Privately Releasing Conjunctions and the Statistical-Queries Barrier

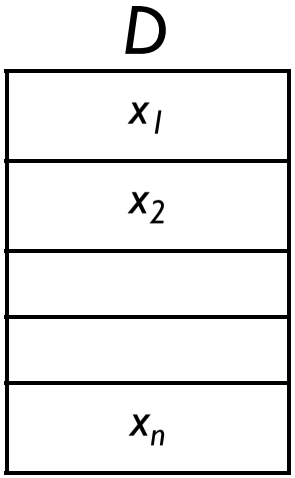
Anupam Gupta (CMU)
Moritz Hardt (Princeton / IBM Almaden)
Aaron Roth (MSR New England / U. Penn)
Jonathan Ullman (Harvard)

Talk Overview

- Introduce differentially private data release
- A learning theoretic approach
- Learning a submodular function

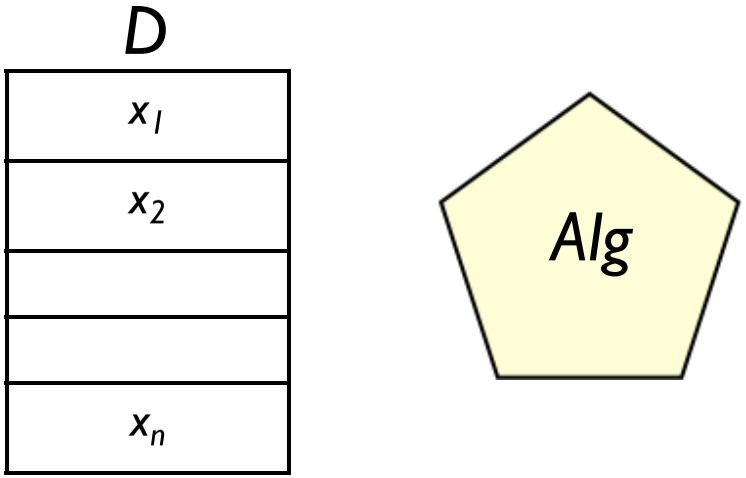
Differential Privacy

[DN03,BDMN05,DMNS06...]



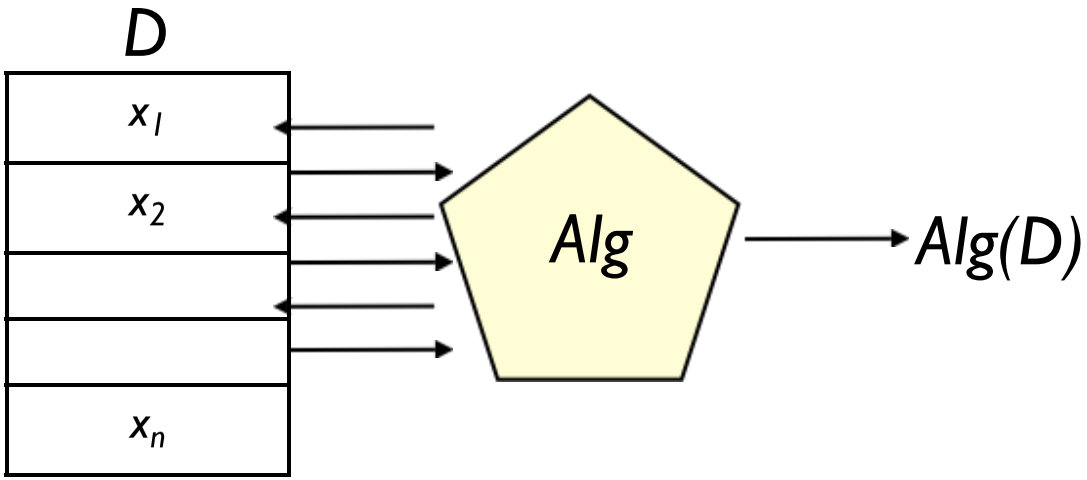
Differential Privacy

[DN03,BDMN05,DMNS06...]



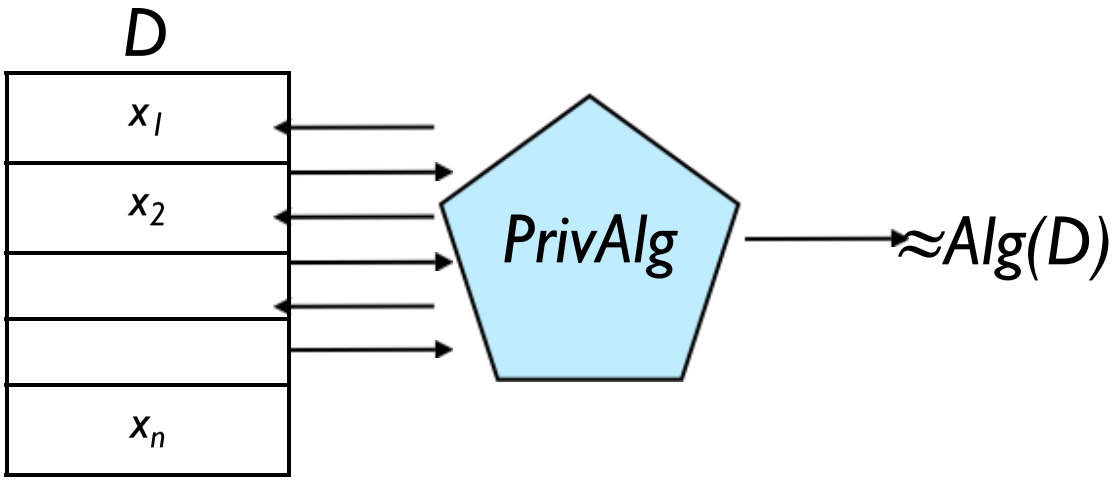
Differential Privacy

[DN03,BDMN05,DMNS06...]



Differential Privacy

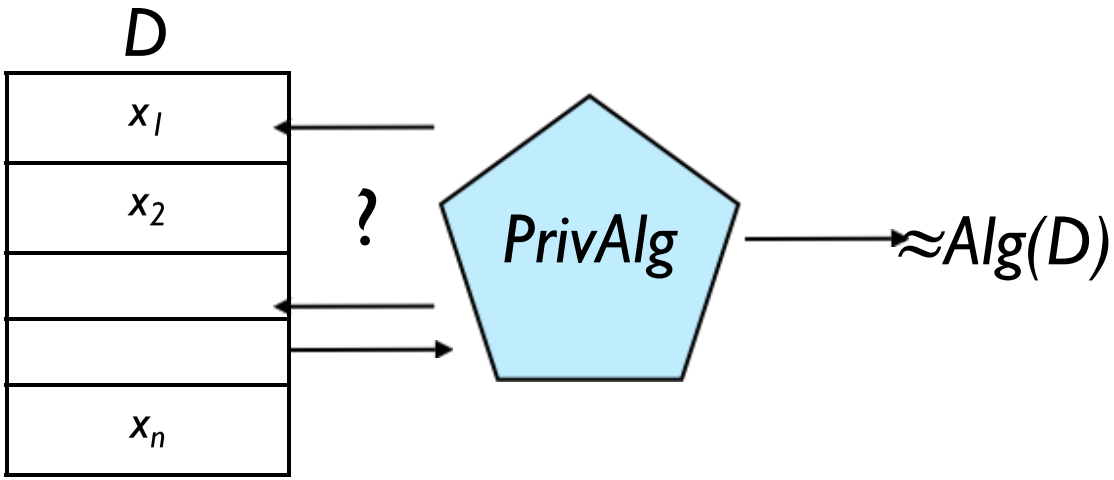
[DN03,BDMN05,DMNS06...]



- Design a private version of *Alg*
  - Should not reveal information about individual rows
- PrivAlg should be as useful and efficient as Alg

Differential Privacy

[DN03,BDMN05,DMNS06...]



- Design a private version of *Alg*
  - Should not reveal information about individual rows
- PrivAlg should be as useful and efficient as Alg

Statistical Queries Model [K98]

Statistical Query:

Predicate on $\{0,1\}^d$

$q: \{0,1\}^d \rightarrow \{0,1\}$

Query on D

$q(D) = E_{x \sim D}[q(x)]$

$D \in (\{0,1\}^d)^n$

x_1
x_2
x_n

Statistical Queries Model [K98]

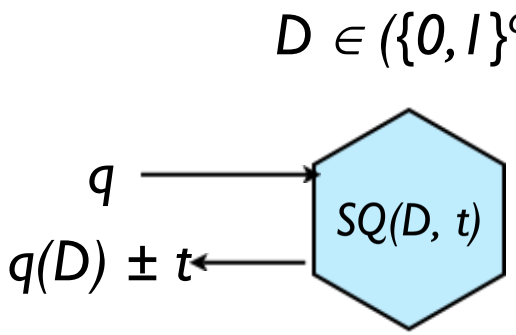
Statistical Query:

Predicate on $\{0,1\}^d$

$q: \{0,1\}^d \rightarrow \{0,1\}$

Query on D

$q(D) = E_{x \sim D}[q(x)]$



Statistical Queries Model:

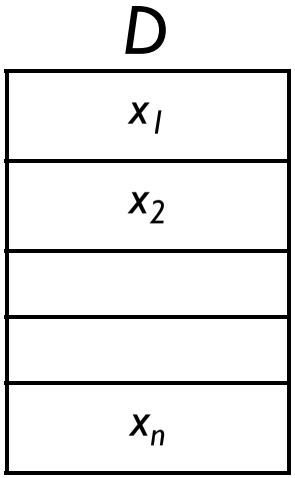
Only access D through an "SQ Oracle"

Differential Privacy with SQs

- Early results in differential privacy show how to simulate the SQ oracle while satisfying differential privacy [BDMN05, DMNS06]

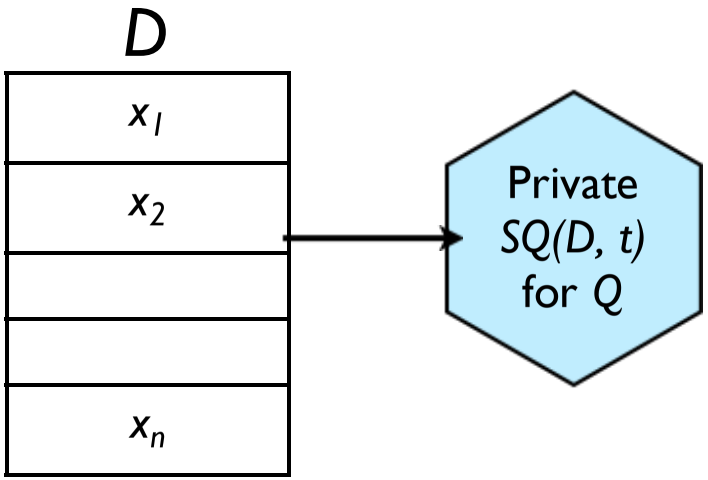
  - Simulation will be successful if #rows >> #queries ($n >> |Q|$)

Data Release



(Large) family of SQs, Q

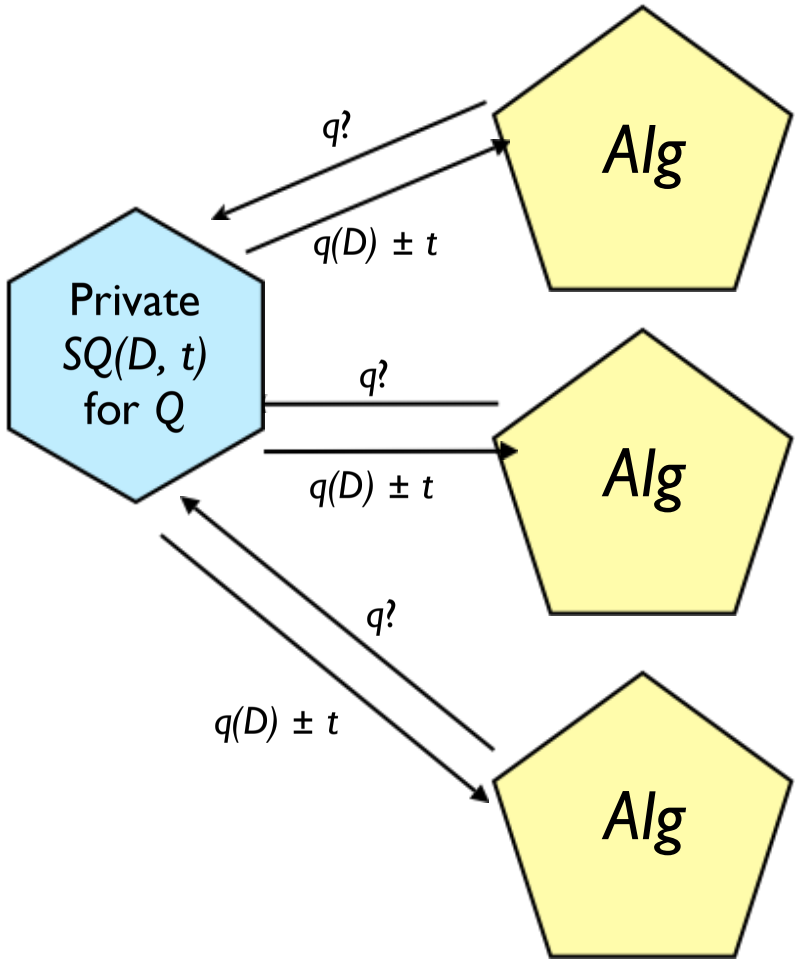
Statistical Query Release



(Large) family of SQs, Q

Statistical Query Release

(Large) family of SQs, Q



Statistical Query Release

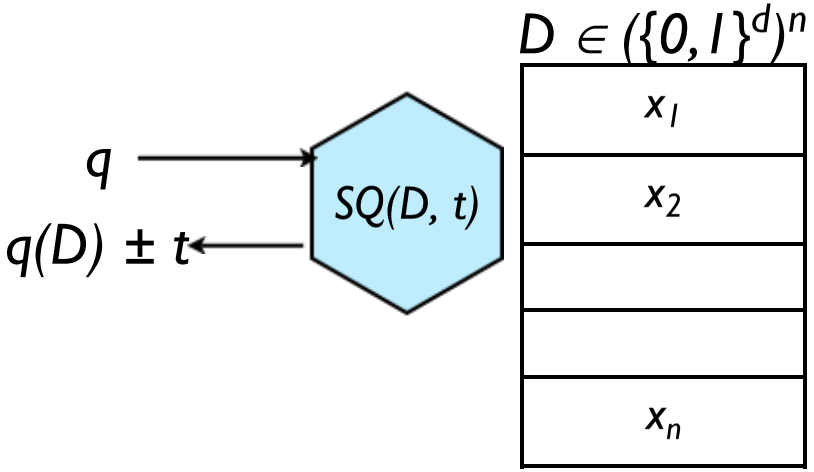
Statistical Query:

Predicate on $\{0,1\}^d$

$q: \{0,1\}^d \rightarrow \{0,1\}$

Query on D

$q(D) = E_{x \sim D}[q(x)]$



Statistical Queries Model:

Only access D through an "SQ Oracle"

Statistical Query Release:

Given family of queries Q, want to release $q(D) \pm \alpha$ using only an SQ oracle

Statistical Query Release

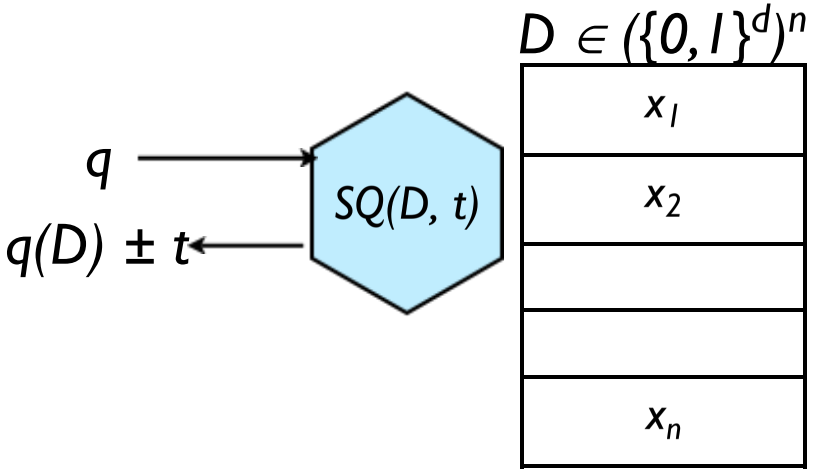
Statistical Query:

Predicate on $\{0,1\}^d$

$q: \{0,1\}^d \rightarrow \{0,1\}$

Query on D

$q(D) = E_{x \sim D}[q(x)]$



Statistical Queries Model:

Only access D through an "SQ Oracle"

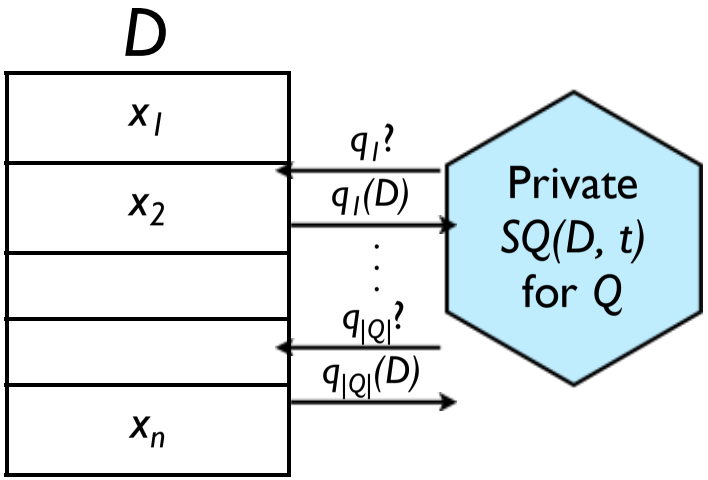
Statistical Query Release:

Given family of queries Q, want to release $q(D) \pm \alpha$ using only an SQ oracle

$q(D) \pm \alpha$ for all $q \in Q$ (worse-case error)

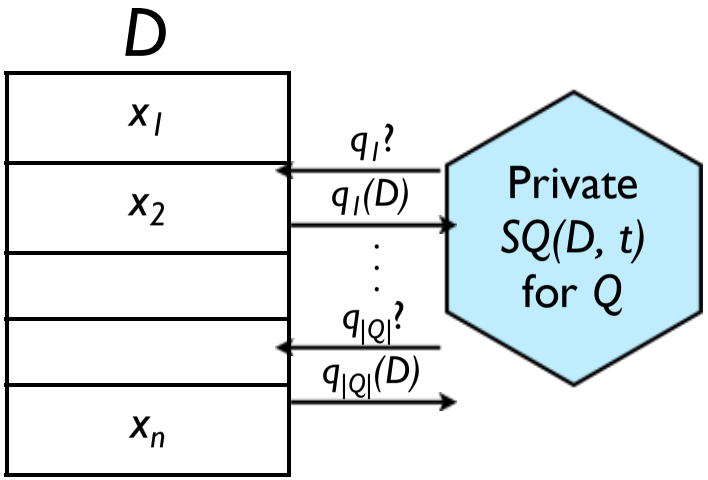
for average $q \in Q$ (average error)

Trivial Solution for SQ Release



(Large) family of SQs, Q

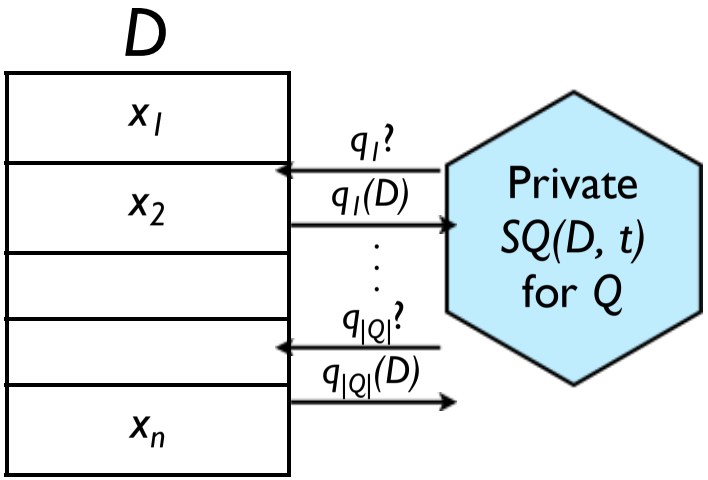
Trivial Solution for SQ Release



(Large) family of SQs, Q

- Two problems:
  - Might require n > $|Q|$
  - Very slow, time $|Q|$

Trivial Solution for SQ Release



(Large) family of SQs, Q

- Two problems:
  - Might require n > $|Q|$
  - Very slow, time $|Q|$

Data Release with SQs

- Inefficient algorithms exist for data release [BLR08,DNRRV09,DRV09,RR10,HR10,HLM11...]
  - Can simulate an SQ oracle even when #queries >> #rows ($|Q| >> n$)
  - These all use a trivial solution for data release

Data Release with SQs

- Inefficient algorithms exist for data release [BLR08,DNRRV09,DRV09,RR10,HR10,HLM11...]
  - Can simulate an SQ oracle even when #queries >> #rows ($|Q| >> n$)
  - These all use a trivial solution for data release
- We don't have efficient algorithms, even for very simple families of queries

Data Release with SQs

- Inefficient algorithms exist for data release [BLR08,DNRRV09,DRV09,RR10,HR10,HLM11...]
  - Can simulate an SQ oracle even when #queries >> #rows ($|Q| >> n$)
  - These all use a trivial solution for data release
- We don't have efficient algorithms, even for very simple families of queries
- Main Question: Are there non-trivial algorithms for the SQ release problem?

Our Results

- Theorem: There exists an algorithm that releases all 3^d conjunction queries with *constant average error* with only *poly(d)* queries and *poly(d)* time

- Technique:

  - Answers to the queries for a given database define *submodular function*

  - New efficient algorithm for learning submodular functions

- Immediately gives a *poly(d)*-time private algorithm

Our Results

- Theorem: Number of SQs needed to release Q is approximately the number of SQs needed to *agnostically learn* Q

  - Query-efficient reduction from agnostic learning to query release and vice versa

Our Results

- Theorem: Number of SQs needed to release Q is approximately the number of SQs needed to *agnostically learn* Q
  - Query-efficient reduction from agnostic learning to query release and vice versa
- Corollary: No algorithm making *poly(d)* SQs can release monotone conjunctions with *subconstant worst-case error*
  - Our learning-to-release reduction + lower bounds for agnostic SQ learning monotone disjunctions [F10]

Our Results

- Theorem: There exists an algorithm that releases all 2^d monotone disjunctions with *constant average error* with only *poly(d)* queries and *poly(d)* time

Monotone Disjunctions

Mon. Disjunction Query:

Predicate on $\{0,1\}^d$

$q_S(x_1,\ldots,x_d) = \vee_{i \in S} x_i \quad S \subseteq \{1,\ldots,d\}$

$D \in (\{0,1\}^d)^n$

1	2	3	...	d
1	1	0		1
0	0	0		0
0	0	0		0
0	1	1		1
0	0	1		1
1	0	0		1
0	0	0		0
0	0	1		1
0	0	1		1
0	0	0		0

Monotone Disjunctions

Mon. Disjunction Query:

Predicate on $\{0,1\}^d$

$q_S(x_1,\ldots,x_d) = \vee_{i\in S} x_i$ $\quad S \subseteq \{1,\ldots,d\}$

Query View

Distribution on $\{0,1\}^d$

Function View

Function f:

$2^{\{1,\ldots,d\}} \rightarrow [0,1]$:

$f_D(S) = q_S(D)$

$D \in (\{0,1\}^d)^n$

1	2	3	...	d
1	1	0		1
0	0	0		0
0	0	0		0
0	1	1		1
0	0	1		1
1	0	0		1
0	0	0		0
0	0	1		1
0	0	1		1
0	0	0		0

Monotone Disjunctions

Mon. Disjunction Query:
Predicate on $\{0,1\}^d$

$q_S(x_1,\ldots,x_d) = \vee_{i\in S} x_i$ $\quad S \subseteq \{1,\ldots,d\}$

Query View

Distribution on $\{0,1\}^d$

Releasing Q

Function View

Function f:
$2^{\{1,\ldots,d\}} \rightarrow [0,1]$:
$f_D(S) = q_S(D)$

"Learning" f

$D \in (\{0,1\}^d)^n$

1	2	3	...	d
1	1	0		1
0	0	0		0
0	0	0		0
0	1	1		1
0	0	1		1
1	0	0		1
0	0	0		0
0	0	1		1
0	0	1		1
0	0	0		0

Monotone Disjunctions

Mon. Disjunction Query:
Predicate on $\{0,1\}^d$

$q_S(x_1,\ldots,x_d) = \vee_{i\in S} x_i$ $\quad S \subseteq \{1,\ldots,d\}$

$D \in (\{0,1\}^d)^n$

1	2	3	...	d
1	1	0		1
0	0	0		0
0	0	0		0
0	1	1		1
0	0	1		1
1	0	0		1
0	0	0		0
0	0	1		1
0	0	1		1
0	0	0		0

Query View	Function View
Distribution on $\{0,1\}^d$	Function f: $2^{\{1,\ldots,d\}} \rightarrow [0,1]$: $f_D(S) = q_S(D)$
Releasing Q	"Learning" f
Average error	Function g s.t. $E[\lvert f(S) - g(S)\rvert] \leq \alpha$

Monotone Disjunctions

Mon. Disjunction Query:
Predicate on $\{0,1\}^d$
$q_S(x_1,\ldots,x_d) = \vee_{i\in S} x_i$ $\quad S \subseteq \{1,\ldots,d\}$

$D \in (\{0,1\}^d)^n$

1	2	3	...	d
1	1	0		1
0	0	0		0
0	0	0		0
0	1	1		1
0	0	1		1
1	0	0		1
0	0	0		0
0	0	1		1
0	0	1		1
0	0	0		0

Query View	Function View
Distribution on $\{0,1\}^d$	Function f: $2^{\{1,\ldots,d\}} \rightarrow [0,1]$: $f_D(S) = q_S(D)$
Releasing Q	"Learning" f
Average error	Function g s.t. $E[\lvert f(S) - g(S)\rvert] \leq \alpha$
SQs to D	Point queries to f

Monotone Disjunctions

Mon. Disjunction Query:

Predicate on $\{0,1\}^d$

$q_S(x_1,\ldots,x_d) = \vee_{i \in S} x_i$ $\quad S \subseteq \{1,\ldots,d\}$

Query View

Distribution on $\{0,1\}^d$

Function View

Function f:

$2^{\{1,\ldots,d\}} \rightarrow [0,1]$:

$f_D(S) = q_S(D)$

Useful Fact: f is a submodular function

$D \in (\{0,1\}^d)^n$

1	2	3	...	d
1	1	0		1
0	0	0		0
0	0	0		0
0	1	1		1
0	0	1		1
1	0	0		1
0	0	0		0
0	0	1		1
0	0	1		1
0	0	0		0

Submodular Functions

- Consider a set-valued function $f: 2^{\{1,\ldots,d\}} \rightarrow [0,1]$

Submodular Functions

- Consider a set-valued function $f: 2^{\{1,\ldots,d\}} \rightarrow [0,1]$
- Consider the "*marginal value of i on S*"

$$v_{S,i} = f(S \cup \{i\}) - f(S)$$

Submodular Functions

- Consider a set-valued function $f: 2^{\{1,\ldots,d\}} \rightarrow [0,1]$
- Consider the "*marginal value of i on S*"

$$v_{S,i} = f(S \cup \{i\}) - f(S)$$

- Marginal values are *decreasing*
  - For every $S \subseteq T \subseteq \{1,\ldots,d\}$, and every i in $\{1,\ldots,d\}$

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

$$v_{S,i} \geq v_{T,i}$$

Monotone Disjunctions

Mon. Disjunction Query:
Predicate on $\{0,1\}^d$

$q_S(x_1,\ldots,x_d) = \vee_{i\in S} x_i$ $\quad S \subseteq \{1,\ldots,d\}$

D over $\{0,1\}^d$

1	2	3	...	d
1	1	0		1
0	0	0		0
0	0	0		0
0	1	1		1
0	0	1		1
1	0	0		1
0	0	0		0
0	0	1		1
0	0	1		1
0	0	0		0

Query View

Distribution on $\{0,1\}^d$

Function View

Function f:
$2^{\{1,\ldots,d\}} \rightarrow [0,1]$:
$f_D(S) = q_S(D)$

Useful Fact: f is a submodular function

Consider $v_{\emptyset,d} = f(\emptyset \cup \{d\}) - f(\emptyset)$

Monotone Disjunctions

Mon. Disjunction Query:
Predicate on $\{0,1\}^d$

$q_S(x_1,\ldots,x_d) = \vee_{i\in S} x_i$ $\quad S \subseteq \{1,\ldots,d\}$

D over $\{0,1\}^d$

1	2	3	...	d
1	1	0		1
0	0	0		0
0	0	0		0
0	1	1		1
0	0	1		1
1	0	0		1
0	0	0		0
0	0	1		1
0	0	1		1
0	0	0		0

Query View

Distribution on $\{0,1\}^d$

Function View

Function f:
$2^{\{1,\ldots,d\}} \rightarrow [0,1]$:
$f_D(S) = q_S(D)$

Useful Fact: f is a submodular function

Consider $v_{\{1\},d} = f(\{1\}\cup\{d\}) - f(\{1\})$

Monotone Disjunctions

Mon. Disjunction Query:
Predicate on $\{0,1\}^d$

$q_S(x_1,\ldots,x_d) = \vee_{i\in S} x_i$ $\quad S \subseteq \{1,\ldots,d\}$

D over $\{0,1\}^d$

1	2	3	...	d
1	1	0		1
0	0	0		0
0	0	0		0
0	1	1		1
0	0	1		1
1	0	0		1
0	0	0		0
0	0	1		1
0	0	1		1
0	0	0		0

Query View

Distribution on $\{0,1\}^d$

Function View

Function f:
$2^{\{1,\ldots,d\}} \rightarrow [0,1]$:
$f_D(S) = q_S(D)$

Useful Fact: f is a submodular function

Consider $v_{\{1,2\},d} = f(\{1,2\}\cup\{d\}) - f(\{1,2\})$

Monotone Disjunctions

Mon. Disjunction Query:
Predicate on $\{0,1\}^d$

$q_S(x_1,\ldots,x_d) = \vee_{i\in S} x_i$ $\quad S \subseteq \{1,\ldots,d\}$

D over $\{0,1\}^d$

1	2	3	...	d
1	1	0		1
0	0	0		0
0	0	0		0
0	1	1		1
0	0	1		1
1	0	0		1
0	0	0		0
0	0	1		1
0	0	1		1
0	0	0		0

Query View

Distribution on $\{0,1\}^d$

Function View

Function f:
$2^{\{1,\ldots,d\}} \rightarrow [0,1]$:
$f_D(S) = q_S(D)$

Useful Fact: f is a submodular function

Consider $v_{\{1,2,3\},d} = f(\{1,2,3\}\cup\{d\}) - f(\{1,2,3\})$

Releasing Submodular Functions

Our Theorem: There is an algorithm A^f s.t. if f is submodular, A^f makes $d\tilde{O}(1/\alpha^2)$ queries to f, runs in $d\tilde{O}(1/\alpha^4)$ time, and computes $g(S)$ s.t.

$$E(|f(S) - g(S)|) \leq \alpha$$

Releasing Submodular Functions

Our Theorem: There is an algorithm A^f s.t. if f is submodular, A^f makes $d^{\tilde{O}(1/\alpha^2)}$ queries to f, runs in $d^{\tilde{O}(1/\alpha^4)}$ time, and computes $g(S)$ s.t.

$$E(|f(S) - g(S)|) \leq \alpha$$

Corollary: There is a private algorithm $A(D)$ that runs in $d^{\tilde{O}(1/\alpha^2)}$ time and releases all monotone disjunctions with average error α as long as

$$n > d^{\tilde{O}(1/\alpha^2)}$$

Releasing Submodular Functions

Our Theorem: There is an algorithm A^f s.t. if f is submodular, A^f makes $d^{\tilde{O}(1/\alpha^2)}$ queries to f, runs in $d^{\tilde{O}(1/\alpha^4)}$ time, and computes $g(S)$ s.t.

$$E(|f(S) - g(S)|) \leq \alpha$$

Corollary: There is a private algorithm $A(D)$ that runs in $d^{\tilde{O}(1/\alpha^2)}$ time and releases all monotone disjunctions with average error α as long as

$$n > d^{\tilde{O}(1/\alpha^2)}$$

The first poly-time algorithm capable of releasing monotone disjunctions even with constant average error

Proof Outline

- If we're very lucky, f will be an α^2-Lipschitz submodular function
  - Lipschitz submodular functions are very concentrated around their expectation [BLM00, V09, BH11]

Concentration for Submodular Fns

Theorem [BLM00, V09]: For any c-Lipschitz submodular function $f: 2^{\{1,\ldots,d\}} \to [0,1]$

$$Pr[|f(S)-E[f(S)]| > ct] \leq exp(-t/4)$$

Concentration for Submodular Fns

Theorem [BLM00, V09]: For any c-Lipschitz submodular function $f: 2^{\{1,\ldots,d\}} \to [0,1]$

$$Pr_S[|f(S)-E[f(S)]| > ct] \leq exp(-t/4)$$

Compare to Azuma's Inequality (applies to any c-Lipschitz function)

$$Pr_S[|f(S)-E(f(S))| > ct] \leq exp(-t^2/4d)$$

Proof Outline

- If we're very lucky, f will be an α^2-Lipschitz submodular function
  - Lipschitz submodular functions are very concentrated around their expectation [BLM00, V09, BH11]
  - $g(S) = E_T[f(T)]$ has at most α average error

Proof Outline

- If we're very lucky, f will be an α^2-Lipschitz submodular function
  - Lipschitz submodular functions are very concentrated around their expectation [BLM00, V09, BH11]
  - $g(S) = E_T[f(T)]$ has at most α average error
- Can "decompose" f into a family $\{g^T\}$ of at most $d^{\tilde{O}(1/\alpha^2)}$ α^2-Lipschitz submodular functions

Our Algorithm: Decomposition

Our Algorithm: Decomposition

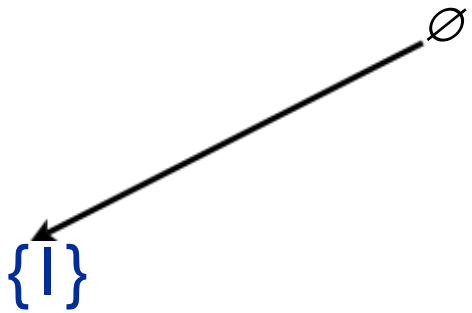
$\varnothing$

Our Algorithm: Decomposition

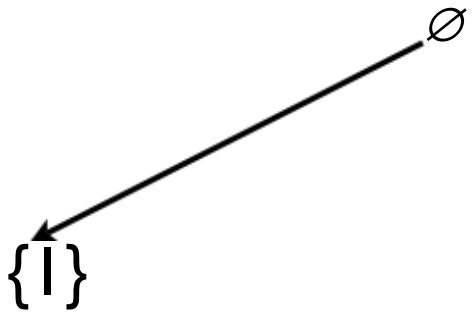
$\varnothing$ $\{1\}$

$f(\{1\}) - f(\varnothing) > \alpha^2$

Our Algorithm: Decomposition



Our Algorithm: Decomposition

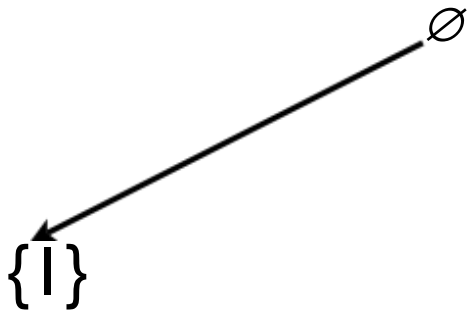


{2}

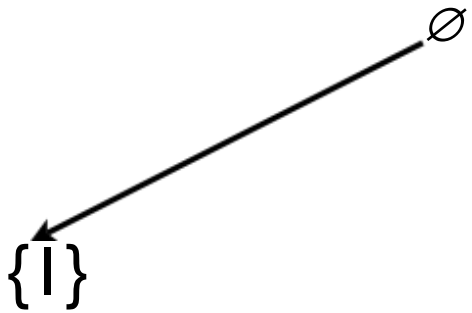
$f(\{2\}) - f(\emptyset) \leq \alpha^2$

$f(\{1,2\}) - f(\{1\}) \leq \alpha^2$

Our Algorithm: Decomposition



Our Algorithm: Decomposition

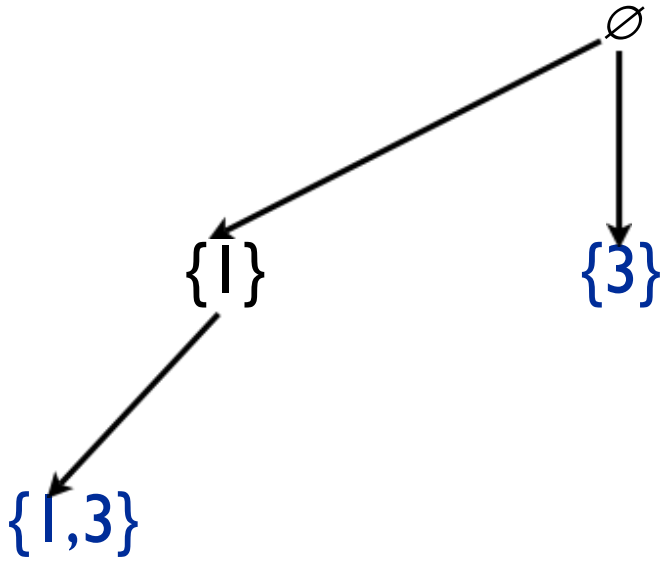


{3}

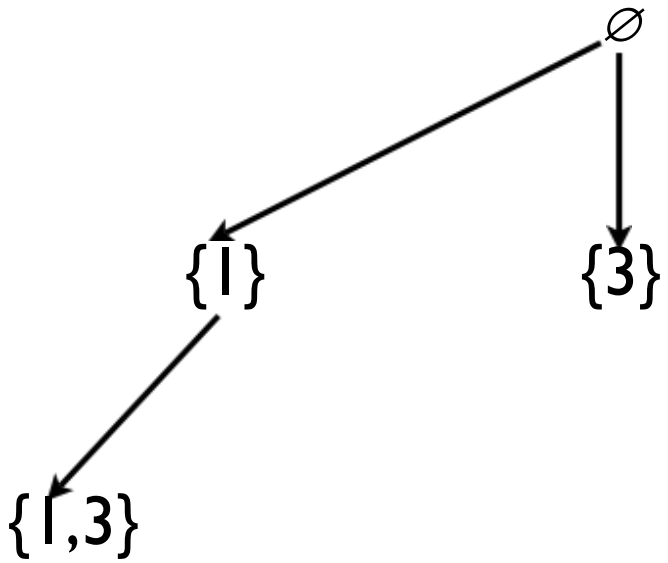
$f(\{3\}) - f(\emptyset) > \alpha^2$

$f(\{1,3\}) - f(\{1\}) > \alpha^2$

Our Algorithm: Decomposition



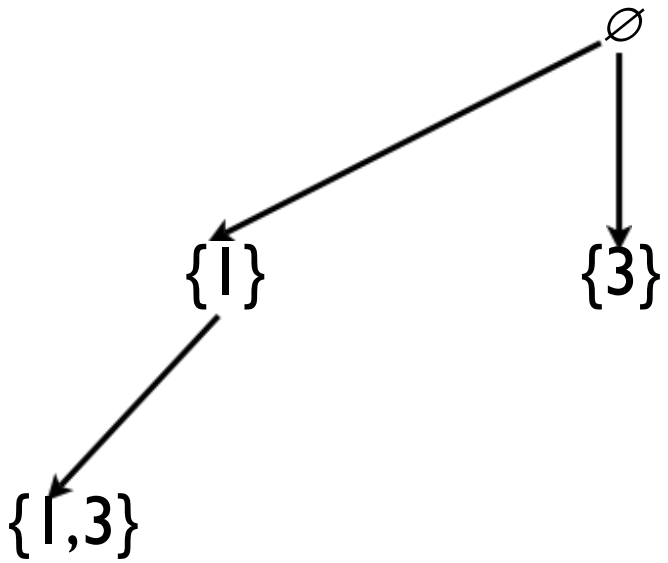
Our Algorithm: Decomposition



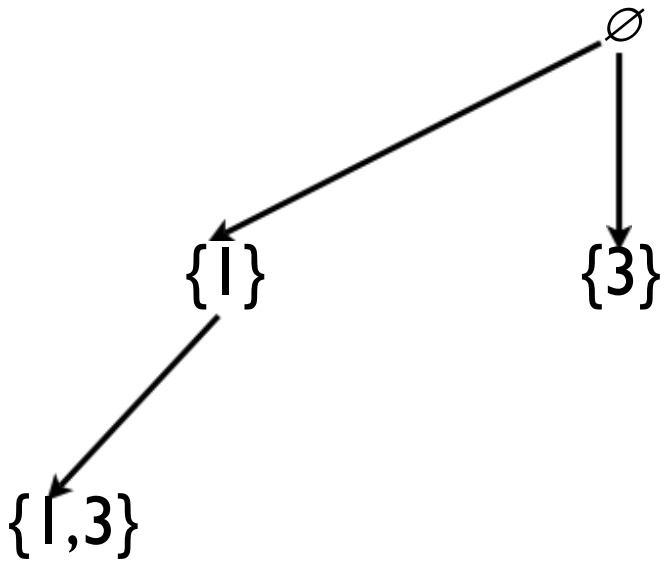
{4}

$$f(\{4\}) - f(\emptyset) \leq \alpha^2$$
$$f(\{1,4\}) - f(\{1\}) \leq \alpha^2$$
$$f(\{3,4\}) - f(\{3\}) \leq \alpha^2$$
$$f(\{1,3,4\}) - f(\{1,3\}) \leq \alpha^2$$

Our Algorithm: Decomposition



Our Algorithm: Decomposition



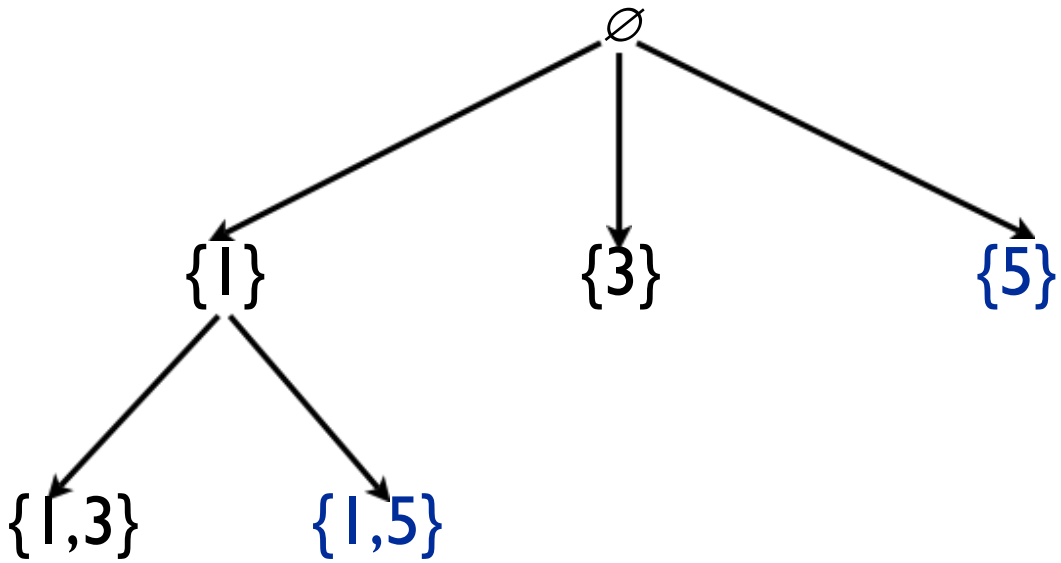
{5}

$f(\{5\}) - f(\varnothing) > \alpha^2$

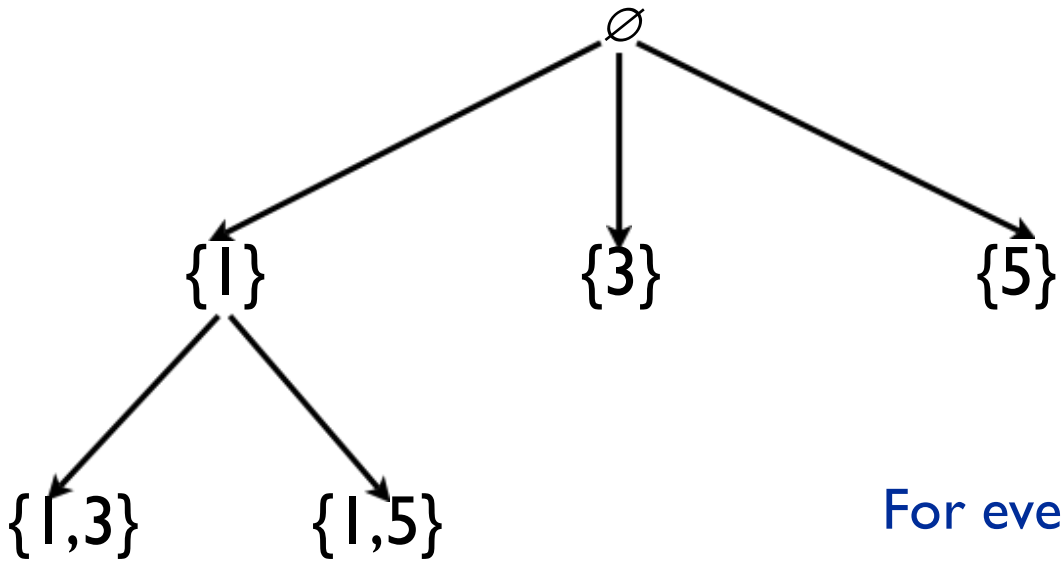
$f(\{1,5\}) - f(\{1\}) > \alpha^2$

$f(\{1,3,5\}) - f(\{1,3\}) \leq \alpha^2$

Our Algorithm: Decomposition



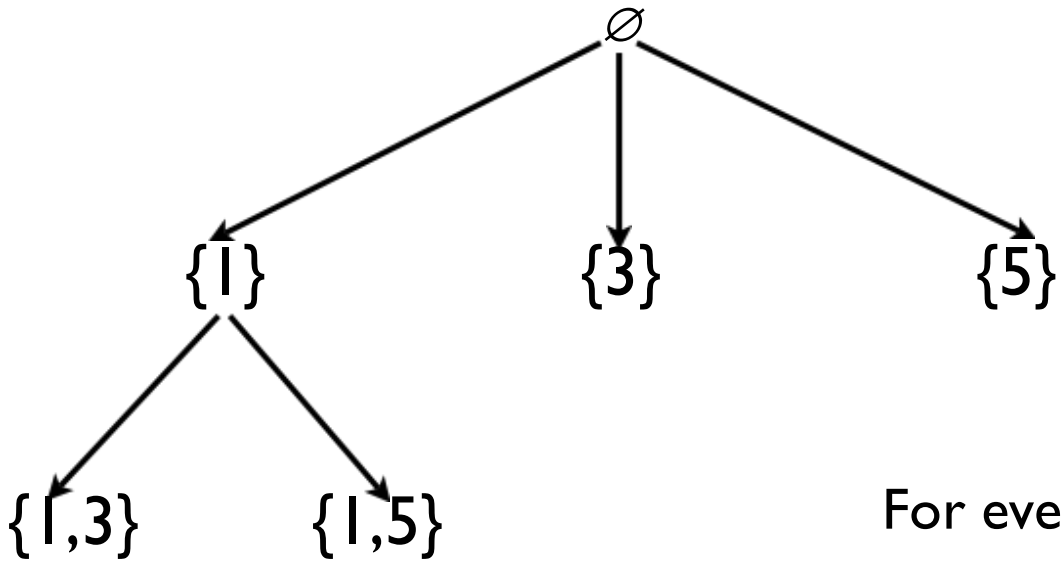
Our Algorithm: Decomposition



For every edge (S,T)

$f(T) - f(S) > \alpha^2$

Our Algorithm: Decomposition



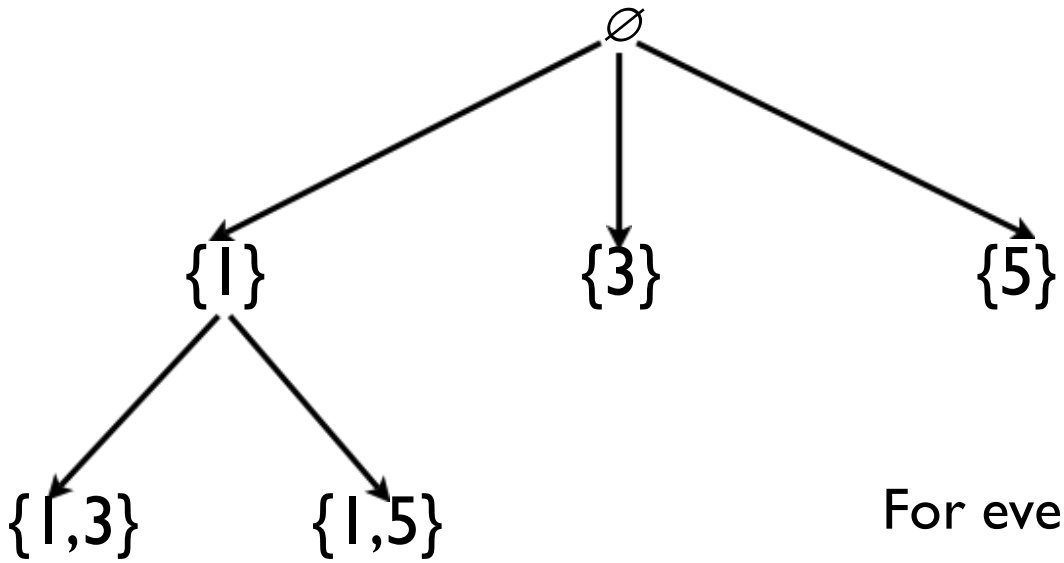
For every edge (S,T)

$f(T) - f(S) > \alpha^2$

On every path $(\emptyset,T)$

$f(T) - f(\emptyset) \leq 1$

Our Algorithm: Decomposition



For every edge (S,T)

$f(T) - f(S) > \alpha^2$

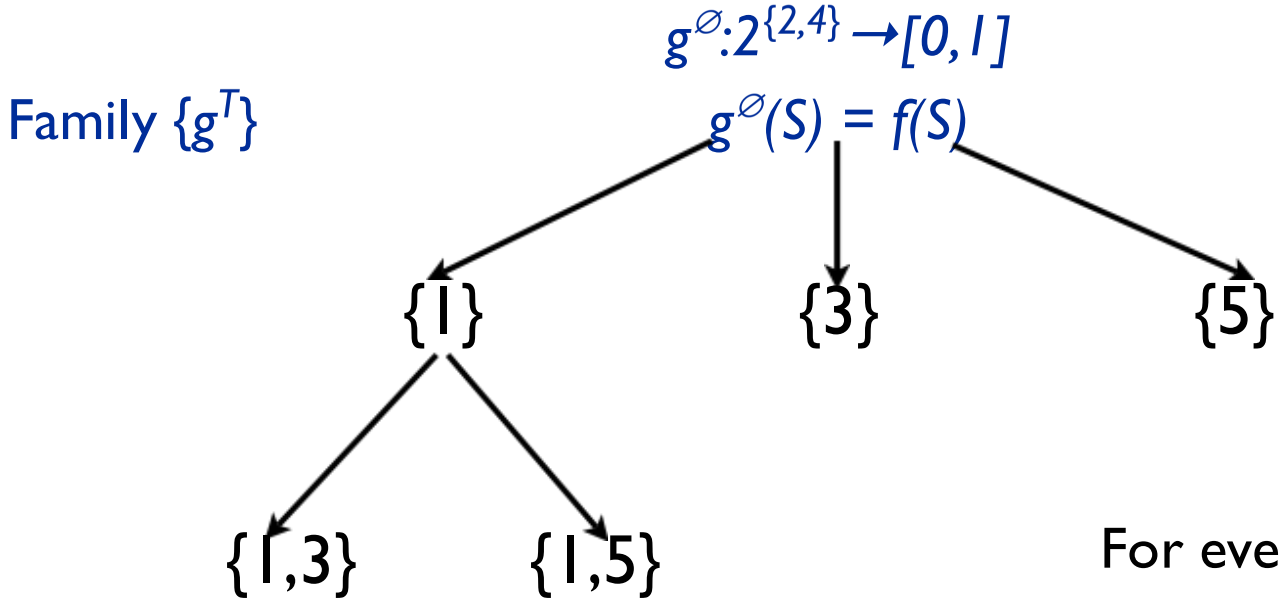
On every path $(\emptyset,T)$

$f(T) - f(\emptyset) \leq 1$

Every path has at most $1/\alpha^2$ nodes

At most $d^{\tilde{O}(1/\alpha^2)}$ total nodes

Our Algorithm: Decomposition



For every edge (S,T)

$f(T) - f(S) > \alpha^2$

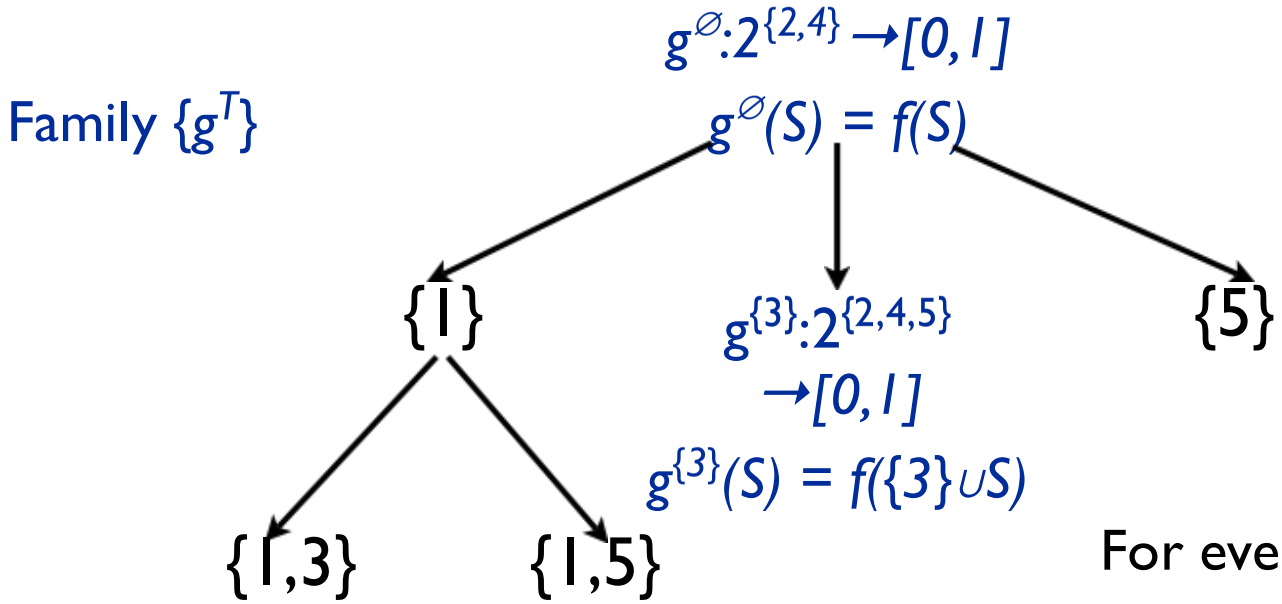
On every path $(\varnothing,T)$

$f(T) - f(\varnothing) \leq 1$

Every path has at most $1/\alpha^2$ nodes

At most $d^{\tilde{O}(1/\alpha^2)}$ total nodes

Our Algorithm: Decomposition



For every edge (S,T)

$f(T) - f(S) > \alpha^2$

On every path $(\varnothing,T)$

$f(T) - f(\varnothing) \leq 1$

Every path has at most $1/\alpha^2$ nodes

At most $d^{\tilde{O}(1/\alpha^2)}$ total nodes

Our Algorithm: Decomposition

$g^{\emptyset}: 2^{\{2,4\}} \rightarrow [0,1]$

$g^{\emptyset}(S) = f(S)$

{1}

$g^{\{3\}}: 2^{\{2,4,5\}} \rightarrow [0,1]$

$g^{\{3\}}(S) = f(\{3\} \cup S)$

{5}