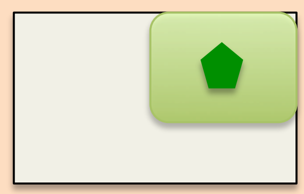
coverage: 100
cost: -60
gain 40

sensor 2



coverage: 30
cost: - 1
gain 29

sensor 3

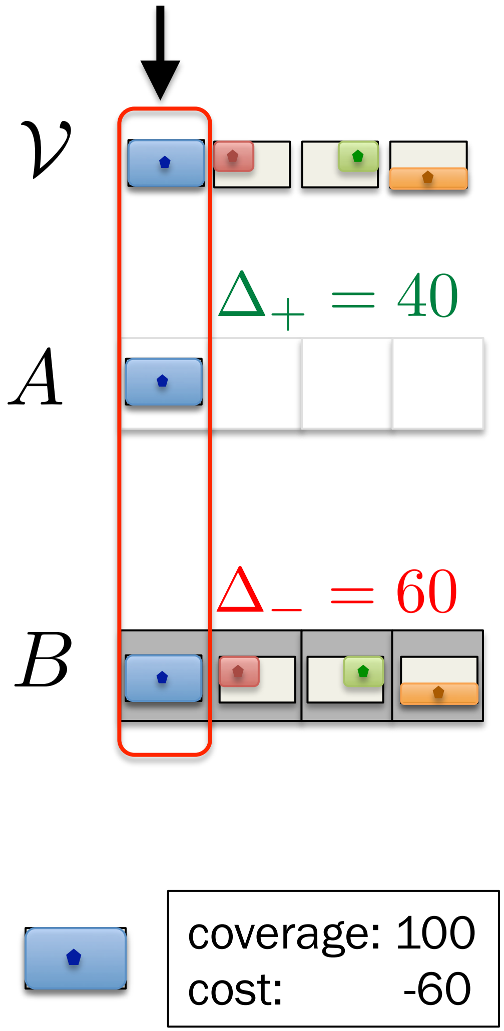


coverage: 30
cost: - 1
gain 29

sensor 4

coverage: 40
cost: - 3
gain 37

Double (bidirectional) greedy



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

for *i=1, …, n* *//add or remove?*

- gain of adding (to A):

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

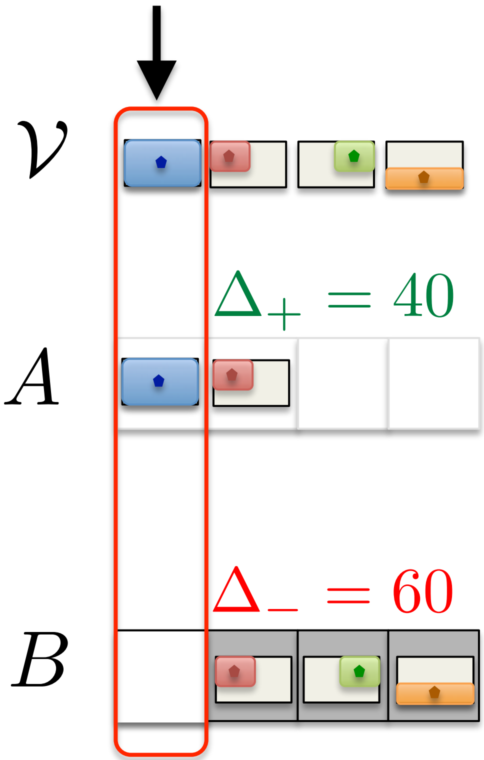
- gain of removing (from B):

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

add with probability

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

Double (bidirectional) greedy



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

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

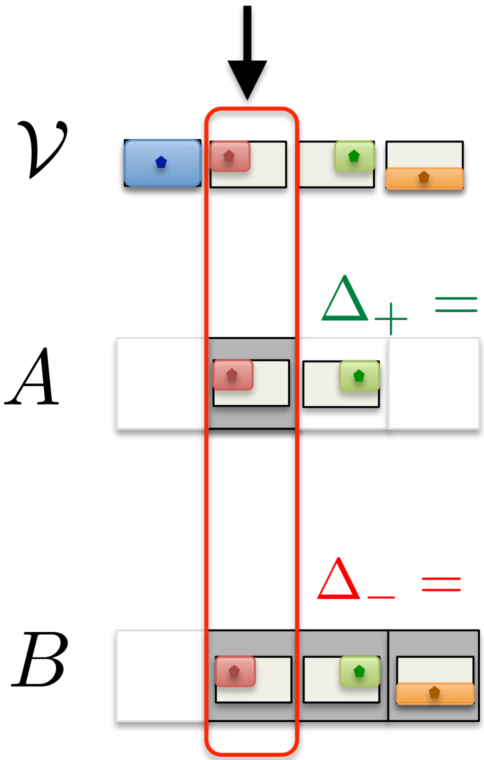
add with probability

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

add to A or remove from B

coverage: 100
cost: -60

Double (bidirectional) greedy



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

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

$\Delta_+ = 29$

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

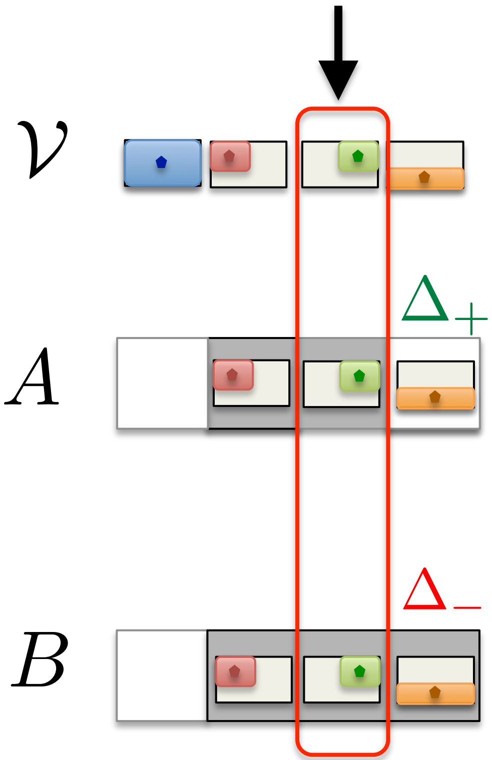
add with probability

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

add to A or remove from B

coverage: 30
cost: - 1

Double (bidirectional) greedy



$\Delta_+ = 29$

$\Delta_- = 0$

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

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

add with probability

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

add to A or remove from B

coverage:	30
cost:	- 1

Double greedy

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

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

F submodular, S_g solution of double greedy. Then

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

optimal solution

Non-monotone maximization

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

- constraints?
  possible, but different algorithms

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

Submodular maximization: summary

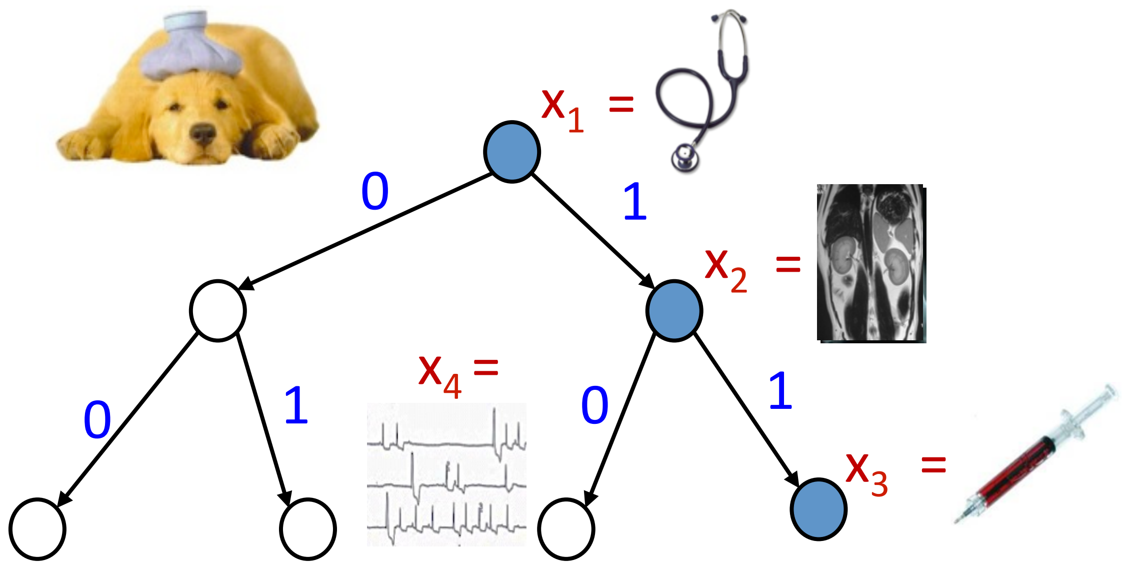
- many applications: diverse, informative subsets

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

- several constraints possible with constant approximation factors
(monotone and non-monotone)

Adaptive/sequential settings

Sequential diagnosis:



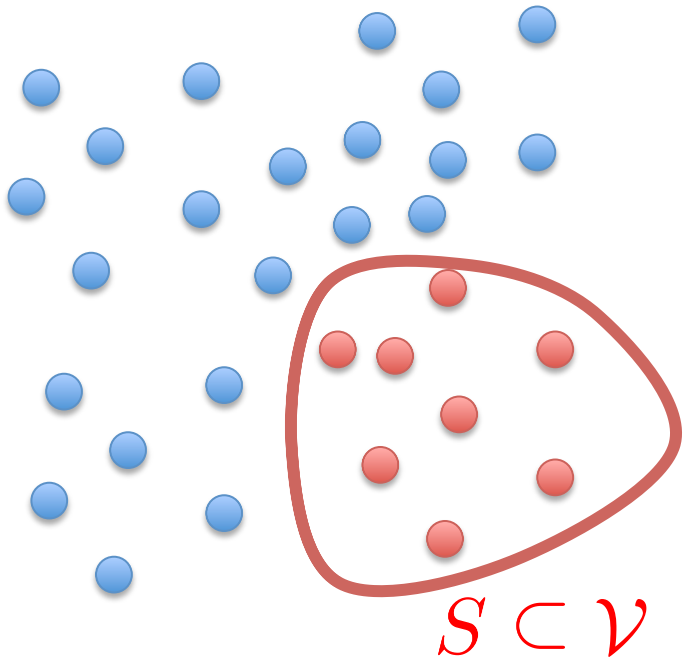
- learning a policy: model updated after observation
- submodularity does not apply directly
- suitable generalization: adaptive submodularity
  greedy results generalize ☺

(Golovin & Krause 2013)

Roadmap

- Submodular set functions
  - links to convexity
  - special polyhedra
- Minimizing submodular functions
  - general and special cases
  - constraints
- Maximizing submodular functions
  - monotone & non-monotone
  - repulsive point processes

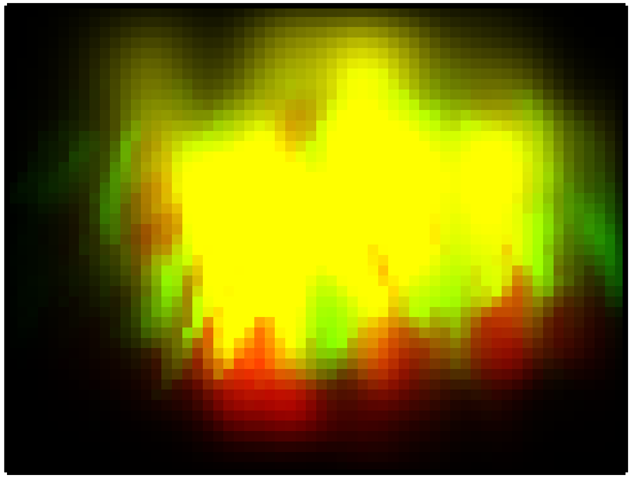
Diversity and distributions



Point process:
distribution over sets

$$P(S)$$

Diversity priors



$$P(S \mid \text{data}) \; \propto \; P(S) \; P(\text{data} \mid S)$$

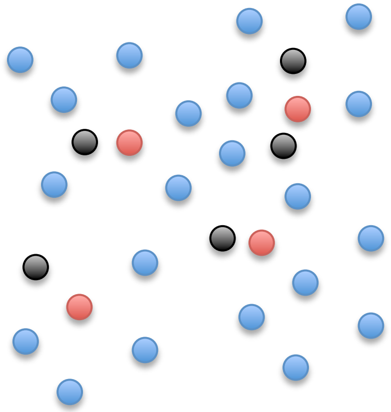
“spread out”

Point processes – examples

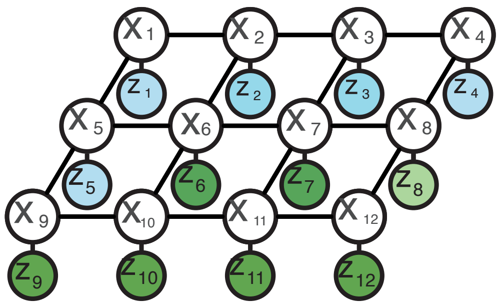
- independent coin flips

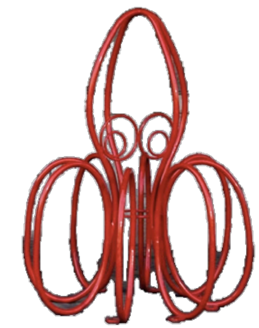
$$P(Y = S) = \prod_{i \in S} p_i \prod_{j \notin S} (1 - p_j)$$

- if $S \cap T = \emptyset$
  then $Y \cap S$ and $Y \cap T$ are independent



Point processes – examples





$$P(x|z) \propto P(z|x)\, P(x) \qquad x \in \{0,1\}^n$$

labels   pixel values

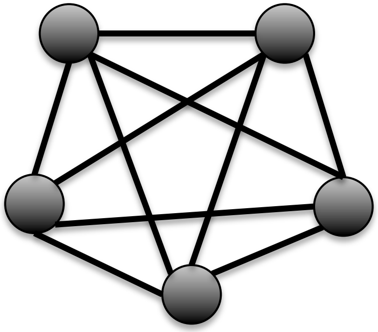
$$\propto \exp\Big\{ -\big(\sum_i \beta_i x_i + \sum_{ij} \nu_{ij} x_i x_j\big)\Big\}$$

our examples: spatial coherence, “attractive” --- positive correlations

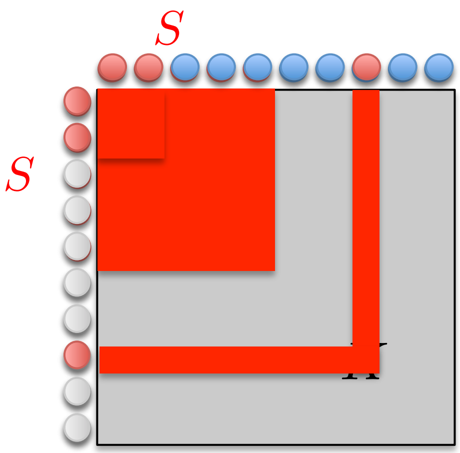
Repulsion?

in a graphical model:

- computationally hard
- dependencies between all elements ➔ fully connected



Determinantal point processes



- normalized similarity matrix K
- sample *Y*:

$$P(S \subseteq Y) = \det(K_S)$$

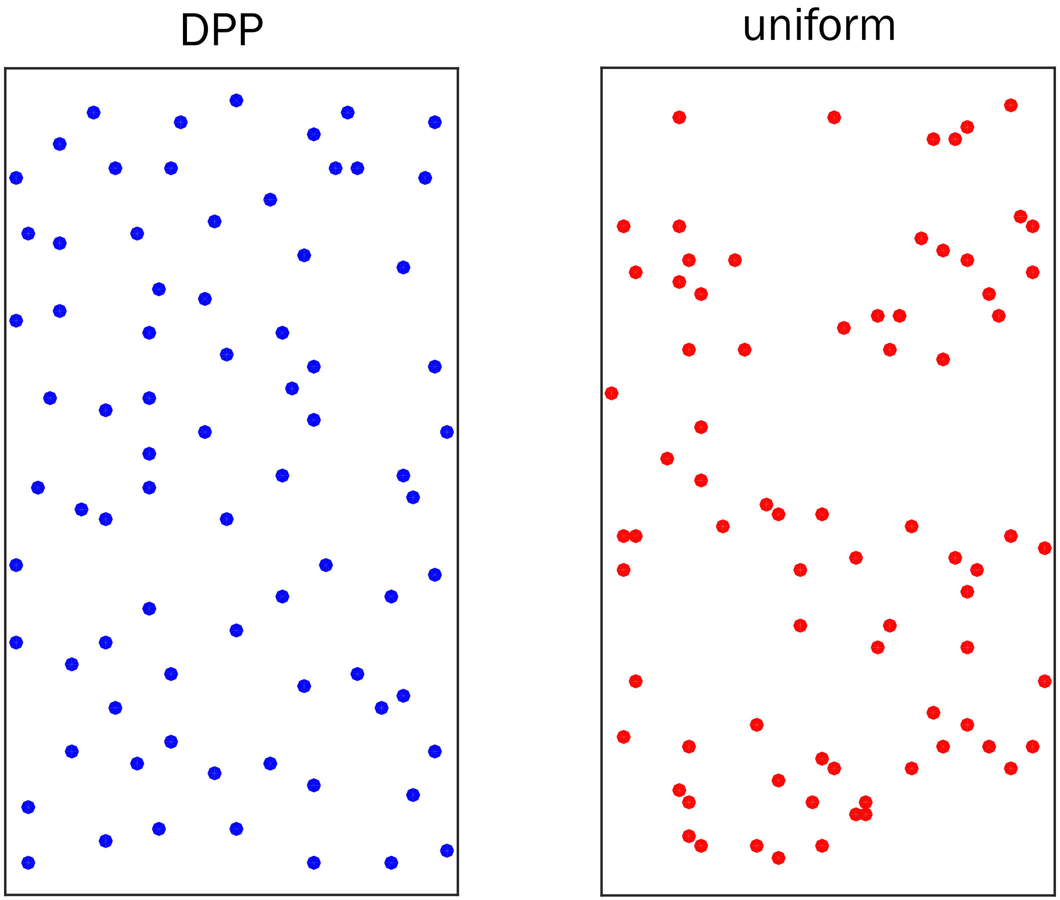
$$P(e_i \in Y) = K_{ii}$$

$$P(e_i, e_j \in Y) = K_{ii}K_{jj} - K_{ij}^2$$
$$= P(e_i \in Y)P(e_j \in Y) - K_{ij}^2$$

repulsion

$F(S) = \log\det(K_S)$ is submodular

DPP sample

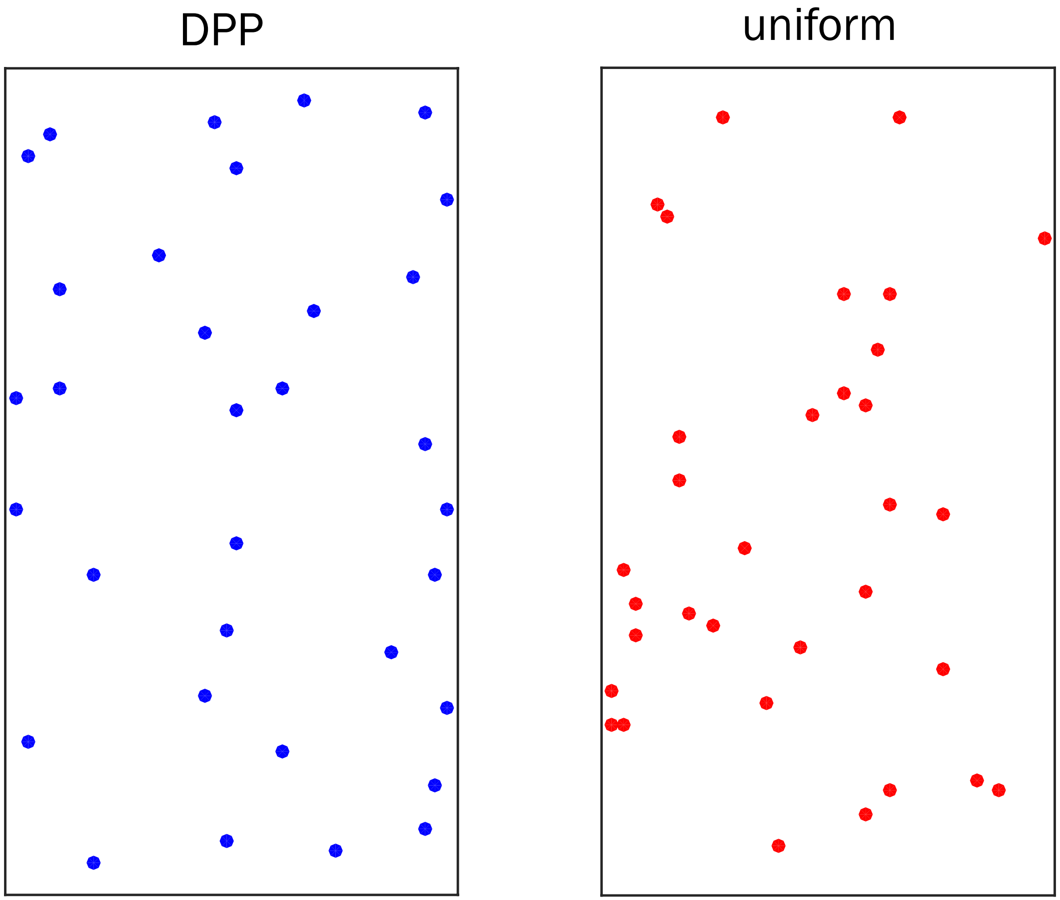


similarities:

$$s_{ij} = \exp(-\frac{1}{2\sigma^2}\|x_i - x_j\|^2)$$

$$\sigma^2 = 35$$

DPP sample – larger bandwidth



$$s_{ij} = \exp(-\frac{1}{2\sigma^2}\|x_i - x_j\|^2)$$

$$\sigma^2 = 135$$

DPPs

- definitions
- computing marginals
- sampling
- computing the mode (MAP)

Determinantal Point Processes

- Macchi 1975: “fermion processes”
- Borodin & Olshanski 2000: “determinantal PP”

2 Definitions:

- marginal kernel *K*: $P(S \subseteq Y) = \det(K_S)$
  - positive semidefinite
  - eigenvalues in [0,1]: $0 \preceq K \preceq 1$

- *L*-ensemble: *(Borodin & Rains, 2005)* $P(Y = T) \propto \det(L_T)$
  - positive semidefinite *L*
  - normalization constant:

$$\sum_{S \subseteq \mathcal{V}} \det(L_S) = \det(L + I_n)$$

2 Definitions

Marginal kernel

$$P(S \subseteq Y) = \det(K_S)$$

- $0 \preceq K \preceq 1$

- *K from L:*

$$K = L(L+I)^{-1}$$

$$K = \sum_{k=1}^{n} \frac{\lambda_k}{1+\lambda_k} v_k v_k^\top$$

L-ensemble

$$P(Y = T) \propto \det(L_T)$$

- $0 \preceq L$

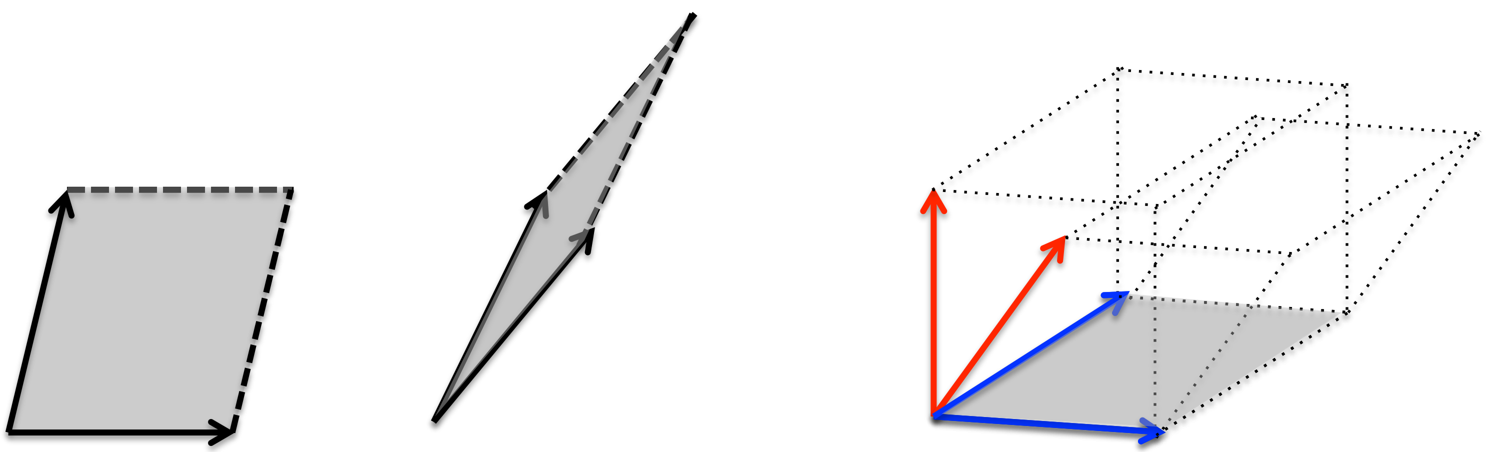
- *L from K:*

$$L = K(I-K)^{-1}$$

$$L = \sum_{k=1}^{n} \lambda_k v_k v_k^\top$$

Geometric view

- data points $x_1, \ldots, x_n$: feature vectors in $\mathbb{R}^d$
- L-ensemble: $L_{ij} = x_i^\top x_j$
- Then $P_L(S) \propto \det(L_S) = \mathrm{Vol}^2(\{x_i\}_{i \in S})$



What happens if dimension d < number of points n?

"Everything" is simple ☺

- normalization

$$\sum_{S \subseteq \mathcal{V}} \det(L_S) = \det(L + I_n)$$

- marginal probabilities: from marginal kernel

$$K = L(L+I)^{-1}$$

- conditioning:

$$P(Y = A \cup B \mid A \subseteq Y) = \frac{\det(L_{A \cup B})}{\det(L + I_{\mathcal{V} \setminus A})}$$

also a DPP *(Borodin & Rains, 2005)*

- …

How many points in the sample?

- L has eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$
- cardinality |*Y*| of sample: Poisson Binomial
  flip n coins, $p_k(\text{head}) = \frac{\lambda_k}{\lambda_k+1}$ -- how many heads?

$$\mathbb{E}[|Y|] = \sum_{k=1}^{n} \frac{\lambda_k}{\lambda_k + 1} = \text{trace}(K)$$

Can we sample efficiently?

Sampling: main idea

- Every DPP is a mixture of "elementary" DPPs

$$P_L(Y) = \sum_T \pi_T P^T(Y) \qquad = \frac{1}{Z} \sum_{T \subseteq \{1,\dots,n\}} \prod_{k \in T} \lambda_k P^T(Y)$$

1. Sample a component P^T with probability π_T

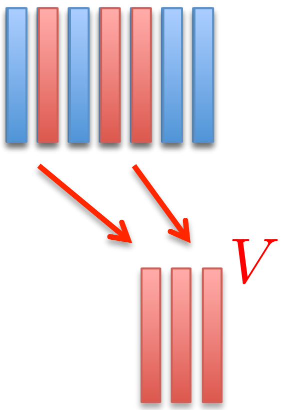
2. Sample Y from P^T

- L has n eigenvectors
- $T \subseteq \{1, \dots, n\}$ indexes a set of eigenvectors
- $$\pi_T = \prod_{k \in T} \frac{\lambda_k}{\det(L+I)}$$

Sampling Y

- compute eigendecomposition $L = \sum_{k=1}^{n} \lambda_k v_k v_k^\top$

1. sample eigenvectors:
   $V = \emptyset$
   add v_k to V with probability $\frac{\lambda_k}{\lambda_k+1}$



2. sample |V| points:

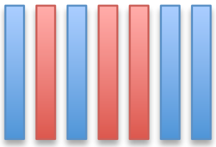
➔ recall: Bernoulli process,

$$\mathbb{E}[|Y|] = \sum_{k=1}^{n} \frac{\lambda_k}{\lambda_k + 1}$$

(Hough et al 2006)

Elementary DPP $P^A(Y)$

- "elementary" DPP: all eigenvalues of *K* are 0 or 1.
- pick a set *A* of eigenvectors v_k of our *L*



$$K^A = \sum_{k \in A} v_k v_k^\top$$

– eigenvalues: $\underbrace{1, 1, \ldots, 1}_{|A| \text{ times}}, \underbrace{0, 0, \ldots, 0}_{n - |A|}$

– sample from this DPP: $|Y| = |A|$ a.s.

– Why?

$$\mathbb{E}\big[|Y|\big] = |A|$$

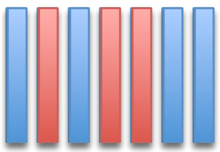
$$\text{for } |Y| > |A| :$$

$$P_K(Y) = \det(K_Y^A) = 0$$

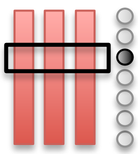
Sampling Y

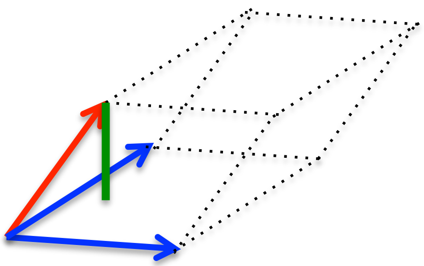
- compute eigendecomposition $$L = \sum_{k=1}^{n} \lambda_k v_k v_k^\top$$

1. sample eigenvectors: $V = \emptyset$
   add v_k to V with probability $\frac{\lambda_k}{\lambda_k + 1}$



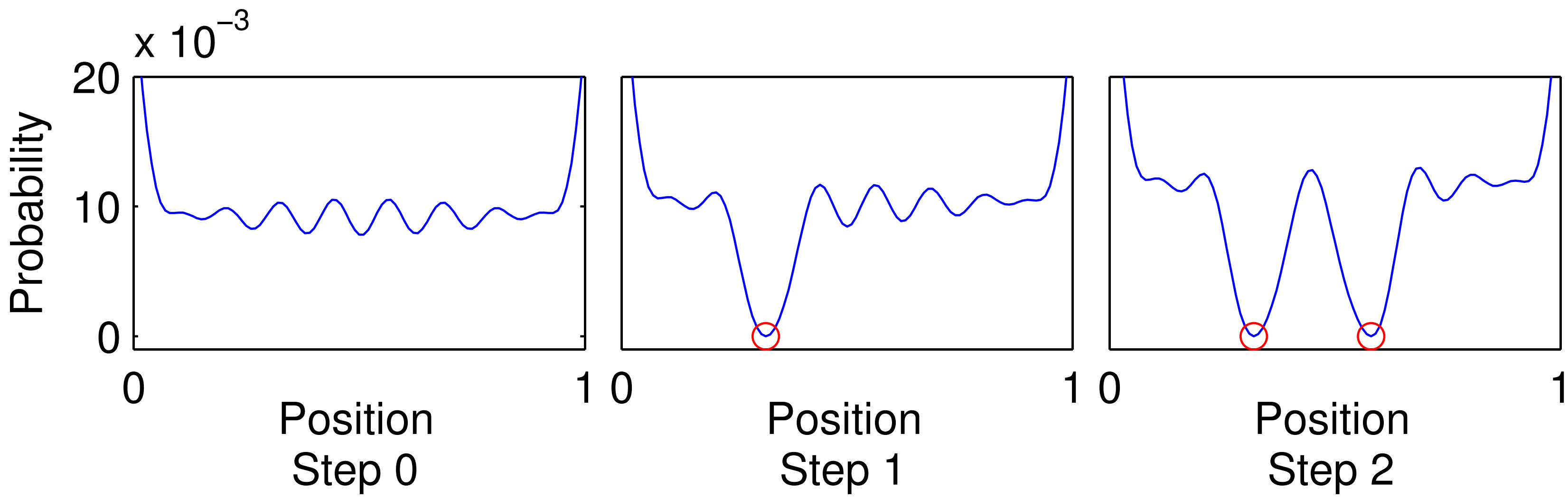
2. sample $|V|$ points: $Y = \emptyset$





(Hough et al 2006)

Sampling



from: (Kulesza & Taskar, FTML)

Finding the mode

$$P(Y = T) \propto \det(L_T)$$

- find $T = \arg\max_{T \subseteq \mathcal{V}} \; P(T)$

$$= \arg\max_{T \subseteq \mathcal{V}} \; \log\det(L_T)$$

submodular

non-monotone

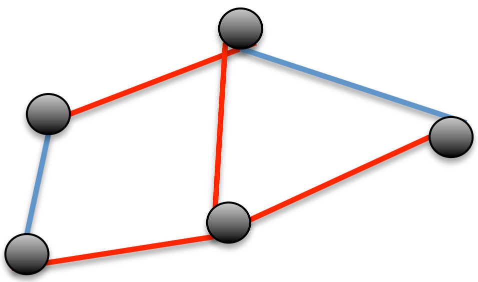
- submodular maximization problem!

The simplest DPP

$$K = \begin{bmatrix} p_1 & 0 & 0 & 0 \\ 0 & p_2 & 0 & 0 \\ 0 & 0 & p_3 & 0 \\ 0 & 0 & 0 & p_4 \end{bmatrix}$$

$$P(Y = S) \;=\; \prod_{i \in S} p_i \prod_{j \notin S} (1 - p_j)$$

Example: random spanning trees



- sample a spanning tree uniformly at random
- probability of a set of edges $S \subseteq \mathcal{E}$ occurring together?

$$\Pr(S \subseteq T)$$

- negative correlation: This is a DPP!

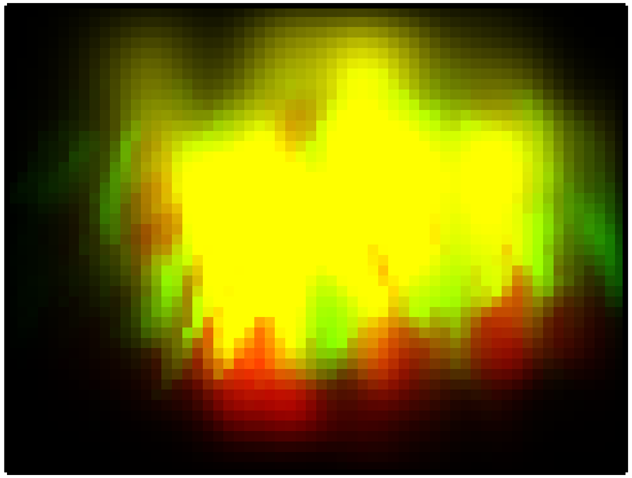
$$\Pr(S \subseteq T) \;\; \leq \;\; \prod_{e \in S} \Pr(e \in T)$$

feature vector for edge $e = (u, v)$

$$b_e = \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow u \\ \\ \leftarrow v \\ \\ \end{matrix}$$

$$x_e = \mathcal{L}^{\dagger/2} b_e$$

Application: pose estimation



Application: pose estimation

$$L_{ij} = x_i^\top x_j \quad = q_i \phi_i^\top \phi_j q_j \qquad\qquad Q_{ij} = \phi_i^\top \phi_j$$

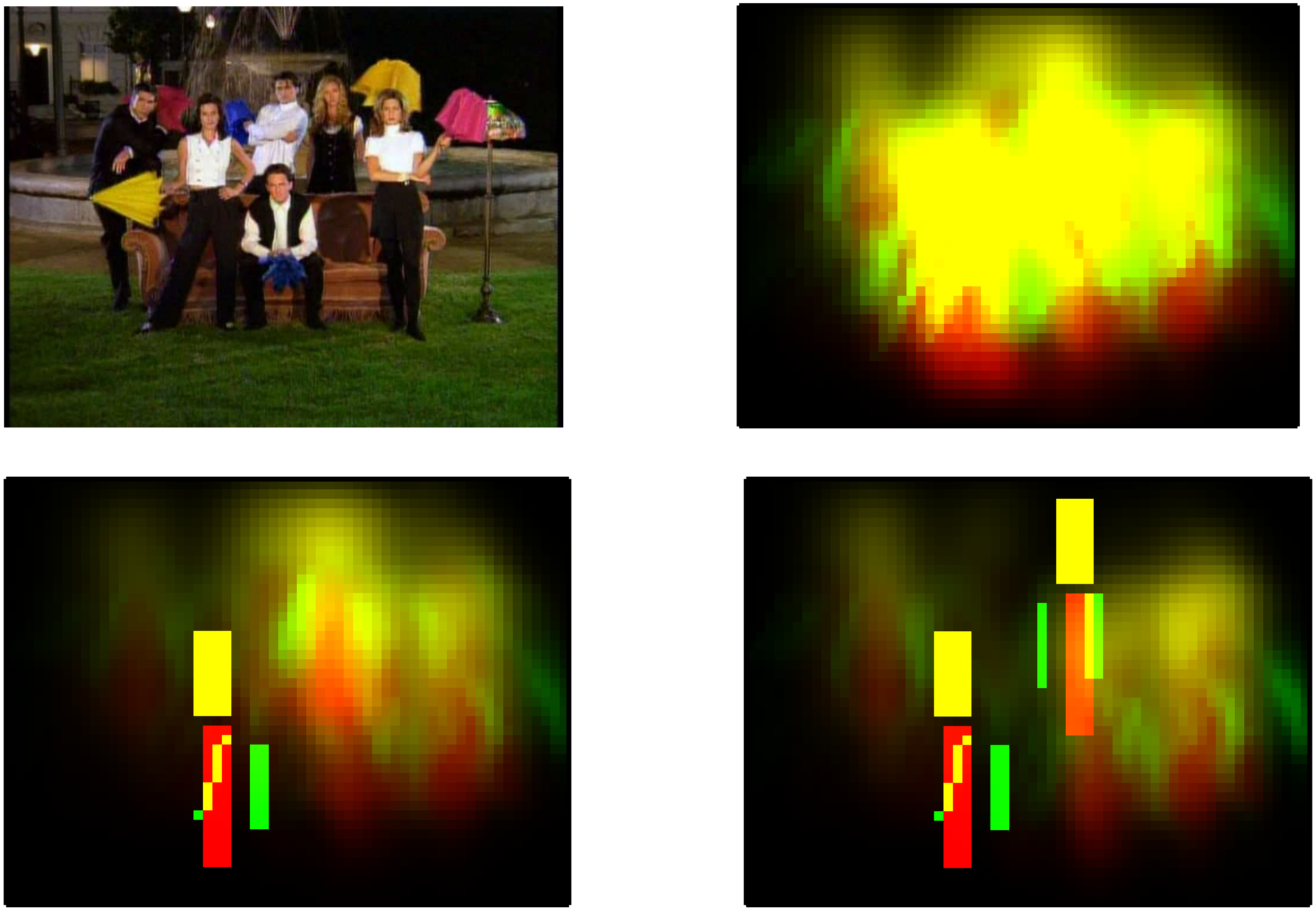
quality score (q_i) — normalized feature vector (ϕ_i)

$$\det(L_S) = \left(\prod_{i \in S} q_i^2 \right) \det(Q_S)$$

- quality model: part detectors for likelihood of body part at location / orientation
- similarity model: location
- data: 73 still frames from TV shows, each 3+ people

(Kulesza&Taskar 2010)

Pose estimation



(Kulesza & Taskar 10)

Summary

- Submodular set functions
  - links to convexity
  - special polyhedra
- Minimizing submodular functions
  - general and special cases: polynomial-time
  - constraints: NP-hard, approximations
- Maximizing submodular functions
  - monotone & non-monotone: NP-hard, constant-factor approximations
  - determinantal point processes

Submodularity and machine learning

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

submodularity
& machine
learning!

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

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