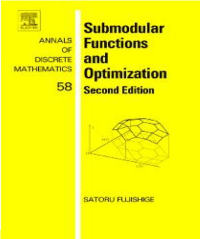
Learning Submodular Functions

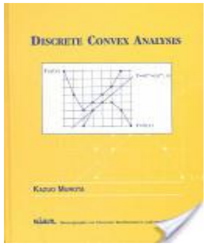
Maria-Florina Balcan
Carnegie Mellon University

2-Minute Version

Submodular fns: important objects (combinatorial fns satisfying diminishing returns) that come up in many areas.

Traditionally: Optimization, operations research





Most recently

- Algorithmic Game Theory [Lehman-Lehman-Nisan'01], ...
- Machine Learning [Bilmes'03] [Guestrin-Krause'07], ...
- Social Networks [Kleinberg-Kempe-Tardos'03]

This talk: learning submodular fns from data.

2-Minute Version

This talk: learning submodular functions from data.

- Can model pbs of interest to many areas, e.g., social networks & alg. game theory.

- General learnability results in a statistical setting; surprising lower bounds showing unexpected structure.

Novel structural results

Alg. Game Theory Economics

Matroid Theory

Discrete Optimization

- Much **better upper bounds** in cases with more structure, coming from social networks & algorithmic game theory.
    - Application for learning influence fnc in diffusion networks.

Structure of the talk

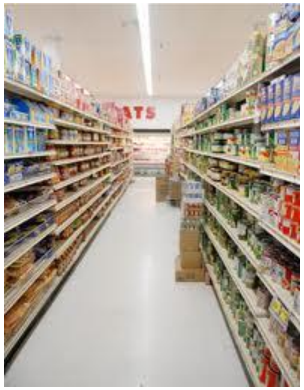
- Submodular functions. Why are they important.

- Learning submodular functions.

  With connections and applications to Algorithmic Game Theory, Economics, Social Networks.

Submodular functions

- First of all, it's a function over sets.
  - e.g., value on some set of items in a store.

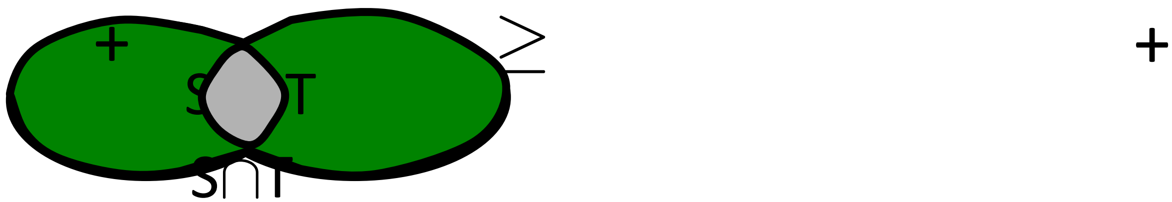


- Ground set V={1,2, …, n}.

Submodular functions

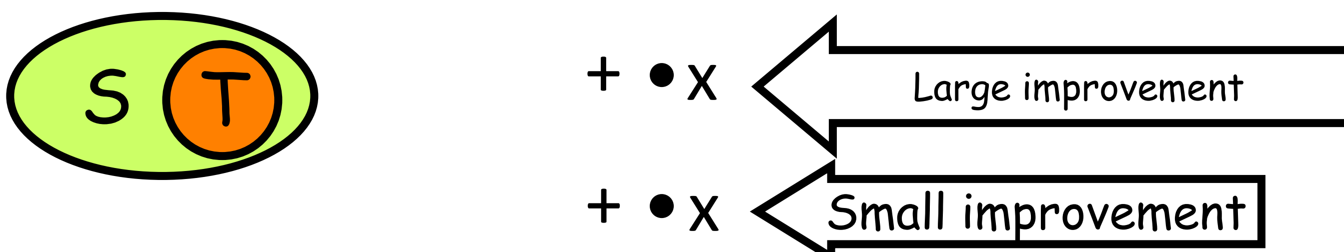
- $V=\{1,2, \ldots, n\}$; set-function $f : 2^V \rightarrow R$ submodular if

  For all $S,T \subseteq V$: $f(S)+f(T) \geq f(S \cap T)+f(S \cup T)$



- Equivalent decreasing marginal return:

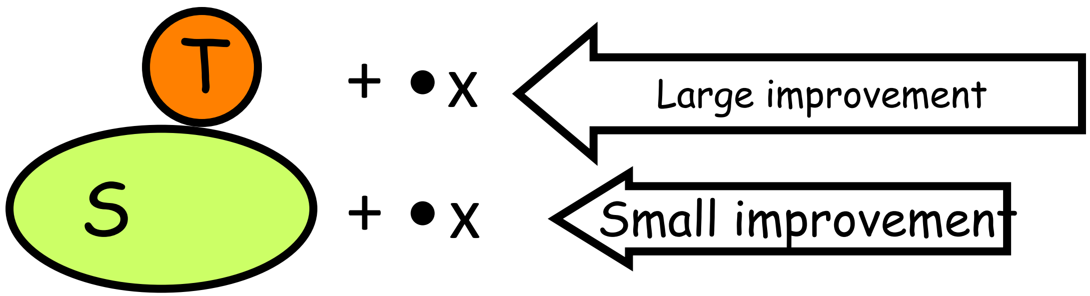
  For $T \subseteq S$, $x \notin S$, $f(T \cup \{x\}) - f(T) \geq f(S \cup \{x\}) - f(S)$



Submodular functions

- $V=\{1,2, \ldots, n\}$; set-function $f : 2^V \rightarrow R$ **submodular** if

For $T \subseteq S$, $x \notin S$, $f(T \cup \{x\}) - f(T) \geq f(S \cup \{x\}) - f(S)$

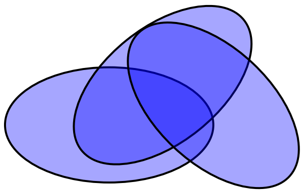


E.g.,

+

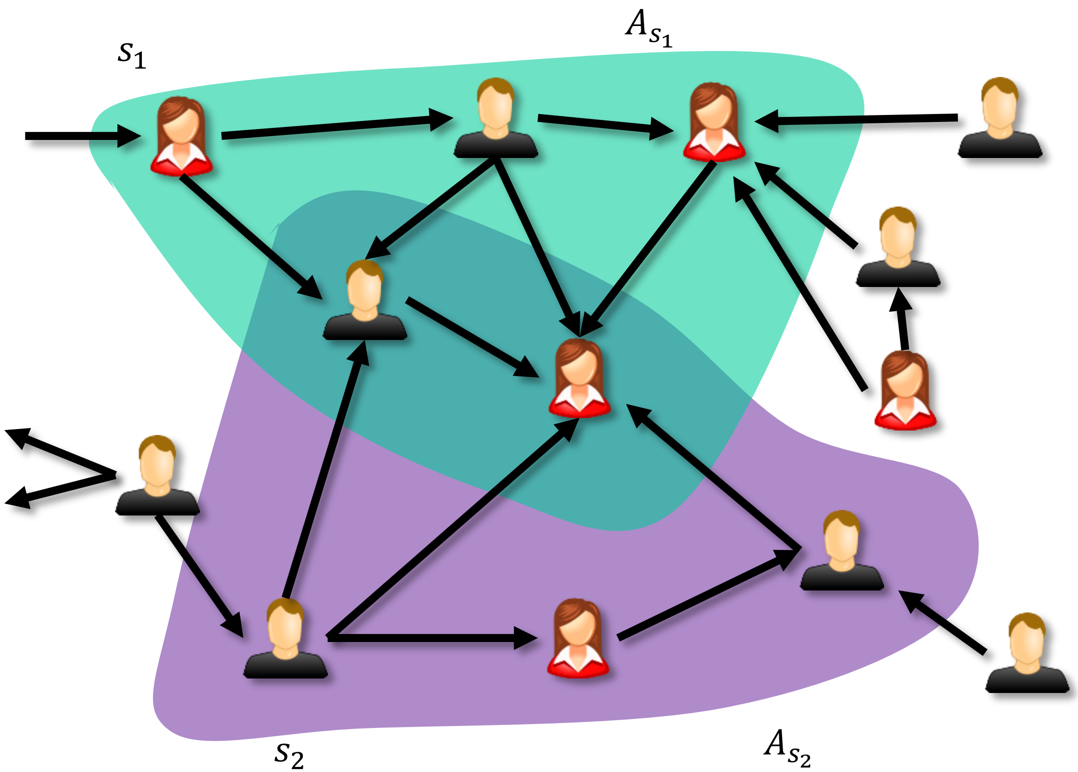
Coverage and Reachability Functions

- **Coverage function:** Let $A_1, \dots, A_n$ be sets.
  For each $S \subseteq V$, let $f(S) = |\cup_{j \in S} A_j|$



- **Reachability function:** $f(S) = \#$ nodes reachable from S.

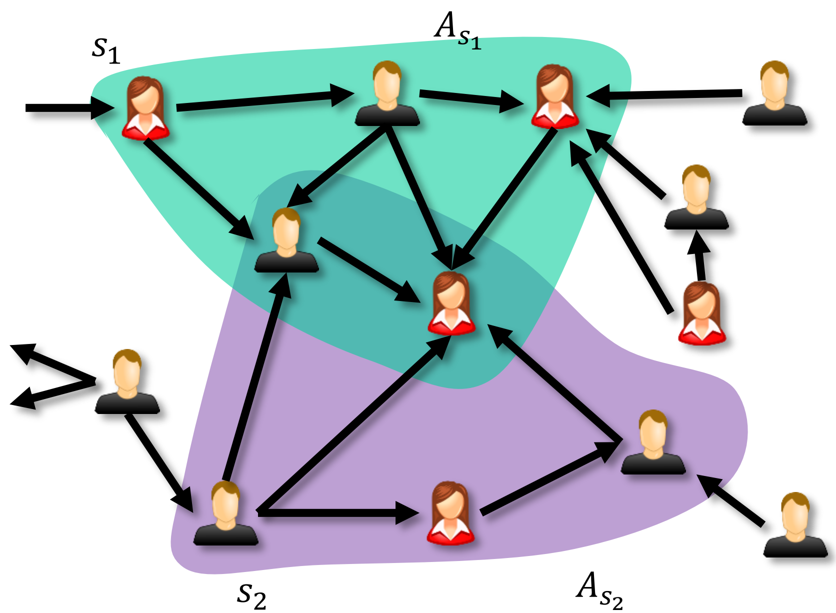
E.g., in a network, A_s nodes reachable from s



Coverage and Reachability Functions

- **Reachability function:** $f(S) = \#$ nodes reachable from S.

  E.g., in a network, A_s nodes reachable from s



Diminishing Returns

- Marginal value of x given S is # number of new nodes that x can reach, but cannot be reached from any of the nodes in S.
- $T \subset S$, $x \notin S$, more chance reach new nodes when adding x to T, than when adding x to S.

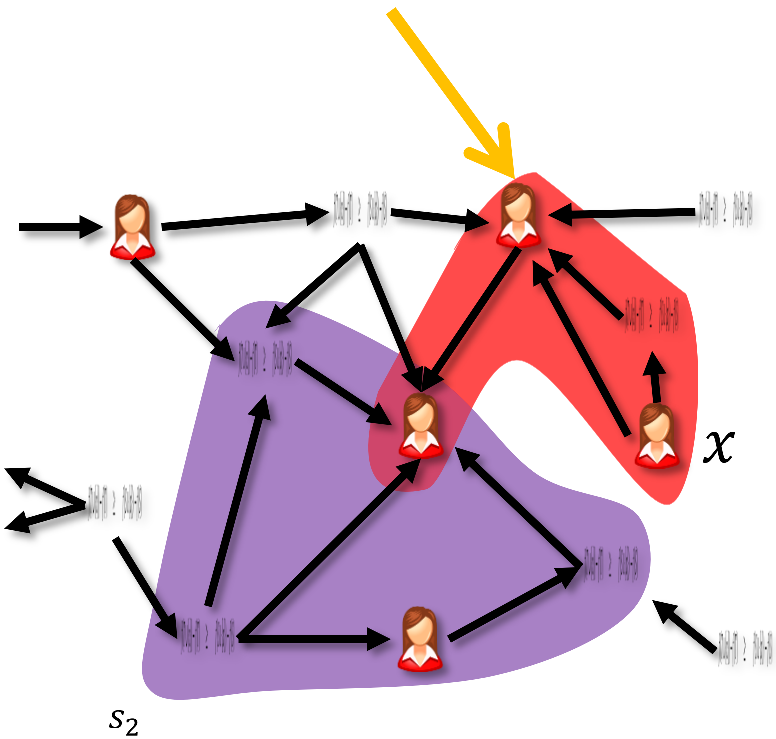
Reachability function is submodular

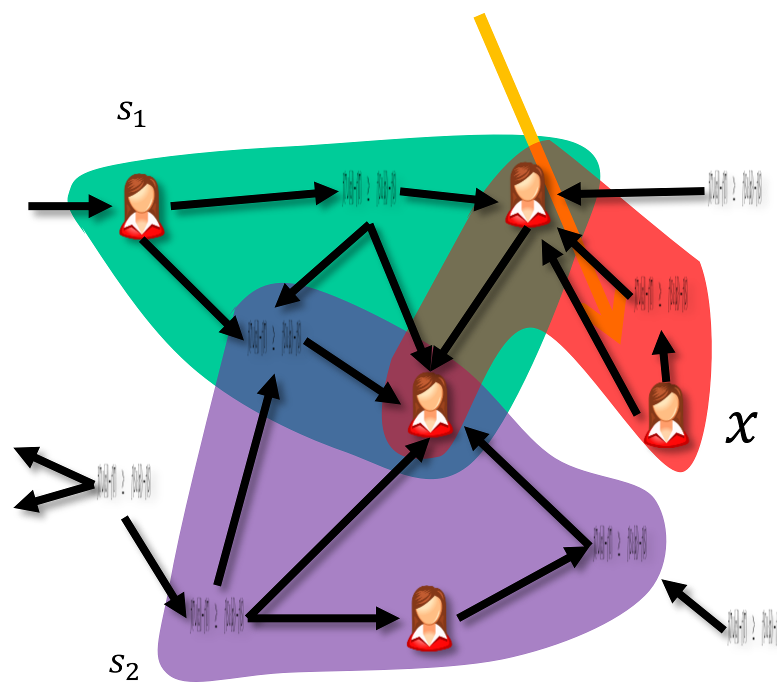
Marginal value of x = # new nodes reachable from x.

$T = \{s_2\}, \quad f(T) = 5$ $\qquad$ $S = \{s_1, s_2\}, \quad f(S) = 8$

$$3 \geq 2$$

$$f(T \cup \{x\}) - f(T) \geq f(S \cup \{x\}) - f(S)$$

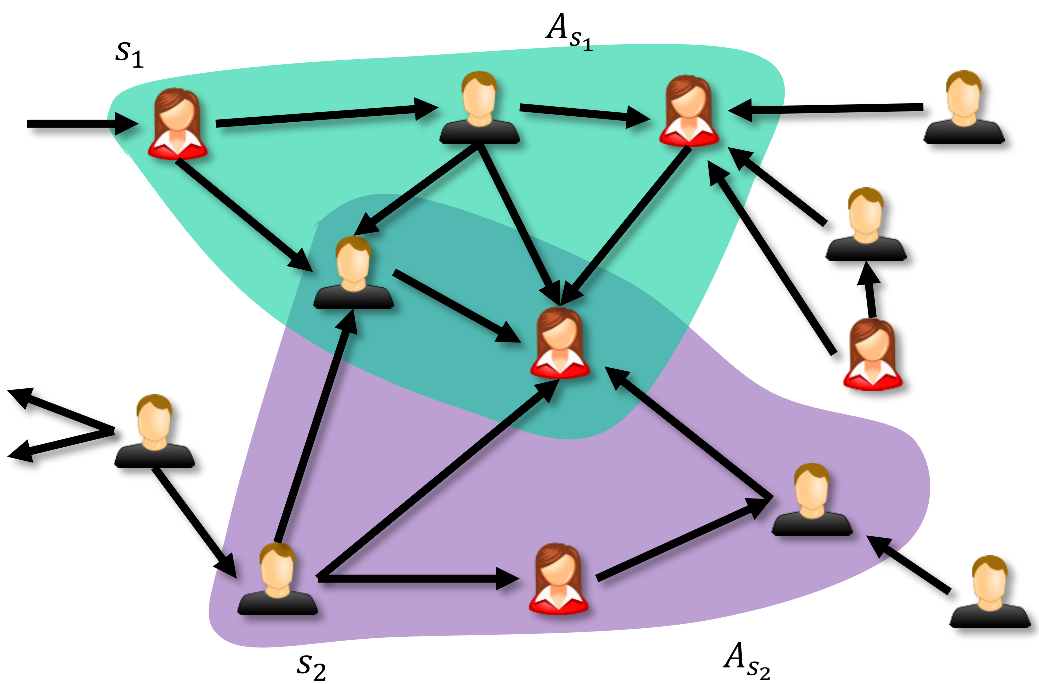




Probabilistic Reachability Functions

- Given a distribution over graphs

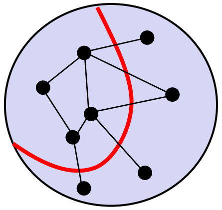
  $f(S) = E_G[\#\text{ reachable from } S|G]$ also submodular.



Submodular functions

More examples:

- **Concave Functions** Let $h : \mathbb{R} \to \mathbb{R}$ be concave. For each $S \subseteq V$, let $f(S) = h(|S|)$
- **Vector Spaces** Let $V=\{v_1,\ldots,v_n\}$, each $v_i \in R^n$. For each $S \subseteq V$, let $f(S) = rank(V[S])$
- **Cut Function in a Graph** Let $f(S)$ = # of edges between S and $V \setminus S$.



This talk: focus on

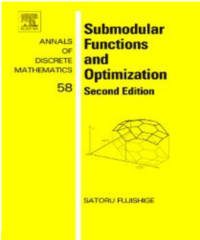
Monotone: $f(S) \leq f(T)\ ,\ \forall\ S \subseteq T$

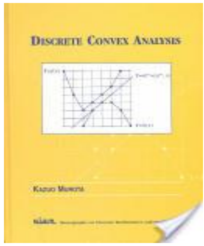
Non-negative: $f(S) \geq 0, \forall\ S \subseteq V$

Submodular functions

- A lot of work on Optimization Problems involving Submodular Functions.

  Traditionally: Optimization, operations research





  Most recently

  - Algorithmic Game Theory [Lehman-Lehman-Nisan'01], ….
  - Machine Learning [Bilmes'03] [Guestrin-Krause'07], …
  - Social Networks [Kleinberg-Kempe-Tardos'03]

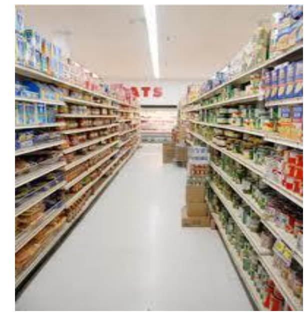
- **This talk**: learning them from data.

Learning submodular functions

Valuation Functions in Economics

Supermarket chain

- V = all the items you sell.
- f(S) = valuation on set of items S.



Learning submodular functions

Influence Function in Social Networks

- V = set of nodes.
- f(S) = expected number of nodes S will influence.

f is a probabilistic reachability fnc in classic diffusion models (e.g., independent cascade model, random threshold model) [Kleinberg-Kempe-Tardos'03]

Past Work

Assume an explicit model on how info spreads ; use it to estimate the influence fnc.

Could be mis-specified.

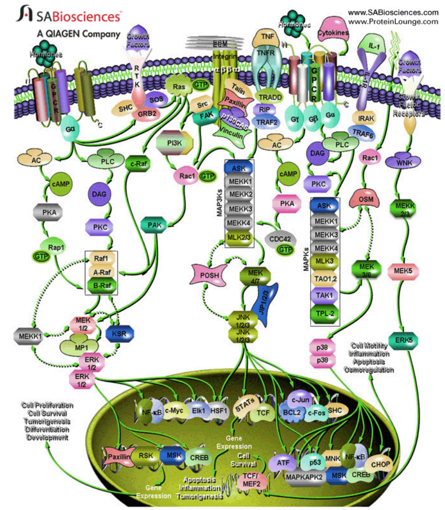
Our Work

Learn the influence function directly from data

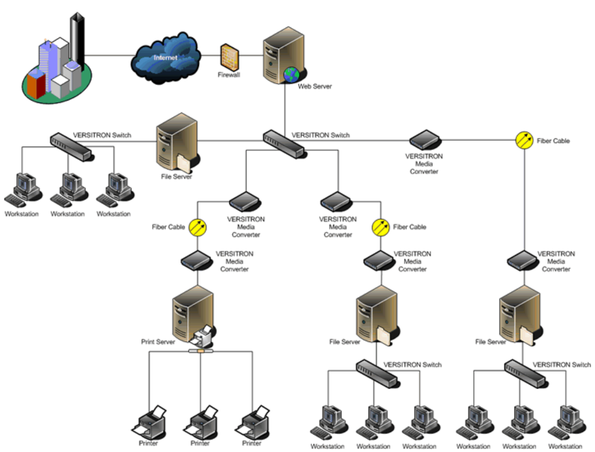
Learning submodular functions

Influence Function in Networks

epidemiology: influenza spread



biology:
gene expression cascade



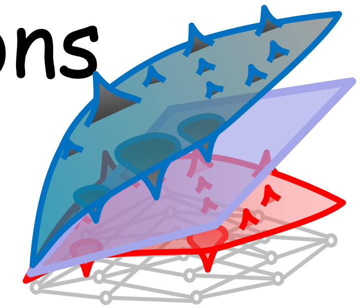
virus report

Symantec

TREND MICRO

cybersecurity:
computer virus spread

Learning Submodular Functions



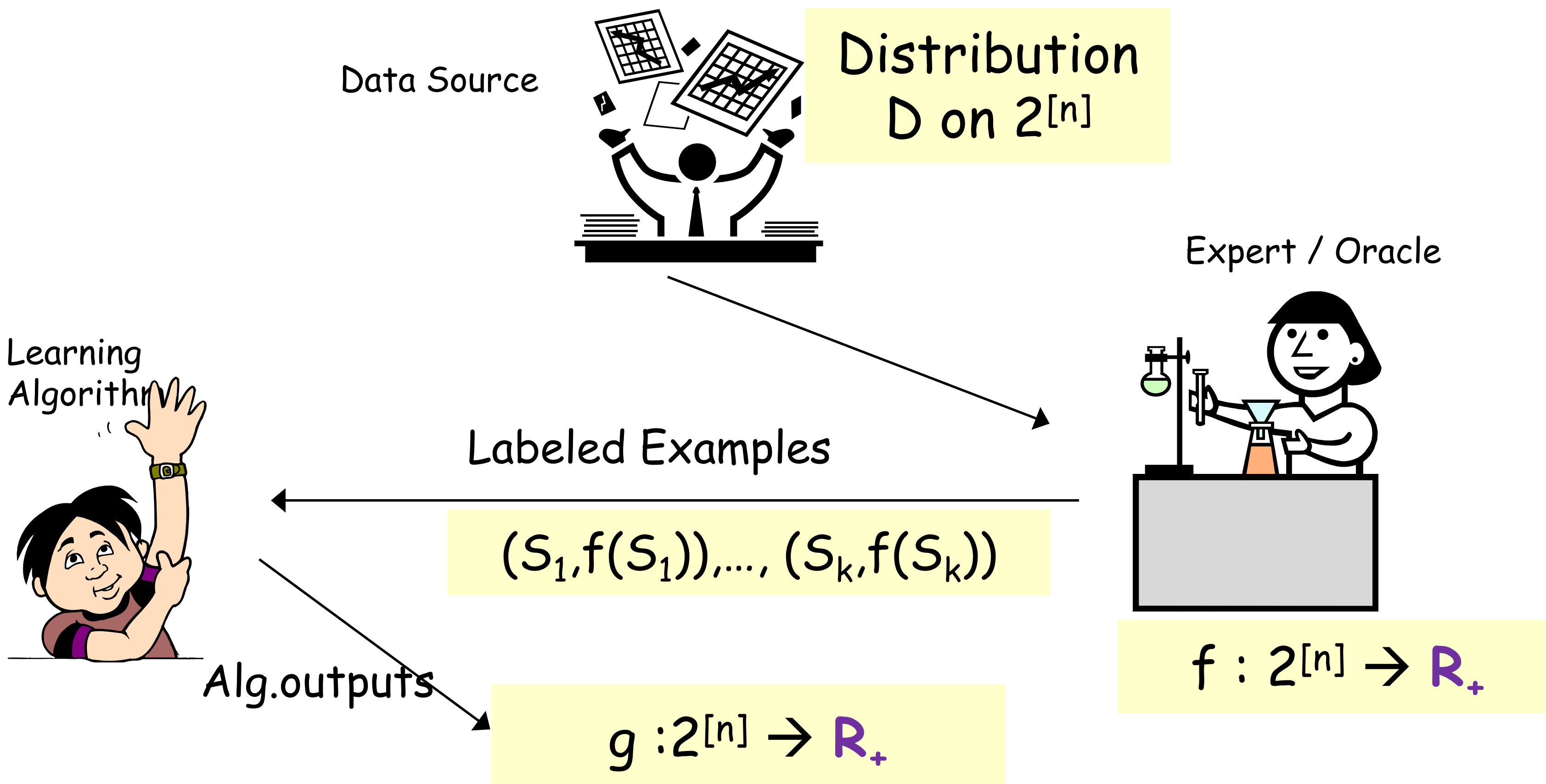
General Learnability Results

- Upper & lower bounds on their intrinsic complexity.
  - Implications to Alg. Game Theory, Economics, Discrete Optimization, Matroid Theory.
- Highlights importance of beyond worst case analysis.

Better Results for Cases with More Structure

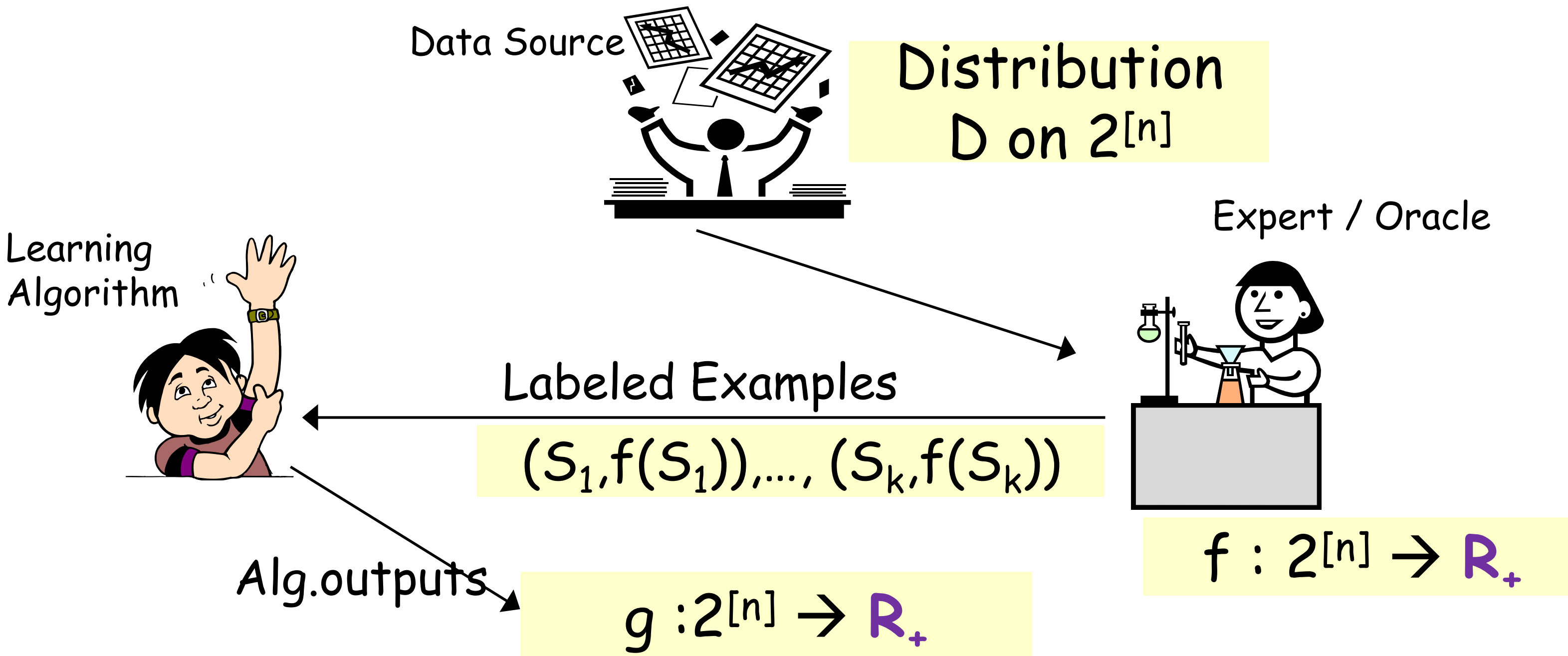
Large Scale Application to Social Networks

Statistical learning model



PMAC model for learning real valued functions

[Balcan&Harvey, STOC 2011 & Nectar Track, ECML-PKDD 2012]

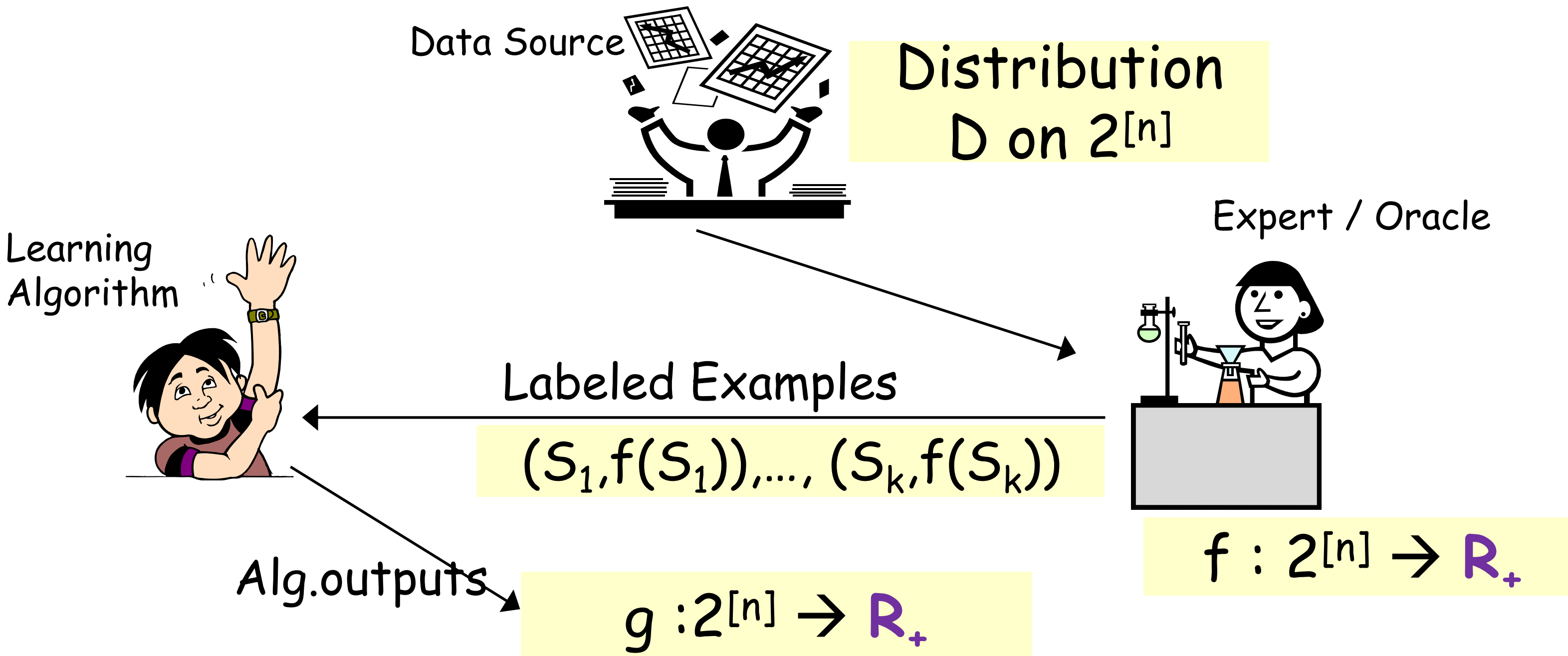


- Algo sees $(S_1,f(S_1)),\ldots,(S_k,f(S_k))$, S_i i.i.d. from D, produces g.
- With probability $\geq 1-\delta$ we have $\Pr_S[g(S) \leq f(S) \leq \alpha\, g(S)] \geq 1-\epsilon$

Probably Mostly Approximately Correct

PMAC model for learning real valued functions

[Balcan&Harvey, STOC 2011 & Nectar Track, ECML-PKDD 2012]



- Algo sees $(S_1,f(S_1)),\ldots,(S_k,f(S_k))$, S_i i.i.d. from D, produces g.
- With probability $\geq 1-\delta$ we have $\Pr_S[g(S) \leq f(S) \leq \alpha\, g(S)] \geq 1-\epsilon$

$\alpha = 1$, recover PAC model.

Learning submodular functions

[Balcan&Harvey, STOC 2011 & Necktar Track, ECML-PKDD 2012]

Theorem: (General upper bound)

Poly time alg. for PMAC-learning (w.r.t. an arbitrary distribution) with an approx. factor $\alpha=O(n^{1/2})$.

Theorem: (General lower bound)

No algo can PMAC learn the class of submodular fns with approx. factor $\tilde{o}(n^{1/3})$.

- Even if value queries allowed; even for rank fns of matroids.

Corollary: Matroid rank fns do **not** have a concise, approximate representation.

Surprising answer to open question in Economics of

Paul Milgrom

Noam Nisan

Moral: Exploit Additional Structure

- Product distribution.
  [Balcan-Harvey,STOC'11] [Feldman-Vondrak,FOCS'13]
- Bounded Curvature (i.e., almost linear)
  [Iyer-Jegelka-Bilmes, NIPS'13]

- Learning valuation fns from AGT and Economics.
  [Balcan-Constantin-Iwata-Wang, COLT '12]
  [Badanidiyuru-Dobzinski-Fu- Kleinberg-Nisan-Roughgarden, SODA'12]
- Learning influence fns in information diffusion networks
  [Du, Liang, Balcan, Song, ICML'14; NIPS'14]

- Learning values of coalitions in cooperative game theory
  [Balcan, Procacia, Zick, IJCAI'15]

Learning submodular functions

[Balcan&Harvey, STOC 2011 & Necktar Track, ECML-PKDD 2012]

Theorem: (General upper bound)

Poly time alg. for PMAC-learning (w.r.t. an arbitrary distribution) with an approx. factor $\alpha = O(n^{1/2})$.

Theorem: (General lower bound)

No algo can PMAC learn the class of submodular fns with approx. factor $\tilde{o}(n^{1/3})$.

- Even if value queries allowed; even for rank fns of matroids.

Surprising answer to open question in Economics of

Paul Milgrom

Noam Nisan

A General Upper Bound

Theorem:

∃ an alg. for PMAC-learning the class of non-negative, monotone, submodular fns (w.r.t. an arbitrary distribution) with an approx. factor $O(n^{1/2})$.

Subadditive Fns are Approximately Linear

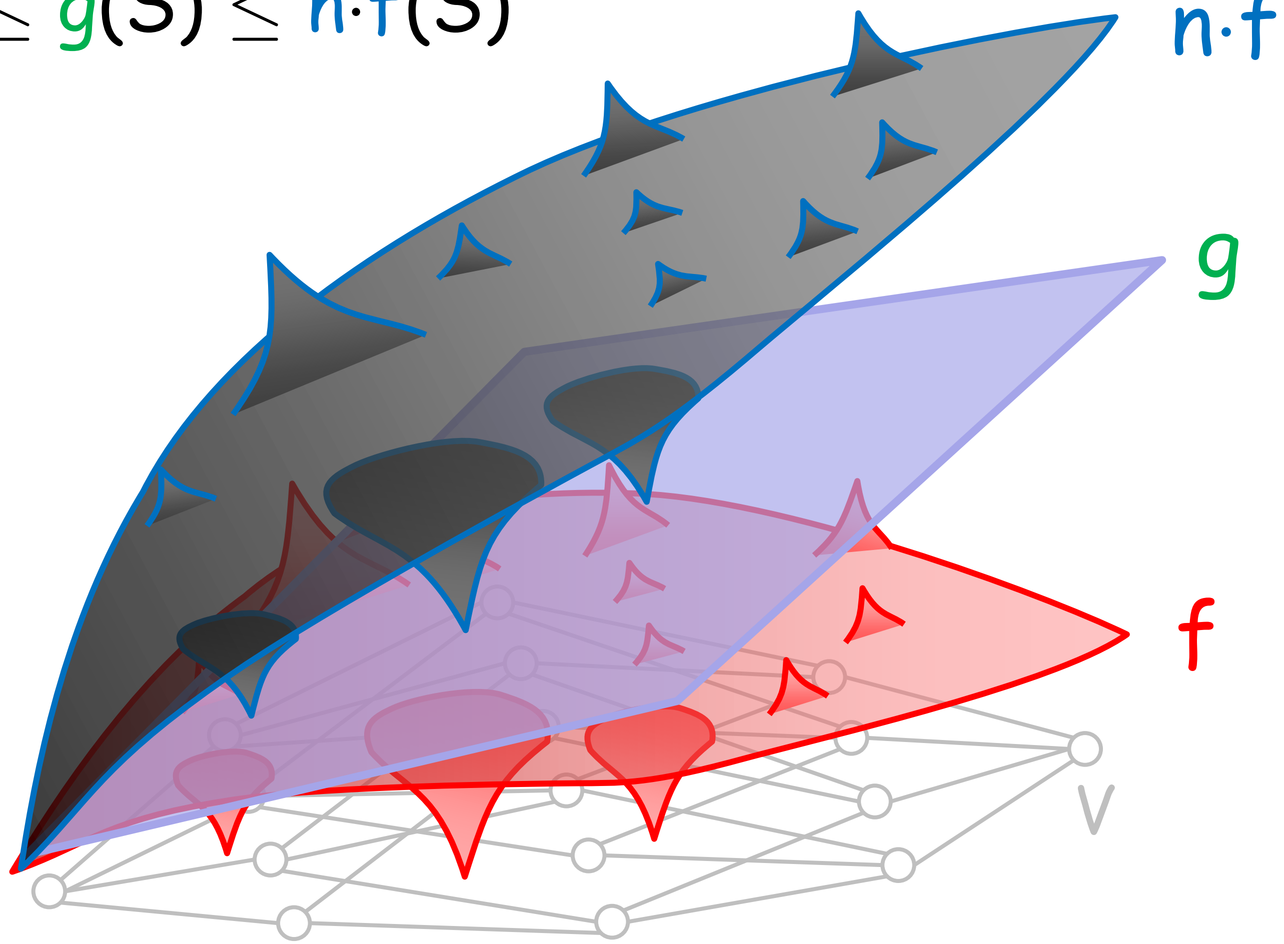
- Let f be non-negative, monotone and subadditive
- **Claim:** f can be approximated to within factor n by a **linear** function g.

- **Proof Sketch:** Let $g(S) = \sum_{x \text{ in } S} f(\{x\})$.
  Then $f(S) \leq g(S) \leq n \cdot f(S)$.

Subadditive:	$f(S)+f(T) \geq f(S \cup T)$	$\forall\, S,T \subseteq V$
Monotonicity:	$f(S) \leq f(T)$	$\forall\, S \subseteq T$
Non-negativity:	$f(S) \geq 0$	$\forall\, S \subseteq V$

Subaddtive Fns are Approximately Linear

$f(S) \leq g(S) \leq n \cdot f(S)$



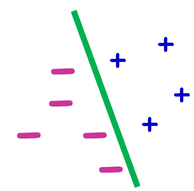
PMAC Learning Subadditive Valuations

$f(S) \leq g(S) \leq n \cdot f(S)$ where $g(S) = w \cdot \chi(S)$

features

- Labeled examples $((\chi(S), f(S)), +)$ and $((\chi(S), n \cdot f(S)), -)$ are linearly separable in R^{n+1}.

- **Idea: reduction to learning a linear separator.**



  Problem: data not i.i.d.

  Solution: create a related distib. P. Sample S from D; flip a coin. If heads add $((\chi(S), f(S)), +)$. Else add $((\chi(S), n \cdot f(S)), -)$.

  - Claim: A linear separator with low error on P induces a linear function with an approx. factor of n on the original data.

PMAC Learning Subadditive Valuations

Algorithm:

Input: $(S_1, f(S_1)) \ldots, (S_m, f(S_m))$

- For each S_i, flip a coin.
  - If heads add $((\chi(S), f(S_i)), +)$.
  - Else add $((\chi(S), n \cdot f(S_i)), -)$.
- Learn a linear separator $u=(w,-z)$ in R^{n+1}.

Output: $g(S)=1/(n+1)\ w \cdot \chi(S)$.

- **Theorem:** For $m = \Theta(n/\epsilon)$, g approximates f to within a factor n on a $1-\epsilon$ fraction of the distribution.

PMAC Learning Submodular Fns

Algorithm:

Input: $(S_1, f(S_1)) \ldots, (S_m, f(S_m))$

- For each S_i, flip a coin.
  - If heads add $((\chi(S), f^2(S_i)), +)$.
  - Else add $((\chi(S), n\, f^2(S_i)), -)$.
- Learn a linear separator $u=(w,-z)$ in R^{n+1}.

Output: $g(S)=1/(n+1)^{1/2}\ w \cdot \chi(S)$

- **Theorem:** For $m = \Theta(n/\epsilon)$, g approximates f to within a factor $n^{1/2}$ on a $1-\epsilon$ fraction of the distribution.

Proof idea: f non-negative, monotone, submodular can be approximated within $n^{1/2}$ by a \sqrt{linear function}. [GHIM, 09]

PMAC Learning Submodular Fns

Algorithm:

Input: $(S_1, f(S_1)) \ldots, (S_m, f(S_m))$

- For each S_i, flip a coin.
  - If heads add $((\chi(S), f^2(S_i)), +)$.
  - Else add $((\chi(S), n\, f^2(S_i)), -)$.
- Learn a linear separator $u=(w,-z)$ in R^{n+1}.

Output: $g(S)=1/(n+1)^{1/2}\ w \cdot \chi(S)$

- **Theorem:** For $m = \Theta(n/\epsilon)$, g approximates f to within a factor $n^{1/2}$ on a $1-\epsilon$ fraction of the distribution.

Proof idea: f non-negative, monotone, submodular can be approximated within $n^{1/2}$ by a \sqrt{linear function}. [GHIM, 09]

A General Lower Bound

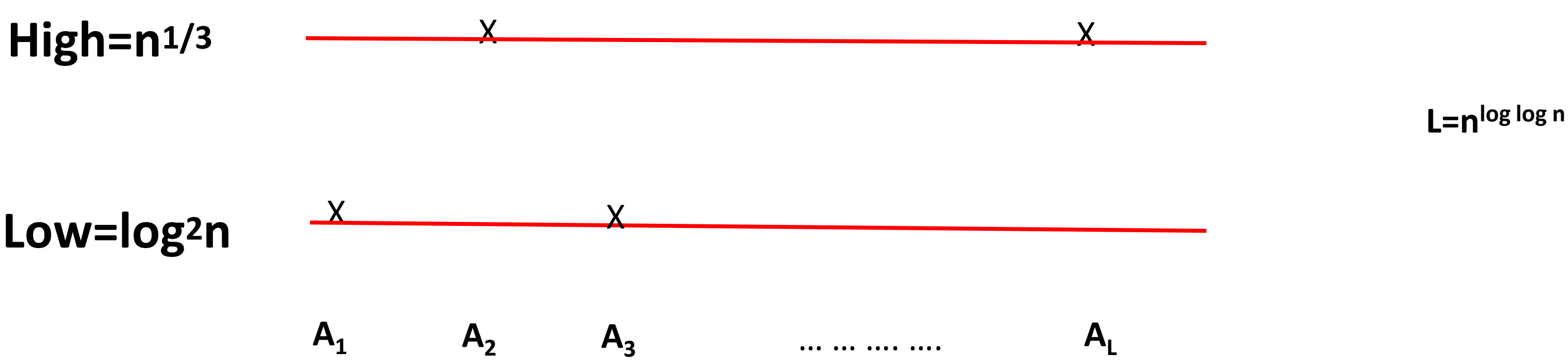
Theorem

No algorithm can PMAC learn the class of non-neg., monotone, submodular fns with an approx. factor $\tilde{o}(n^{1/3})$.

Plan:

Use the fact that any matroid rank fnc is submodular.

Construct a hard family of matroid rank functions.



Vast generalization of partition matroids.

Partition Matroids

$A_1, A_2, \ldots, A_k \subseteq V=\{1,2, \ldots, n\}$, all disjoint; $u_i \leq |A_i|-1$

$Ind=\{I: |I \cap A_j| \leq u_j$, for all $j\}$

Then (V, Ind) is a matroid.

If sets A_i are not disjoint, then (V,Ind) might not be a matroid.

- E.g., n=5, $A_1=\{1,2,3\}$, $A_2=\{3,4,5\}$, $u_1 = u_2=2$.
- {1,2,4,5} and {2,3,4} both maximal sets in Ind; do not have the same cardinality.

Almost partition matroids

$k=2$, $A_1, A_2 \subseteq V$ (not necessarily disjoint); $u_i \leq |A_i|-1$

$Ind=\{I: |I \cap A_j| \leq u_j , |I \cap (A_1 \cup A_2)| \leq u_1 + u_2 - |A_1 \cap A_2|\}$

Then (V, Ind) is a matroid.

Almost partition matroids

More generally

$A_1, A_2, \ldots, A_k \subseteq V=\{1,2, \ldots, n\}$, $u_i \leq |A_i|-1$; $f : 2^{[k]} \to Z$

=<0

$$f(J)= \sum_{j \in J} u_j + |A(J)| - \sum_{j \in J} |A_j|, \forall J \subseteq [k]$$

$$\text{Ind} = \{ I: |I \cap A(J)| \leq f(J), \forall J \subseteq [k] \}$$

Then (V, Ind) is a matroid (if nonempty).

Rewrite f, $f(J)=|A(J)|-\sum_{j \in J}(|A_j| - u_j), \forall J \subseteq [k]$

Almost partition matroids

More generally $f : 2^{[k]} \rightarrow Z$

$$f(J)=|A(J)|-\sum_{j \in J}(|A_j| - u_j), \forall J \subseteq [k]$$

$$Ind= \{ I: |I \cap A(J)| \leq f(J), \forall J \subseteq [k] \}$$

Then (V, Ind) is a matroid (if nonempty).

$f : 2^{[k]} \rightarrow Z, f(J)=|A(J)|-\sum_{j \in J}(|A_j| -u_j), \forall J \subseteq [k]; u_i \leq |A_i|-1$

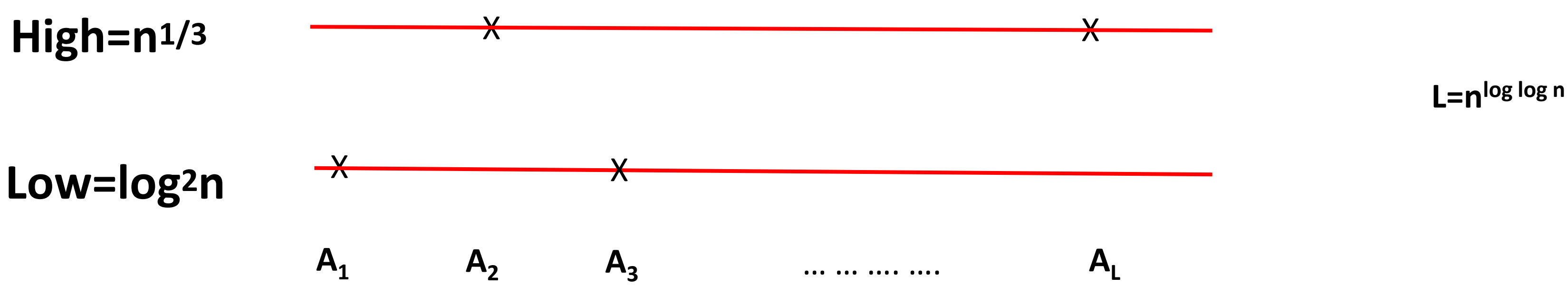
Note: This requires $k \leq n$ (for k > n, f becomes negative)

More tricks to allow $k=n^{\log \log n}$.

Learning submodular valuations

Theorem

No algorithm can PMAC learn the class of non-neg., monotone, submodular fns with an approx. factor $\tilde{o}(n^{1/3})$.



Worst Case Analysis ☺

Moral: Exploit Additional Structure

- Product distribution.
  [Balcan-Harvey,STOC'11] [Feldman-Vondrak,FOCS'13]
- Bounded Curvature (i.e., almost linear)
  [Iyer-Jegelka-Bilmes, NIPS'13]

- Learning valuation fns from AGT and Economics.
  [Balcan-Constantin-Iwata-Wang, COLT '12]
  [Badanidiyuru-Dobzinski-Fu- Kleinberg-Nisan-Roughgarden, SODA'12]
- Learning influence fns in information diffusion networks
  [Du, Liang, Balcan, Song, ICML'14; NIPS'14]

- Learning values of coalitions in cooperative game theory
  [Balcan, Procacia, Zick, IJCAI'15]

Learning Valuation Functions

- Target dependent learnability for classes of valuation fns have a succinct description.

[Balcan-Constantin-Iwata-Wang, COLT 2012]

Well-studied subclasses of subadditive valuations.

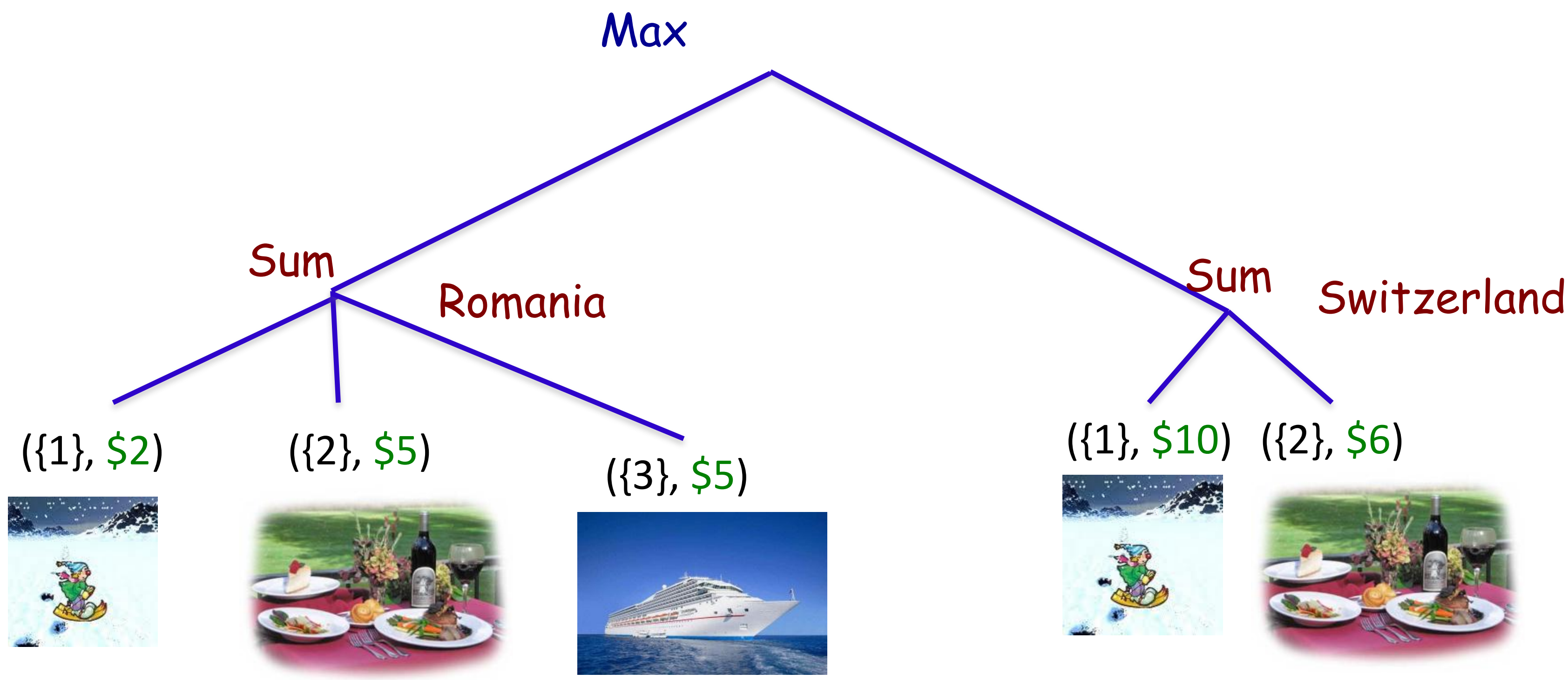
[Algorithmic game theory and Economics]

Additive $\subseteq$ OXS $\subseteq$ Submodular $\subseteq$ XOS $\subseteq$ Subadditive

[Sandholm'99] [Lehman-Lehman-Nisan'01]

XOS valuations

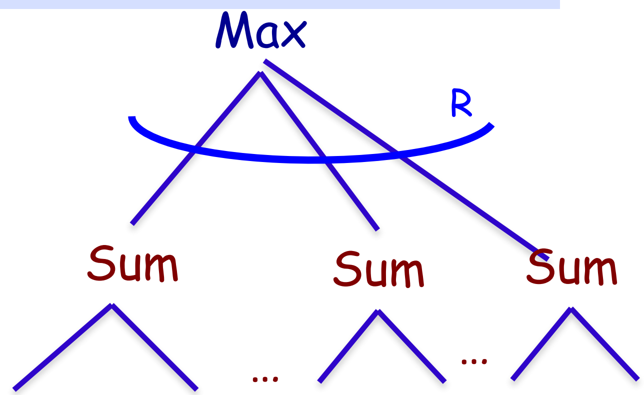
Functions that can be represented as a MAX of SUMs.



g({1,2}) = $16   g({2,3}) = $10   g({1,2,3}) = $16

Target dependent Upper Bound for XOS

Theorem: (Polynomial number of Sum trees)
$O(R^{\epsilon})$ approximation in time $O(n^{1/\epsilon})$.



Main Idea:

- Target approx within $O(R^{\epsilon})$ by a linear function over $O(n^{1/\epsilon})$ feature space (one feature for each $n^{1/\epsilon}$-tuple of items).

- Reduction to learning a linear separator in a higher dim. feature space.

Highlights importance of considering the complexity of the target function.

Learning Influence Functions in Information Diffusion Networks

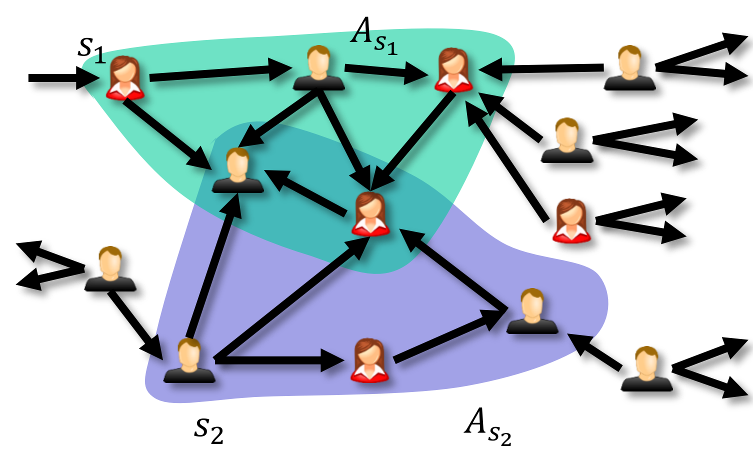
[Du, Liang, Balcan, Song, ICML 2014 , NIPS'14]

Influence Function in Networks

- V = set of nodes.
- f(S) = expected number of nodes S will influence.

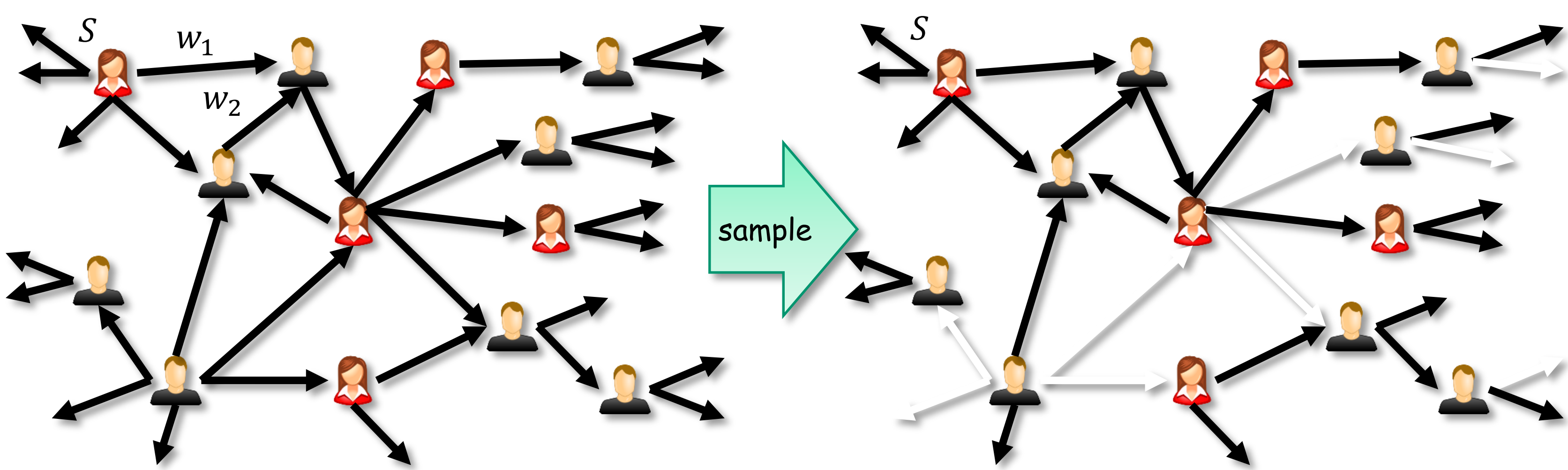
Fact: in classic diffusion models (discrete time independent cascade model/random threshold model, continuous time analogues, etc), the influence function is coverage function. [Kleinberg-Kempe-Tardos'03]

$f(S) = E_G[\#\ \text{reachable from } S|G]$
probabilistic reachability fnc



Discrete-time independent cascade model

- Each edge has a weight $w \in [0,1]$
- Cascade generative process for a source set S
  - presence of edge is sampled independently according to w
  - influenced nodes are those reachable from at least one of the source nodes in the resulting "live edge graph"
- Influence of S is expected number of nodes influenced under this random process

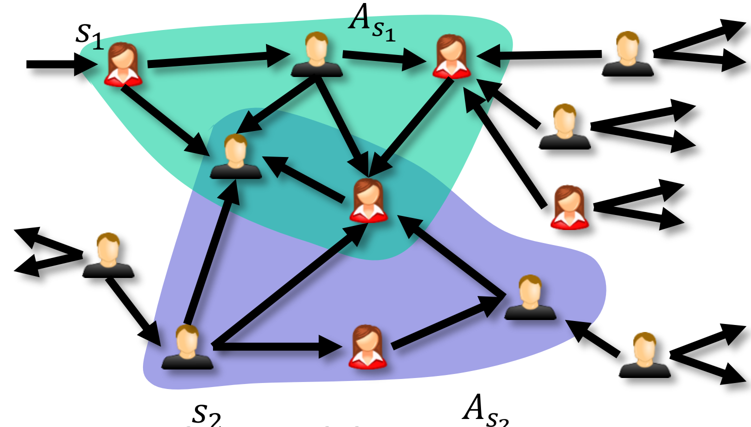


Learning Influence Functions in Information Diffusion Networks

[Du, Liang, Balcan, Song, ICML 2014 , NIPS'14]

Fact: in classic diffusion models, the influence function is a coverage function.

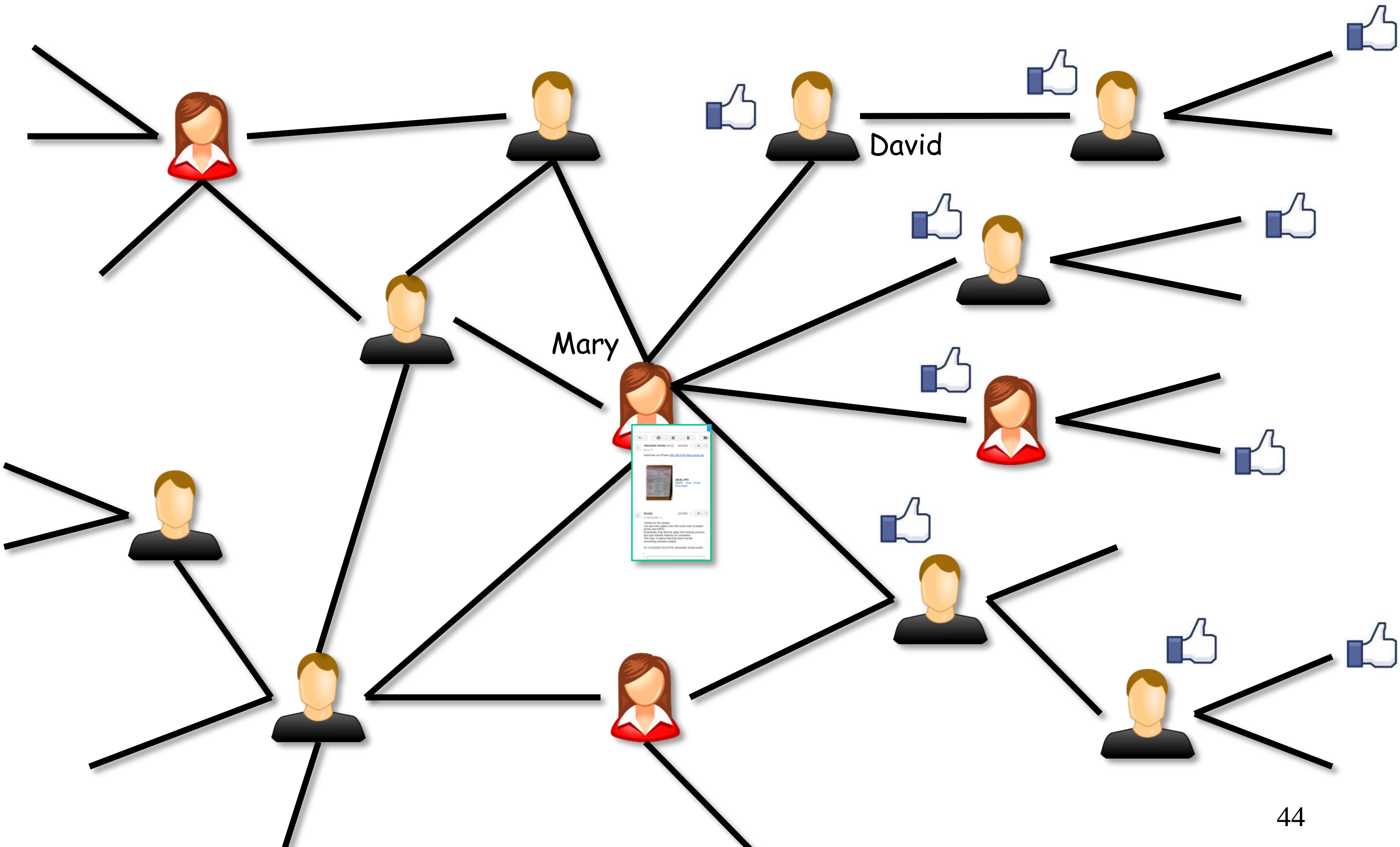
$f(S) = E_G[\# \text{ reachable from } S|G]$
probabilistic reachability fnc



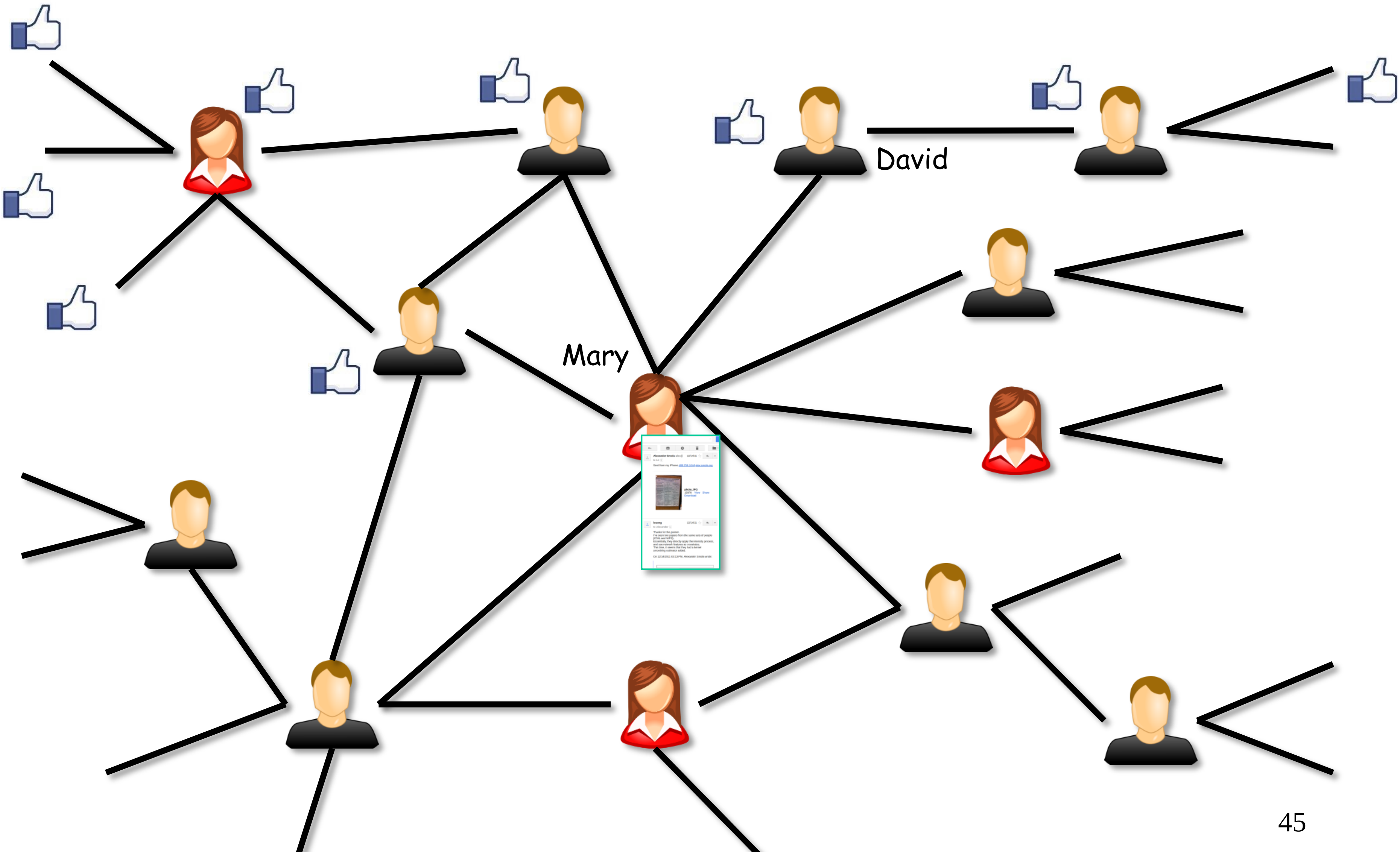
- **Note 1**: Do not know better guarantees for efficient algorithms if access only to value queries.

- **Note 2**: Do better theoretically and empirically, if have access to information diffusion traces or cascades.

Learning Influence Functions based on information propagation traces (cascades)



Another cascade



Learning the influence function

Input: past influence cascades $\{(S_1, I_1), (S_2, I_2), \ldots, (S_m, I_m)\}$.

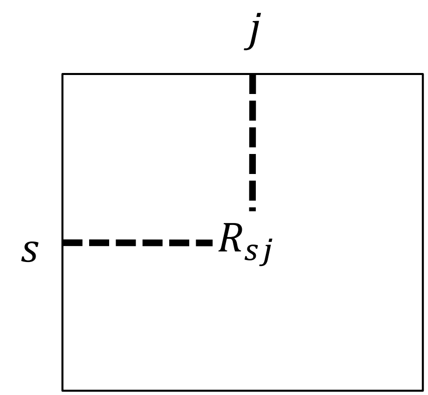
Goal: learn Influence function $f(S) = E[\#\text{influenced}(S)]$.

Assumption: $f(S)$ is a probabilistic coverage function.

I.e., there is a distribution p_R over reachability matrices R s.t.:

$$f(S) = E_{R \sim p_R}[\#\text{influenced}(S|R)]$$

$$|\{j: R_{sj} = 1 \text{ for some } s \in S\}|$$



$R_{sj} = 1$ if s can reach j,
$R_{sj} = 0$ otherwise.

Learning the influence function

Input: past influence cascades $\{(S_1, I_1), (S_2, I_2), \ldots, (S_m, I_m)\}$.

Goal: learn Influence function $f(S) = E[\#\text{influenced}(S)]$.

Idea: $f(S) = \sum_j f_j(S)$, where $f_j(S) = \Pr_{R \sim p_R}(j \text{ is influenced by } S)$.

For each j, will learn $\hat{f}_j(S)$. Output $\sum_j \hat{f}_j(S)$.

Algorithm for learning f_j

Use "random kitchen sink" approach:

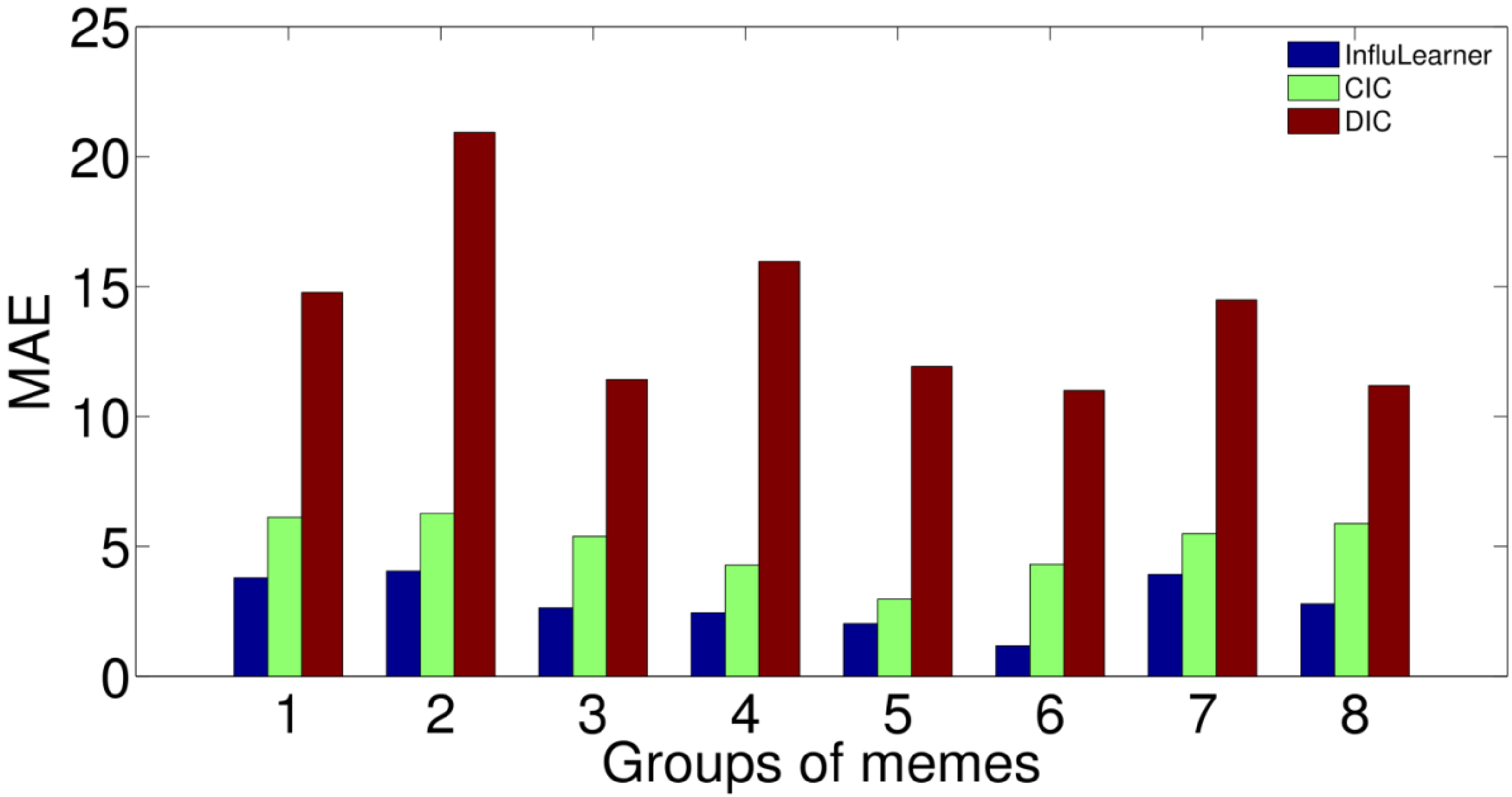
- choose random binary vectors $v_1, v_2, \ldots, v_K$ from q.
- Parametrize $\hat{f}_j(S)$ as $\sum_i w_i \cdot I[\langle I_S, v_i \rangle \geq 1]$ $(\sum_i w_i \leq 1, w_i \geq 0)$

Learn weights via maximum conditional likelihood.

Influence estimation in real data

[Du, Liang, Balcan, Song, ICML 2014 , NIPS'14]

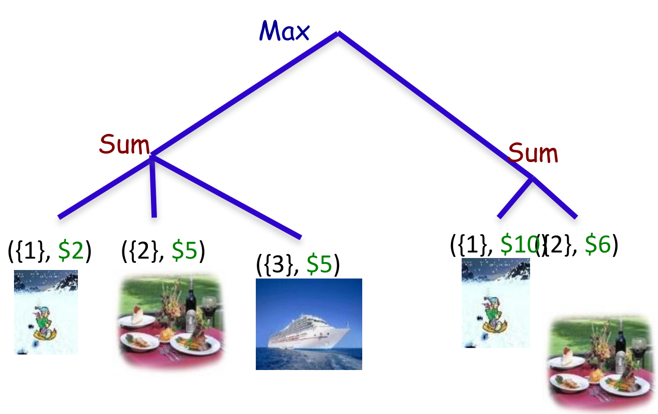
- Memetracker Dataset, blog data cascades : "apple and jobs", "tsunami earthquake", "william kate marriage"

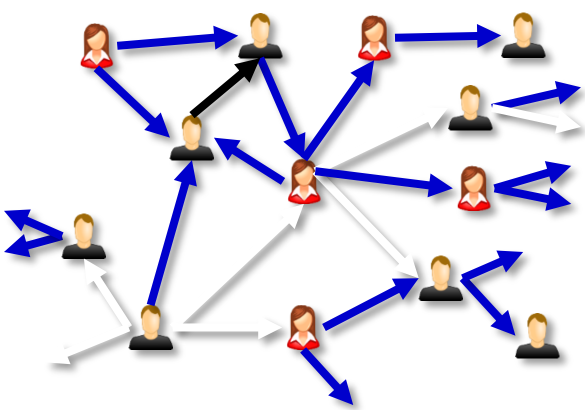


Conclusions

Learnability of submodular, other combinatorial fns

- Can model problems of interest to many areas.
- Very strong lower bounds in the worst case.
- Much better learnability results for classes with additional structure.





Conclusions

Learnability of submodular functions

- Very strong lower bounds in the worst case.
- Highlight the importance of considering the complexity of the target function.

Open Questions:

- Exploit complexity of target for better approx guarantees. [for learning and optimization]
  - What is a natural description language for submodular fns?
- Other interesting applications.