Statistical Algorithms and the Planted Clique Problem

(and random graphs, linear equations, & machine learning)

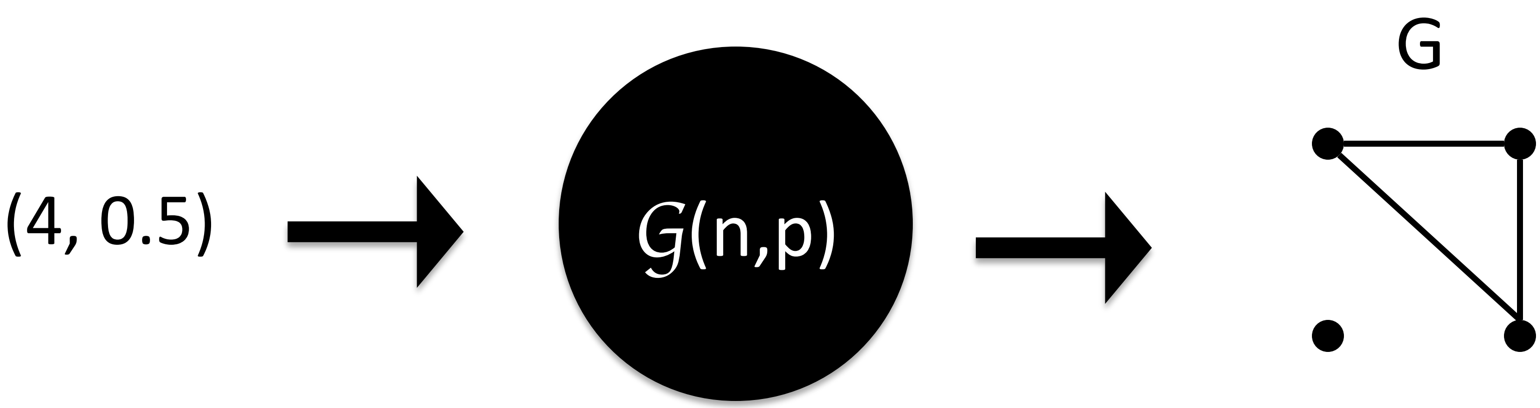
IDS Seminar

Lev Reyzin
UIC

random graphs

Erdős-Rényi Random Graphs

$\mathcal{G}(n,p)$ generates graph G on n vertices by including each possible edge independently with probability p.

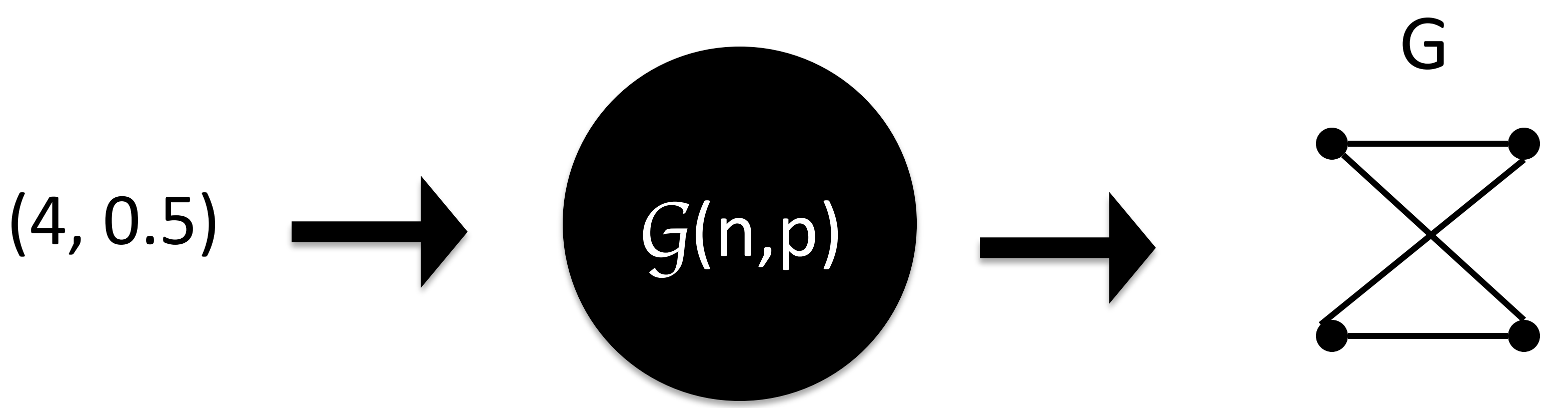


Erdős-Rényi Random Graphs

$\mathcal{G}$(n,p) generates graph G on n vertices by including each possible edge independently with probability p.

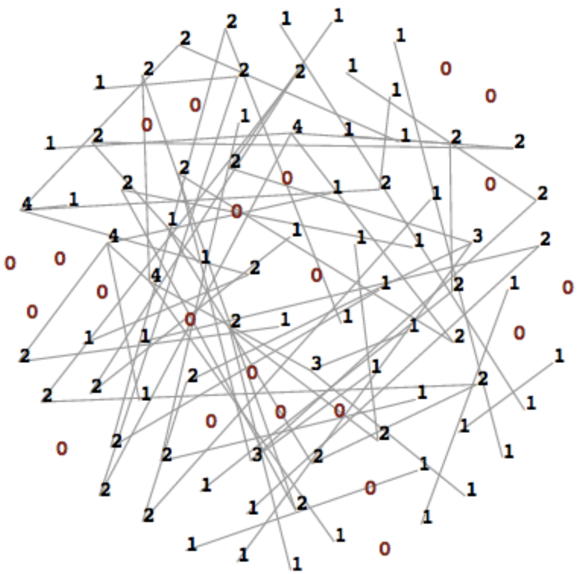
Erdős-Rényi Random Graphs

$\mathcal{G}(n,p)$ generates graph G on n vertices by including each possible edge independently with probability p.

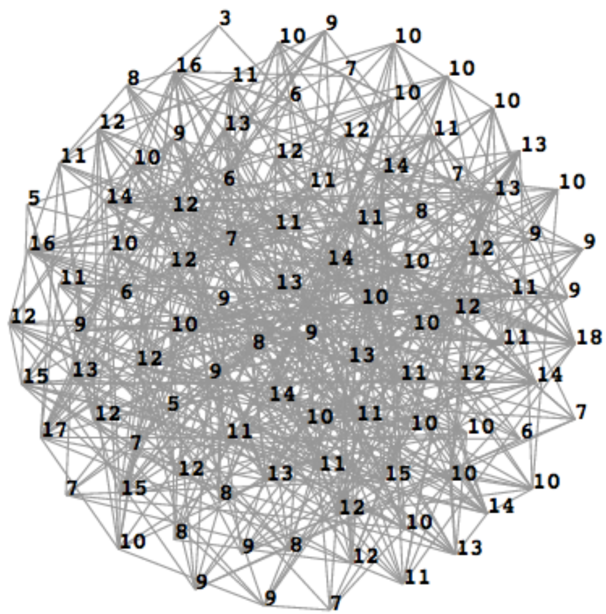


Typical Examples

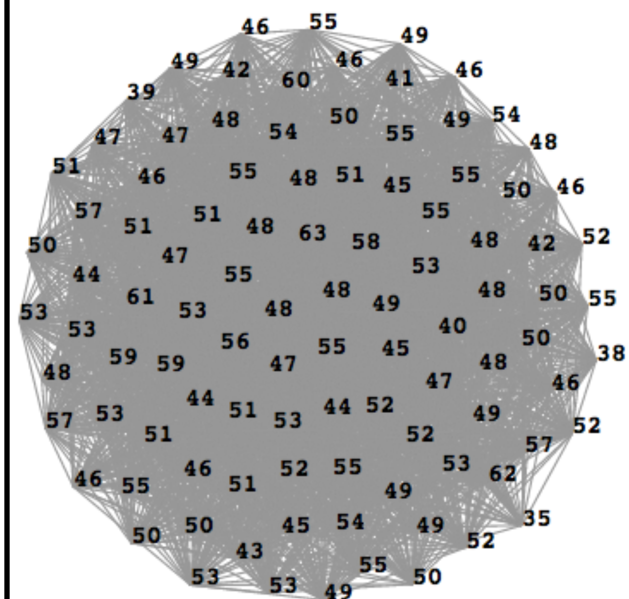
n = 100, p = 0.01



n = 100, p = 0.1



n = 100, p = 0.5



Created using software by Christopher Manning, available on
http://bl.ocks.org/christophermanning/4187201

Erdős-Rényi Random Graphs

E-R random graphs are an interesting "object" of study in combinatorics.

- When does G have a giant component?
- When is G connected?
- How large is the largest clique in G?

Erdős-Rényi Random Graphs

E-R random graphs are an interesting "object" of study in combinatorics.

- When does G have a giant component?

  when $np \rightarrow c > 1$

- When is G connected?

  sharp connectivity threshold at $p = \ln/n$

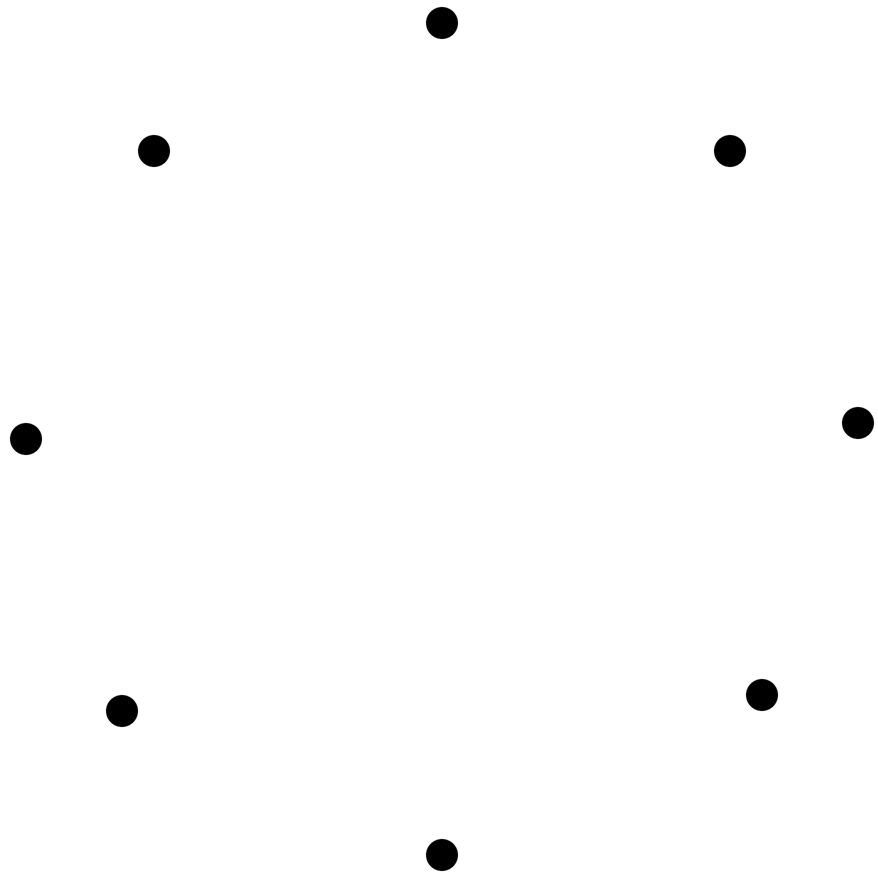
- How large is the largest clique in G?

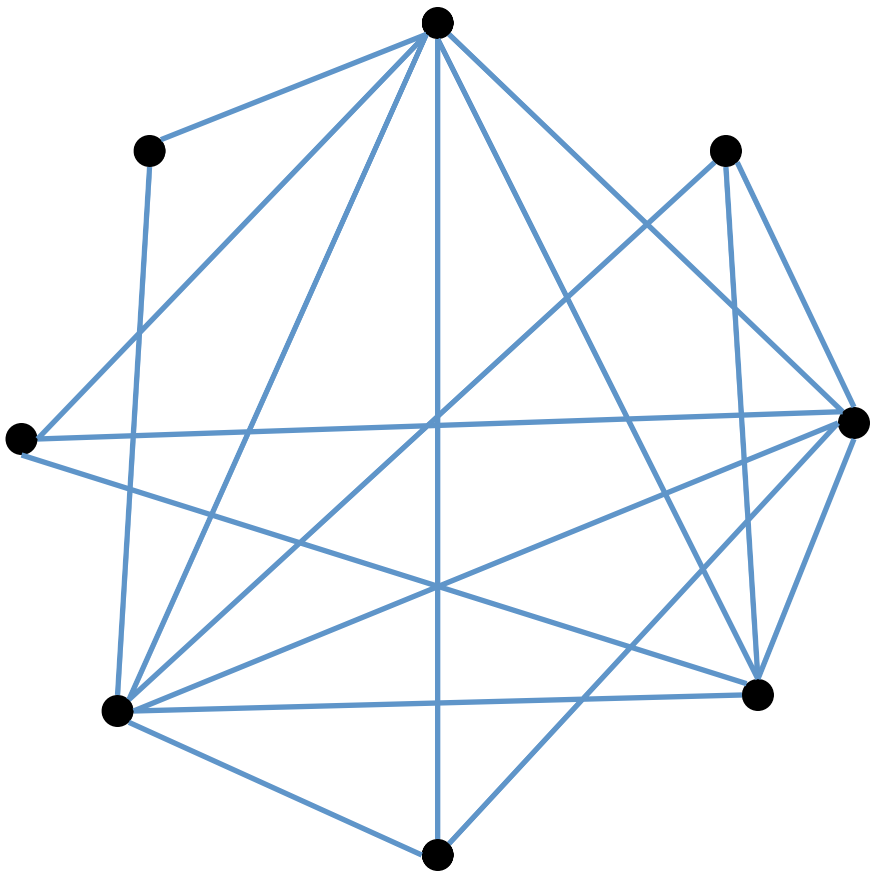
  for $p=½$, largest clique has size $k(n) \approx 2\lg_2(n)$

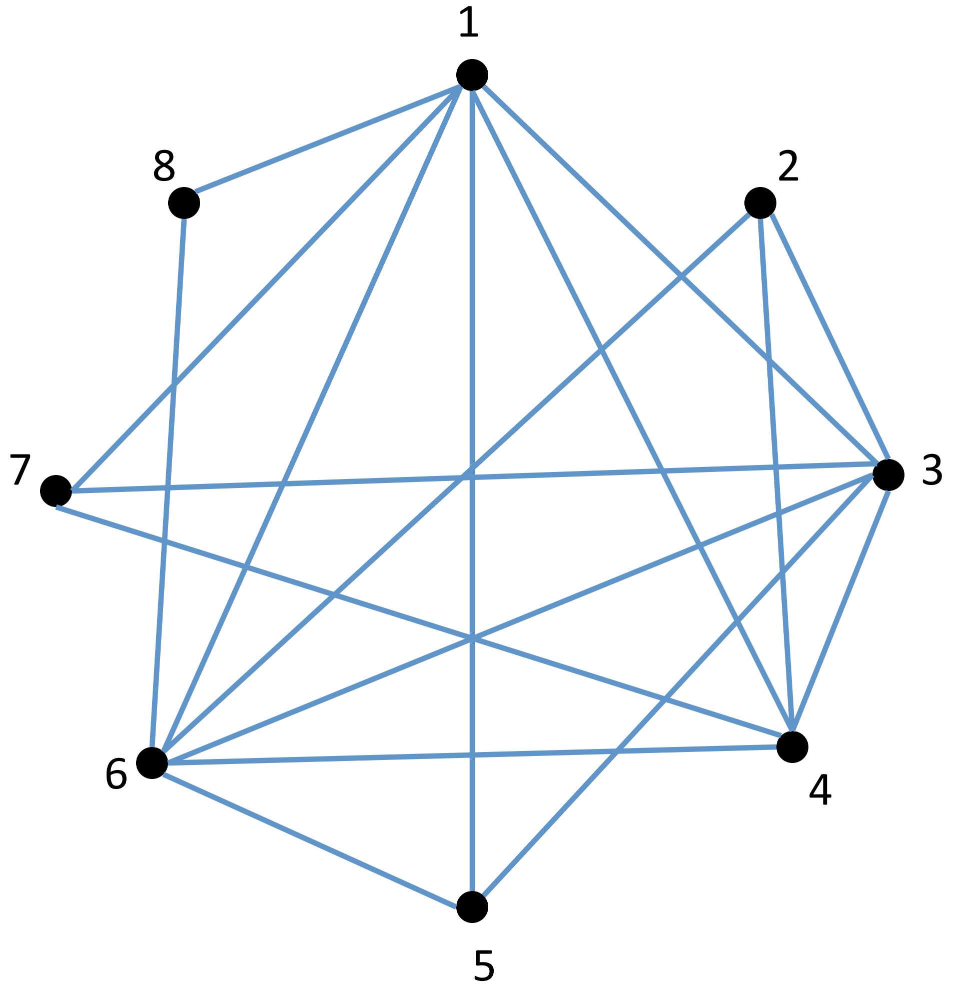
w.h.p. for G ~ $\mathcal{G}$(n,½), k(n) ≈ $2\lg_2(n)$

why?

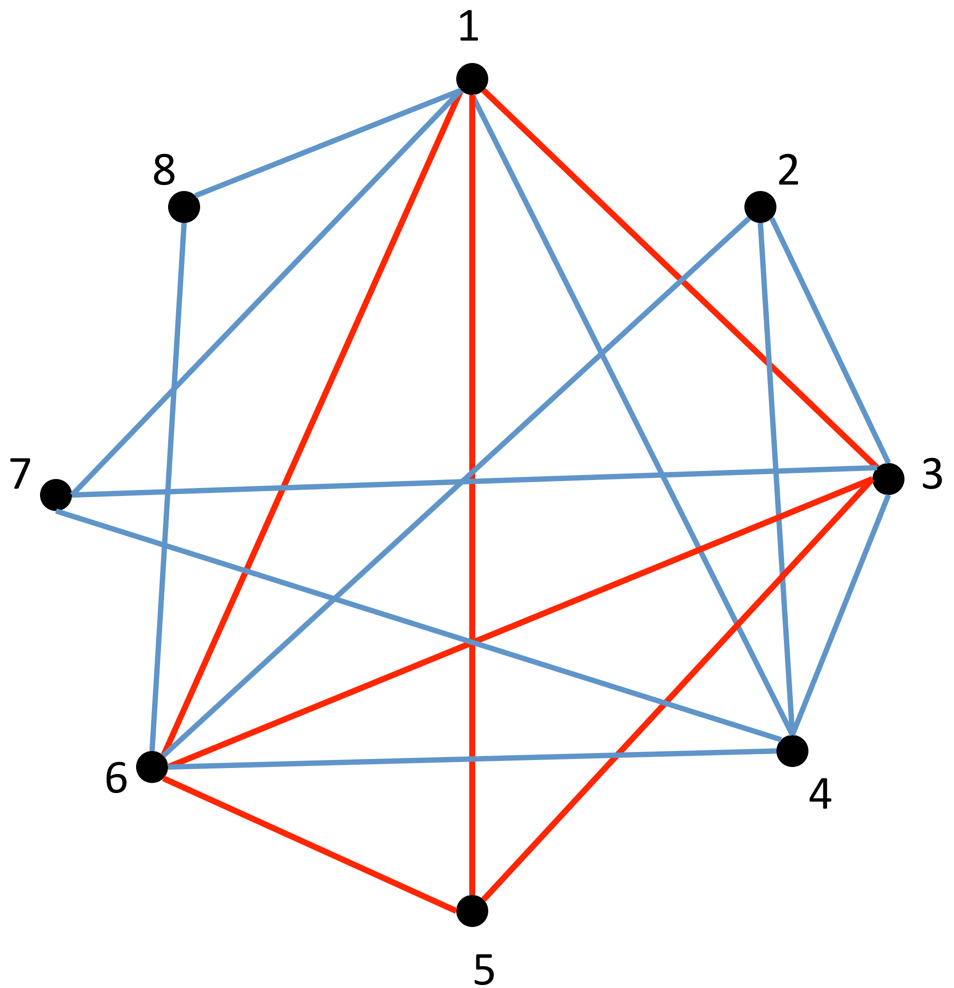
- let X_k be the number of cliques in G ~ $\mathcal{G}$(n,.5)
- $E[X_K] = \binom{n}{k} 2^{-\binom{k}{2}}$ < 1 for k > ≈ $2\lg_2 n$
- in fact, (for large n) the largest clique is almost certainly k(n) = $2\lg_2(n)$ or $2\lg_2(n)+1$ [Matula '76]







Where is the largest clique?


1
8
2
7
3
6
4
5

Finding Large Cliques

for worst-case graphs:

finding largest clique is NP-Hard.

(very very unlikely to have efficient algorithms)

Finding Large Cliques

for worst-case graphs:

finding largest clique is NP-Hard.

(very very unlikely to have efficient algorithms)

W[1]-Hard

(very likely gets harder as target cliques grow)

Finding Large Cliques

for <u>worst-case</u> graphs:

finding largest clique is NP-Hard.
(very very unlikely to have efficient algorithms)
W[1]-Hard
(very likely gets harder as target cliques grow)
hard to approximate to $n^{1-\varepsilon}$ for any $\varepsilon > 0$
(give up)

Finding Large Cliques

for <u>worst-case</u> graphs:

finding largest clique is NP-Hard.

(very very unlikely to have efficient algorithms)

W[1]-Hard

(very likely gets harder as target cliques grow)

hard to approximate to $n^{1-\varepsilon}$ for any $\varepsilon > 0$

(give up)

hope: in E-R random graphs, finding large cliques is easier.

Finding Large Cliques in G ~ $\mathcal{G}(n,½)$

- Finding a clique of size = $\lg_2(n)$ is "easy"

```
initialize T = Ø, S = V
while (S ≠ Ø) {
    pick random v∈S and add v to T
    remove v and its non-neighbors from S
}
return T
```

Finding Large Cliques in G ~ $\mathcal{G}(n,½)$

- Finding a clique of size = $\lg_2(n)$ is "easy"

```
initialize T = Ø, S = V
while (S ≠ Ø) {
    pick random v∈S and add v to T
    remove v and its non-neighbors from S
}
return T
```

- Conjecture [Karp '76]: for any $\varepsilon > 0$, there's no efficient method to find cliques of size $(1+\varepsilon)\lg_2 n$ in E-R random graphs.

Finding Large Cliques in G ~ $\mathcal{G}(n,½)$

- Finding a clique of size = $\lg_2(n)$ is "easy"

```
initialize T = Ø, S = V
while (S ≠ Ø) {
    pick random v∈S and add v to T
    remove v and its non-neighbors from S
}
return T
```

- Conjecture [Karp '76]: for any $\varepsilon > 0$, there's no efficient method to find cliques of size $(1+\varepsilon)\lg_2 n$ in E-R random graphs.

still open (would imply P ≠ NP)

Summary

In E-R random graphs

- clique of size $2\lg_2 n$ exists
- can efficiently find clique of size $\lg_2 n$
- likely cannot efficiently find cliques size $(1+\varepsilon)\lg_2 n$

What to do?

Summary

In E-R random graphs

- clique of size $2\lg_2 n$ exists
- can efficiently find clique of size $\lg_2 n$
- likely cannot efficiently find cliques size $(1+\varepsilon)\lg_2 n$

What to do?

- make the problem easier by “planting” a large clique to be found! [Jerrum ’92]

planted cliques

Planted Clique

the process: G ~ $\mathcal{G}$**(n,p,k)**

1. generate G ~ $\mathcal{G}$(n,p)
2. add clique to random subset of k < n vertices of G

Goal: given G ~ $\mathcal{G}$(n,p,k), find the k vertices where the clique was "planted" (algorithm knows values: n,p,k)

Progress on Planted Clique

For $G \sim \mathcal{G}(n,½,k)$, clearly no hope for $k \leq 2\lg_2 n + 1$.

For $k > 2\lg_2 n+1$, there is an "obvious" $n^{O(\lg n)}$-algorithm:

```
input: G from (n,1/2,k) with k > 2lg₂n+1
1) Check all S⊂V of size |S|=2lg₂n+2 for S
   that induces a clique in G.
2) For each v∈V, if (v,w) is edge for
   all w in S: S = S∪{v}
3) return S
```

Unfortunately, this is not polynomial time.

Progress on Planted Clique

For G ~ $\mathcal{G}$(n,½,k), clearly no hope for k ≤ $2\lg_2 n + 1$.

For k > $2\lg_2 n+1$, there is an "obvious" $n^{O(\lg n)}$-algorithm:

```
input: G from (n,1/2,k) with k > 2lg₂n+1
1) Check all S⊂V of size |S|=2lg₂n+2 for S
   that induces a clique in G.
2) For each v∈V, if (v,w) is edge for
   all w in S: S = S∪{v}
3) return S
```

What is the smallest value of k that we have a polynomial time algorithm for? Any guesses?

State-of-the-Art for Polynomial Time

- $k \geq c\,(n \lg n)^{1/2}$ is trivial. The degrees of the vertices in the plant "stand out." (proof via Hoeffding & union bound)

- $k = c\, n^{1/2}$ is best so far. [Alon-Krivelevich-Sudokov '98]

State-of-the-Art for Polynomial Time

- $k \geq c\ (n \lg n)^{1/2}$ is trivial. The degrees of the vertices in the plant "stand out." (proof via Hoeffding & union bound)
- $k = c\ n^{1/2}$ is best so far. [Alon-Krivelevich-Sudokov '98]

---

input: G from (n,1/2,k) with $k \geq 10\ n^{1/2}$

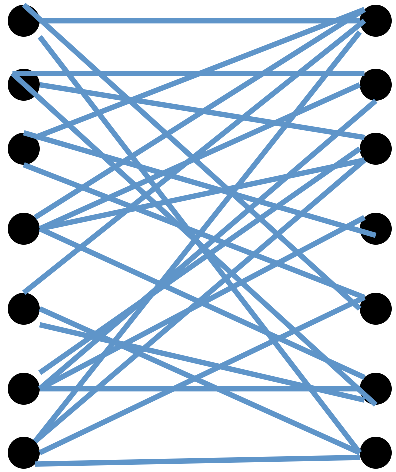
1) find 2^{nd} eigenvector v_2 of A(G)
2) Sort V by decreasing order of absolute values of coordinates of v_2. Let W be the top k vertices in this order.
3) Return Q, the set of vertices with ≥ ¾k neighbors in W

State-of-the-Art for Polynomial Time

- $k \geq c\,(n \lg n)^{1/2}$ is trivial. The degrees of the vertices in the plant "stand out." (proof via Hoeffding & union bound)

- $k = c\,n^{1/2}$ is best so far. [Alon-Krivelevich-Sudokov '98]

Bipartite Version

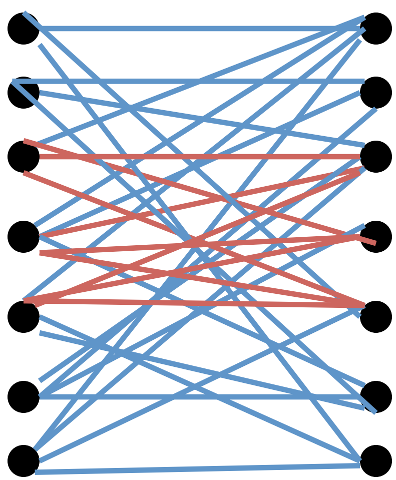


State-of-the-Art for Polynomial Time

- $k \geq c\,(n \lg n)^{1/2}$ is trivial. The degrees of the vertices in the plant "stand out." (proof via Hoeffding & union bound)

- $k = c\, n^{1/2}$ is best so far. [Alon-Krivelevich-Sudokov '98]

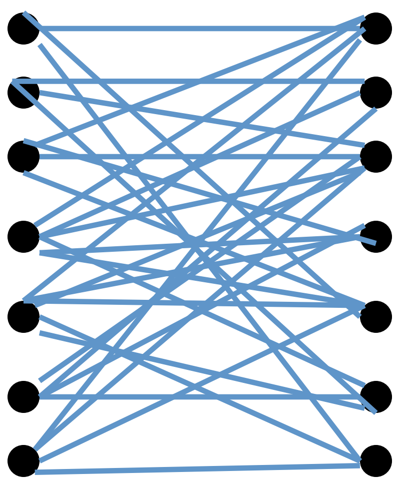
Bipartite Version



State-of-the-Art for Polynomial Time

- $k \geq c\,(n \lg n)^{1/2}$ is trivial. The degrees of the vertices in the plant "stand out." (proof via Hoeffding & union bound)
- $k = c\, n^{1/2}$ is best so far. [Alon-Krivelevich-Sudokov '98]

Bipartite Version



In fact, (bipartite) planted clique was recently used as alternate cryptographic primitive for $k < n^{1/2-\varepsilon}$. [Applebaum-Barak-Wigderson '09]

State-of-the-Art for Polynomial Time

- $k \geq c\,(n \lg n)^{1/2}$ is trivial. The degrees of the vertices in the plant "stand out." (proof via Hoeffding & union bound)
- $k = c\,n^{1/2}$ is best so far. [Alon-Krivelevich-Sudokov '98]

---

my goal: explain why there has been no progress on this problem past $n^{1/2}$. [FGVRX'13]

State-of-the-Art for Polynomial Time

- $k \geq c\,(n \lg n)^{1/2}$ is trivial. The degrees of the vertices in the plant "stand out." (proof via Hoeffding & union bound)

- $k = c\, n^{1/2}$ is best so far. [Alon-Krivelevich-Sudokov '98]

---

my goal: explain why there has been no progress on this problem past $n^{1/2}$. [FGVRX'13]

But first we have to discuss solving linear systems!

linear systems

Solving Linear Systems

n variables

m equations

$$Ax = b$$

m results

solve for n unknowns

the linear equations are over GF(2), ie $\{0,1\}^n$

Solving Random Linear Systems

n variables

m equations

$$Ax = b$$

random (A)

random (x)

m results

Solving Random Linear Systems

n variables

m equations

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

$$A \, x = b$$

random

m results

Solving Random Linear Systems

n variables

m equations

$$A\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} x \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = b$$

m results

Solving Random Linear Systems

n variables

m equations

$$\mathbf{A}\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \mathbf{x}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \mathbf{b}\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

m results

Solving Random Linear Systems

n variables

m equations

m results

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

A x = b

Solving Random Linear Systems

n variables

m equations

$$A x = b:\quad \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

m results

choose any m = poly(n)
solve for unique x in poly time.

How?

A Twist

entries of b flipped independently with prob. 1/100

n variables

m equations

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \cancel{1}\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

m results

choose any m = poly(n)

solve for x (that generated original b) in poly time.

How?

A Twist

entries of b flipped independently with prob. 1/100

n variables

m equations

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \to 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

m results

choose any m = poly(n)

solve for x (that generated original b) in poly time.

it's a big open question is theoretical CS called "noisy parity".

A Twist

entries of b flipped independently with prob. 1/100

n variables

m equations

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \text{ x } 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

A x = b

m results

choose any m = poly(n)

solve for x (that generated original b) in poly time.

current best is $2^{O(n/\lg n)}$ time. [BlumKW ’00]

A Twist

entries of b flipped independently with prob. 1/100

n variables

m equations

$$\underbrace{\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} 0 \\ 0 \\ \cancel{1}\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}}_{b}$$

m results

In fact, LPN was recently used as alternate cryptographic primitive. [Peikart '09]

learning theory

PAC Learning, in One Slide

[Valiant ’84]

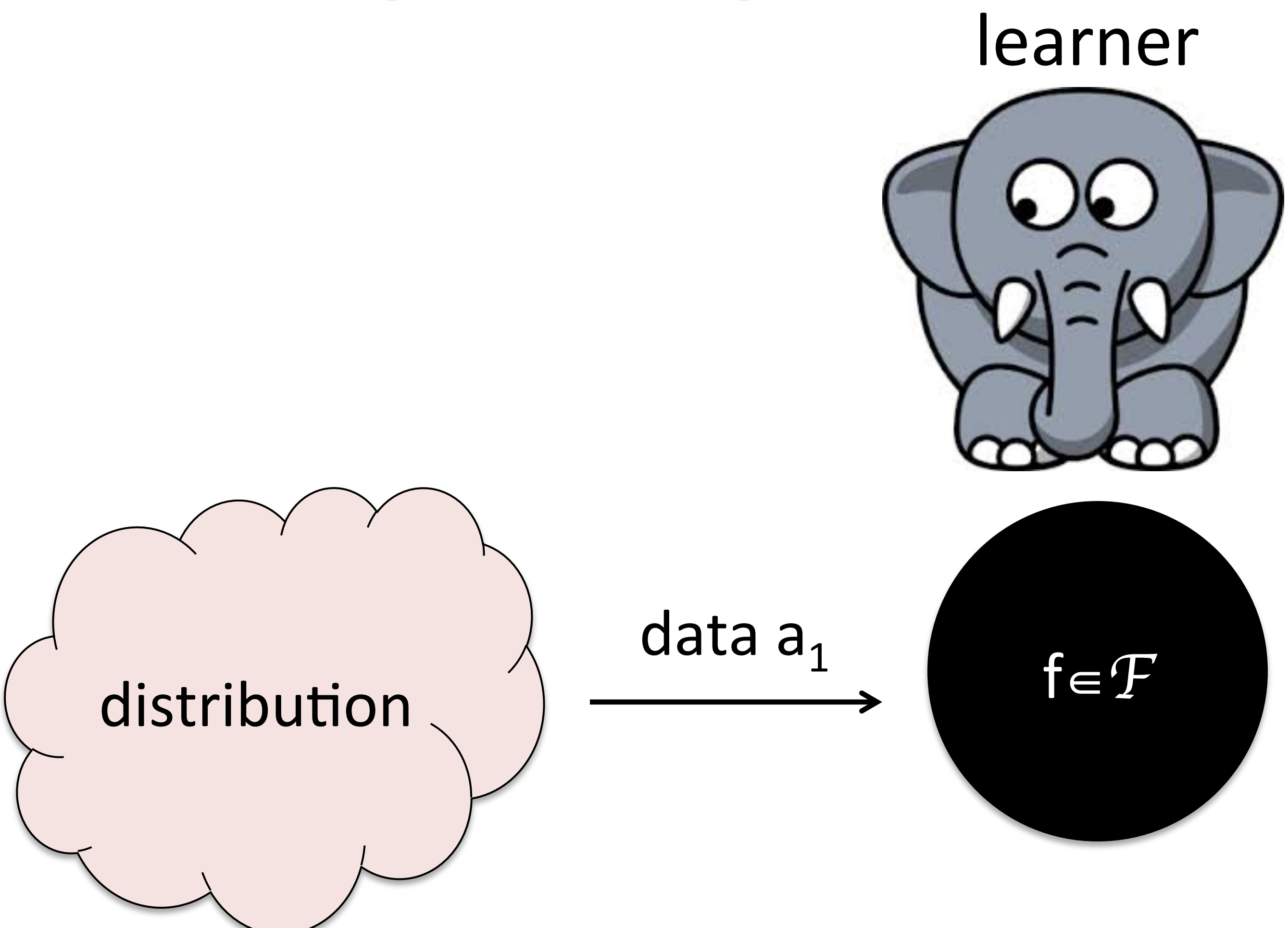
learner

distribution

$f \in \mathcal{F}$

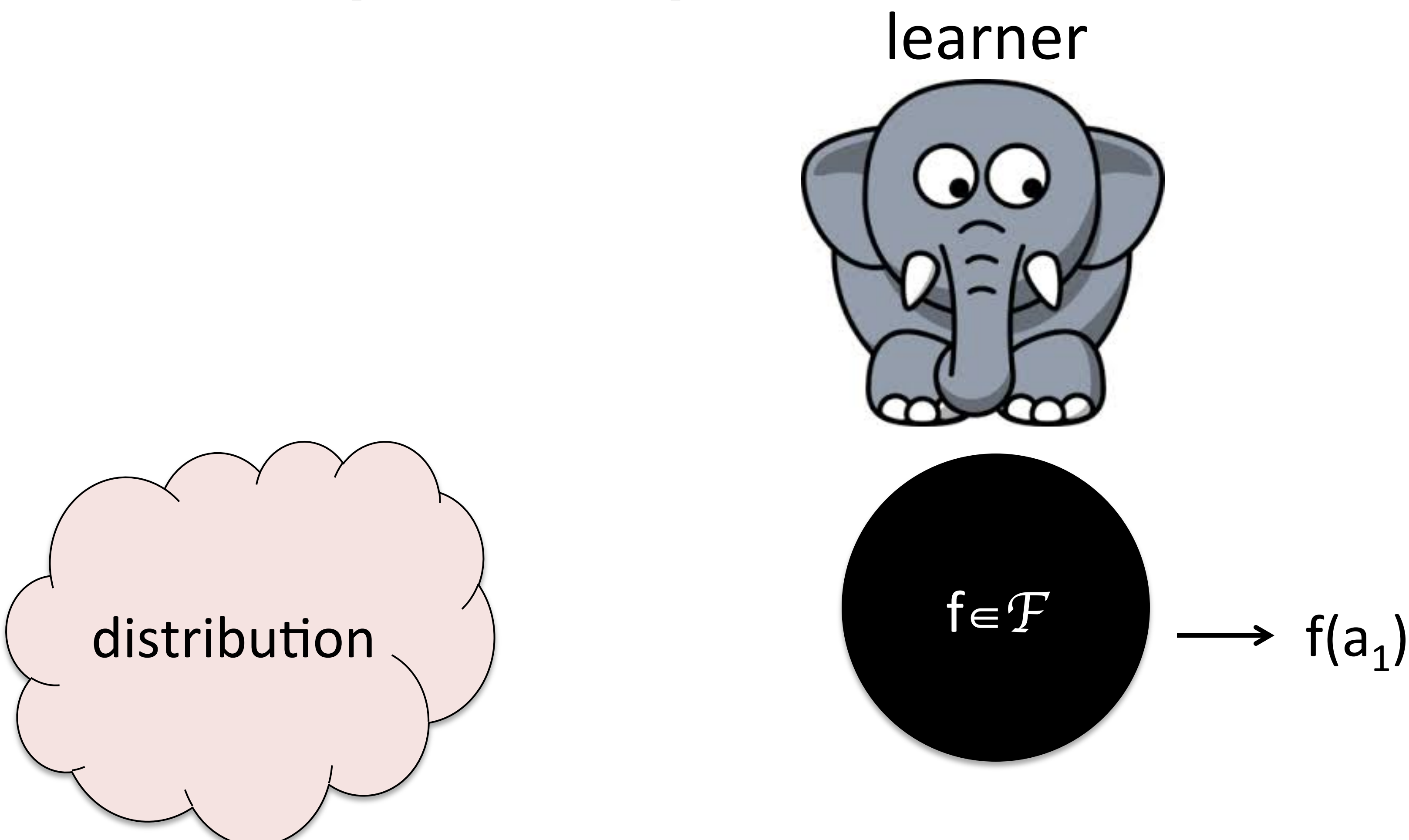
PAC Learning, in One Slide

[Valiant '84]



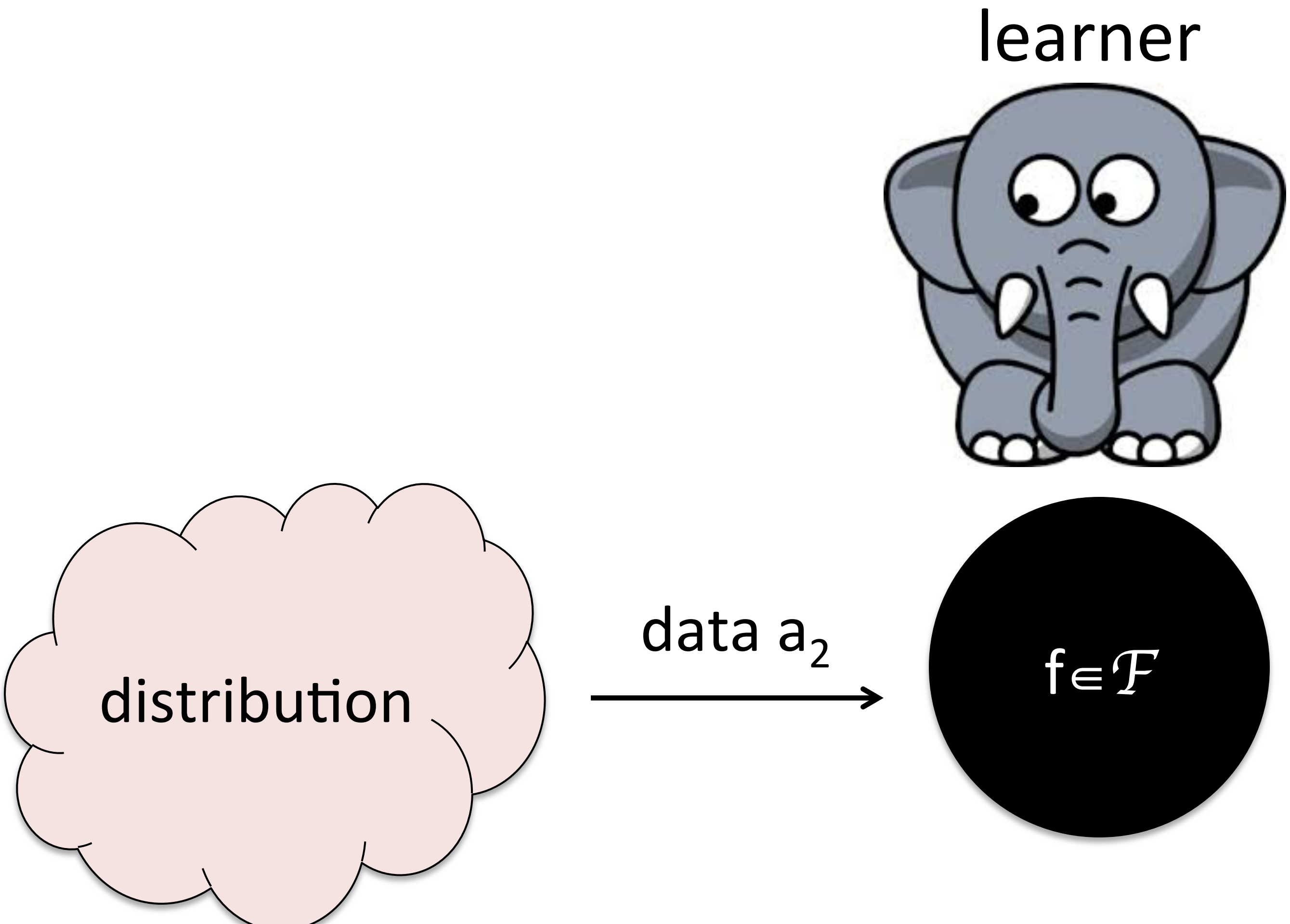
PAC Learning, in One Slide

[Valiant '84]



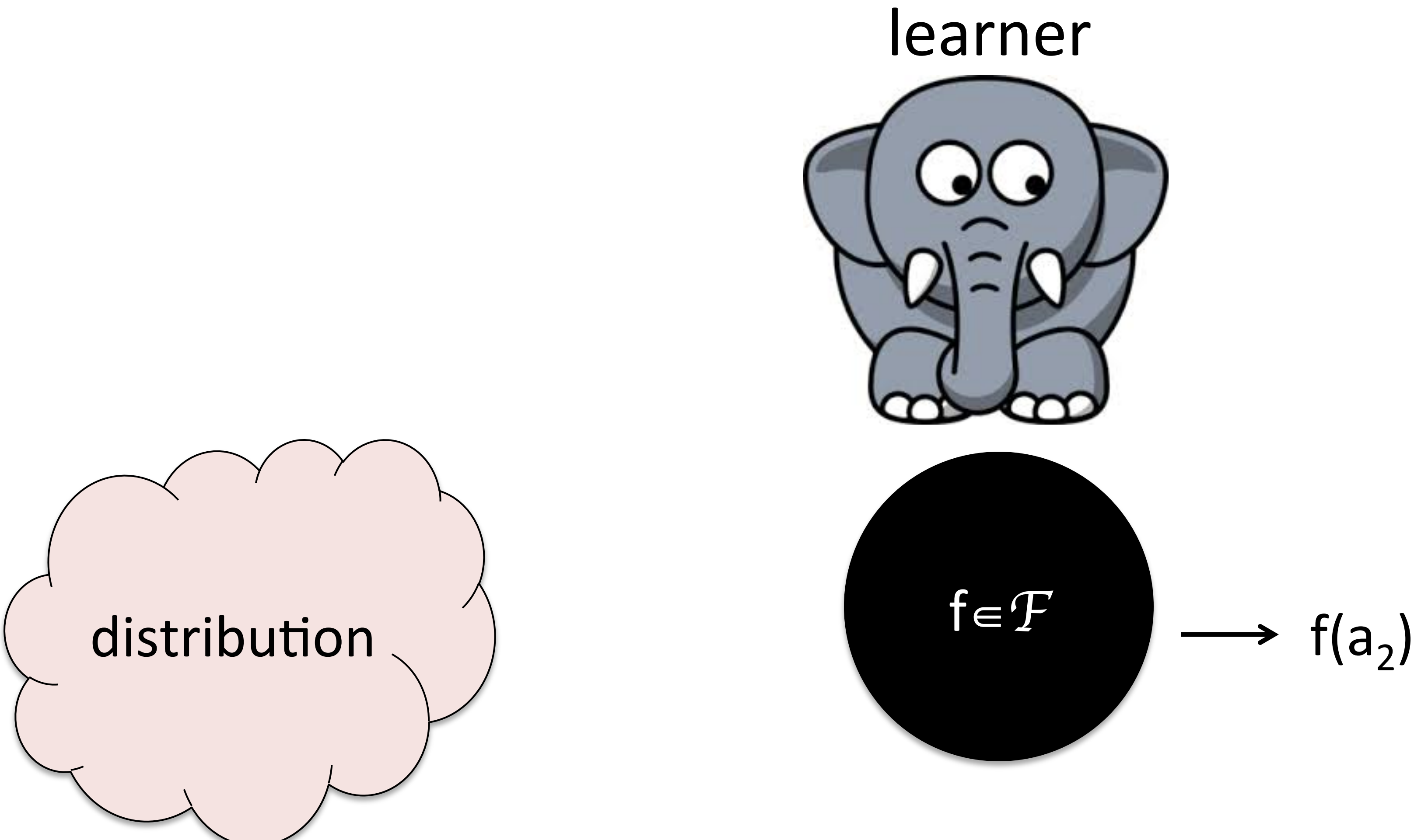
PAC Learning, in One Slide

[Valiant '84]



PAC Learning, in One Slide

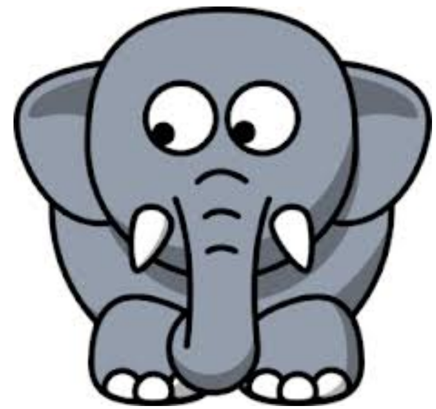
[Valiant '84]

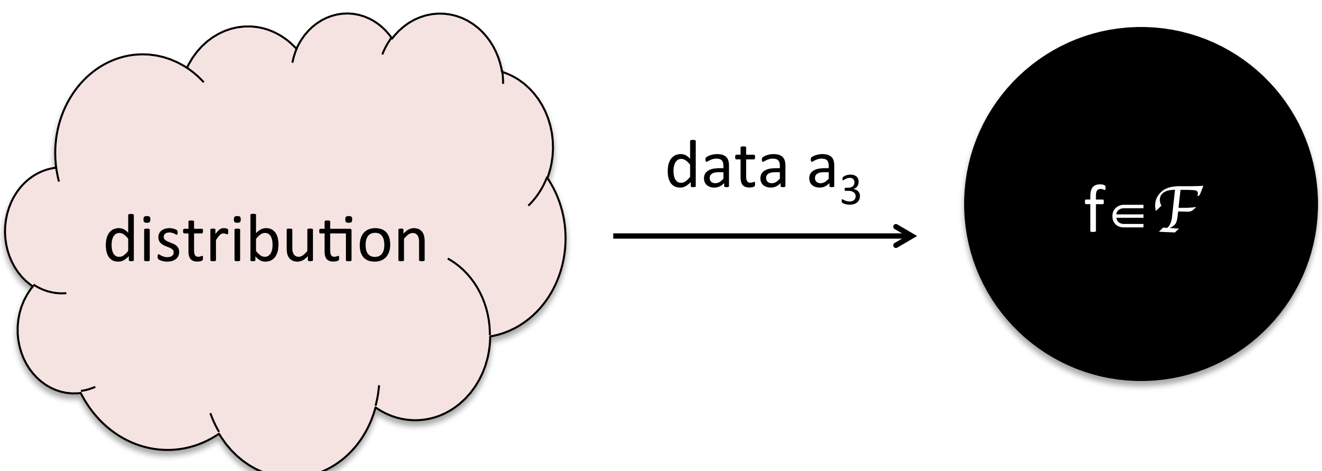


PAC Learning, in One Slide

[Valiant ’84]

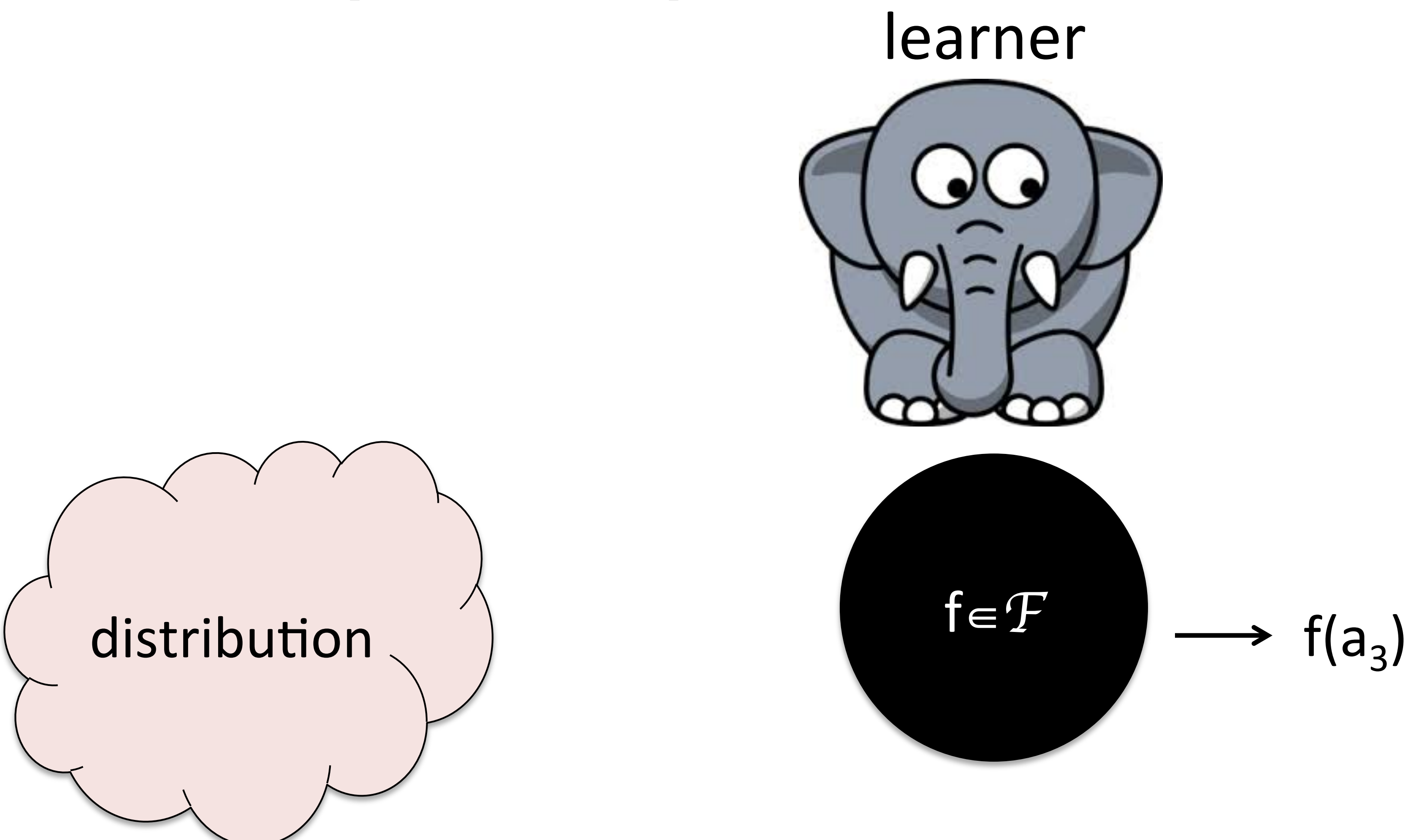
learner





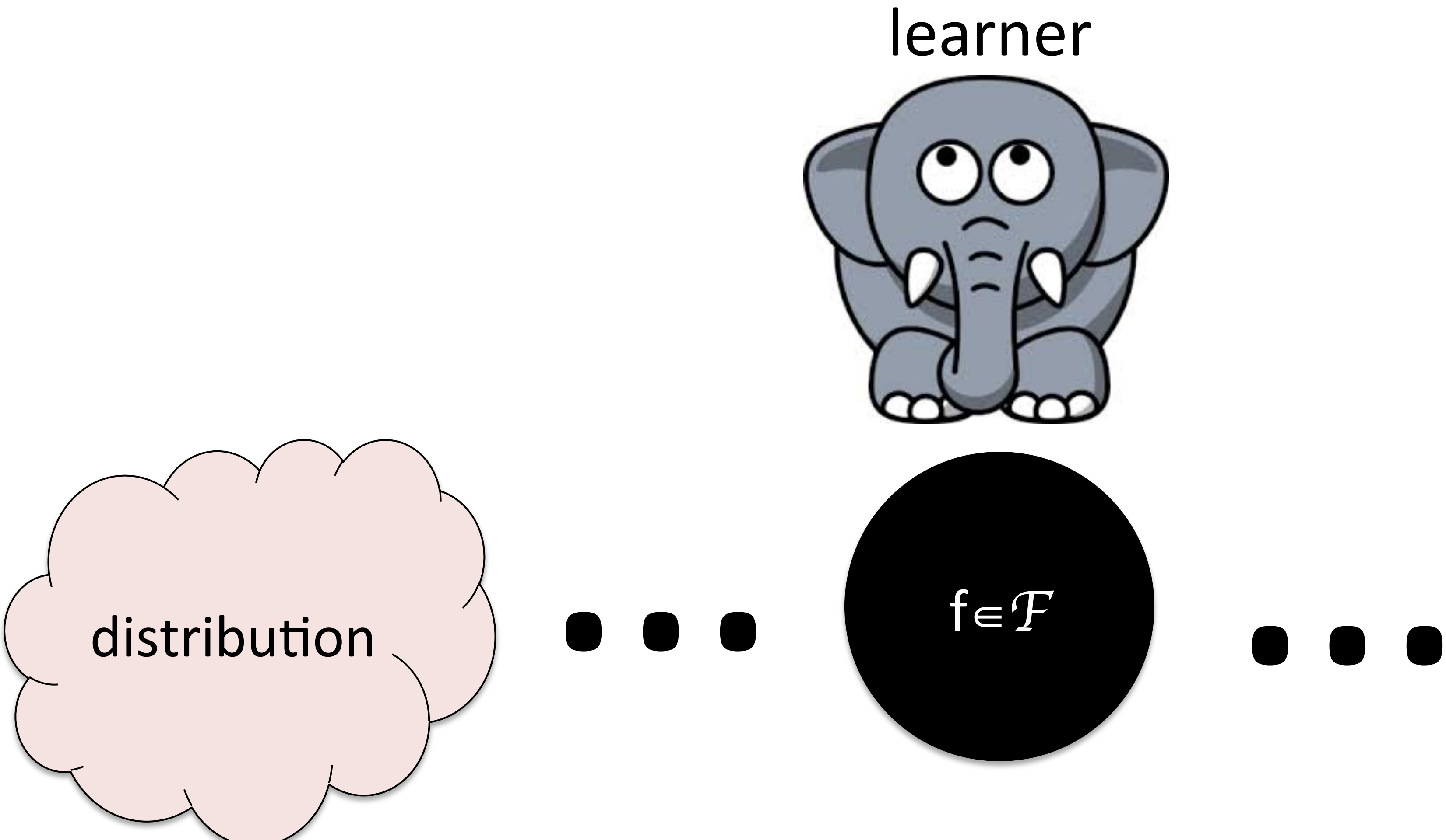
PAC Learning, in One Slide

[Valiant '84]



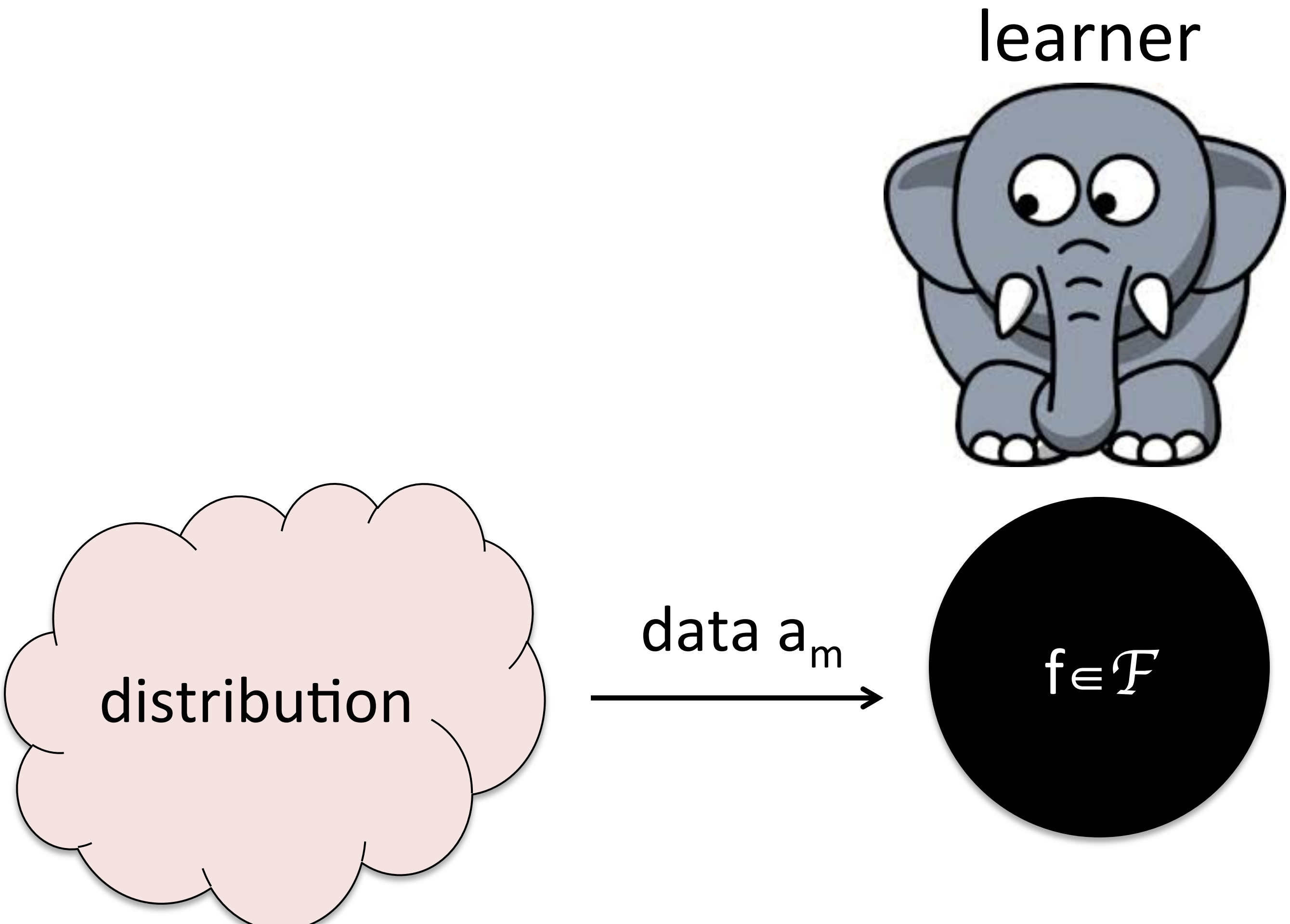
PAC Learning, in One Slide

[Valiant '84]



PAC Learning, in One Slide

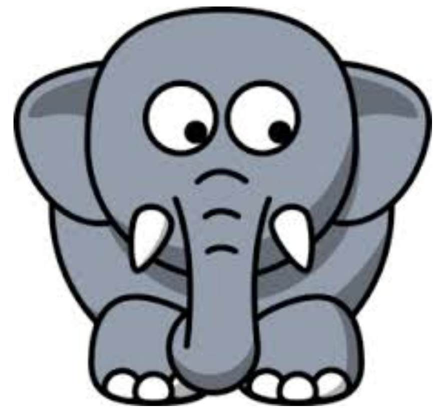
[Valiant '84]



PAC Learning, in One Slide

[Valiant '84]

learner



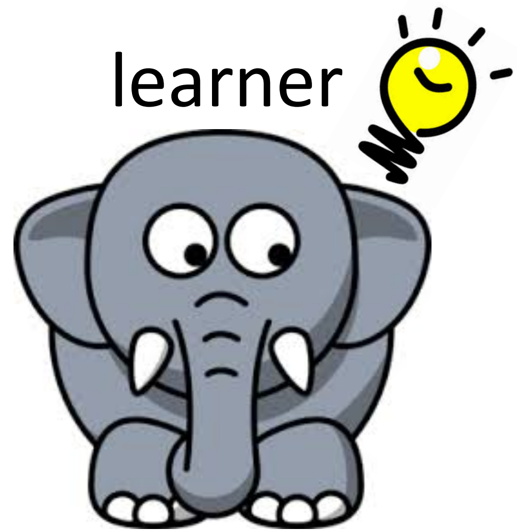
distribution

$f \in \mathcal{F}$ ⟶ $f(a_m)$

PAC Learning, in One Slide

[Valiant ’84]

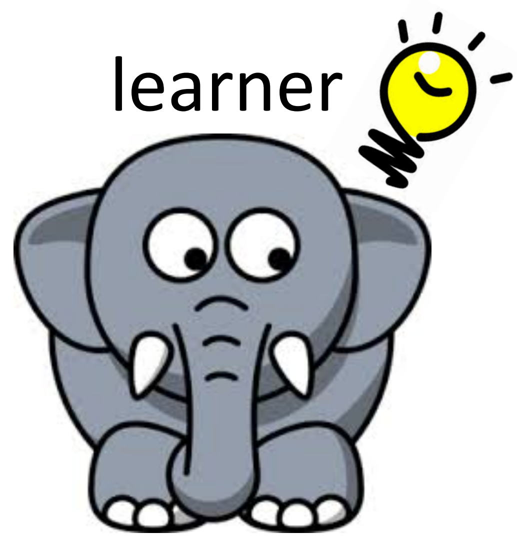
learner



PAC Learning, in One Slide

[Valiant '84]

learner

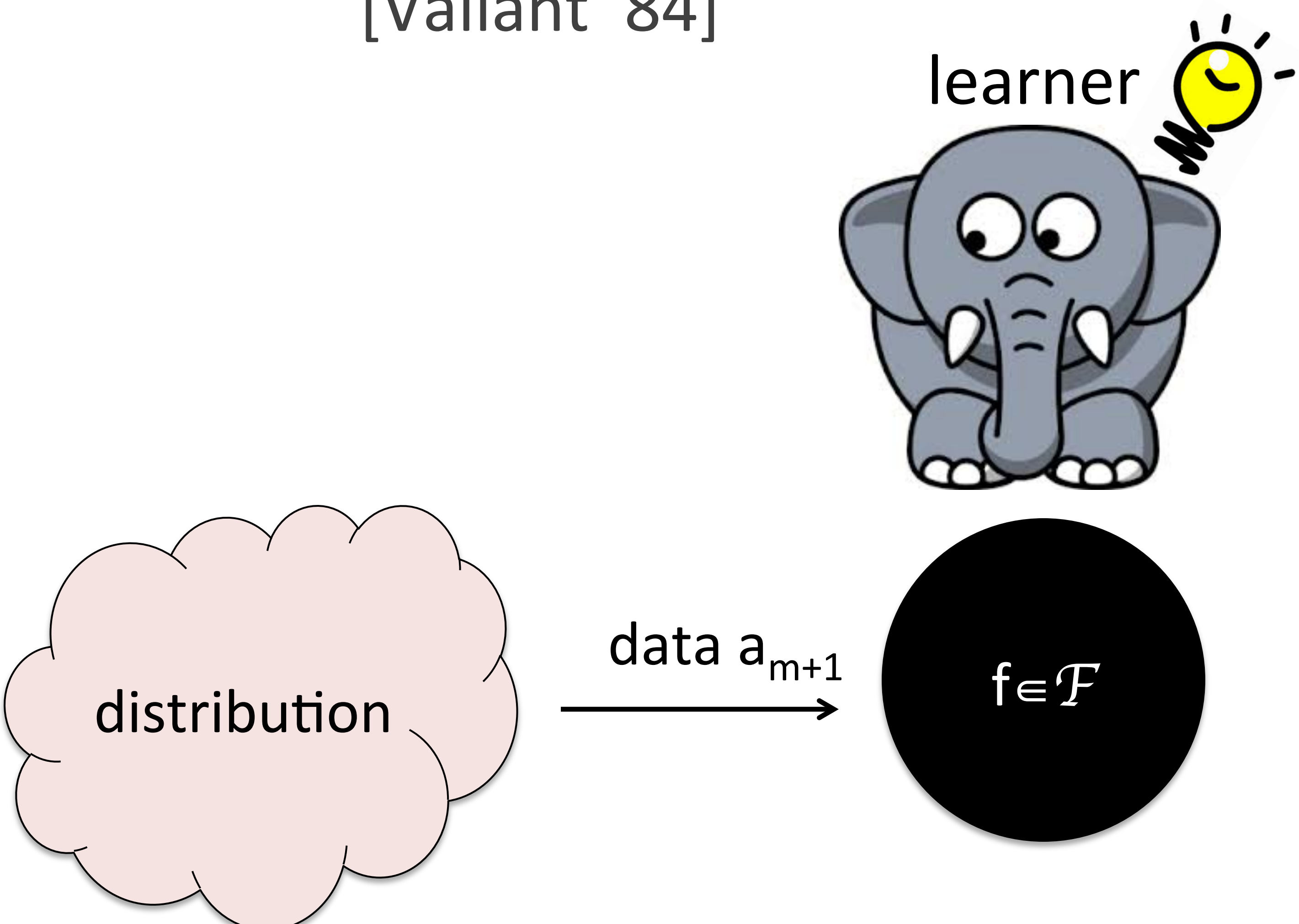


distribution

$f \in \mathcal{F}$ ⟶ $f(a_m)$

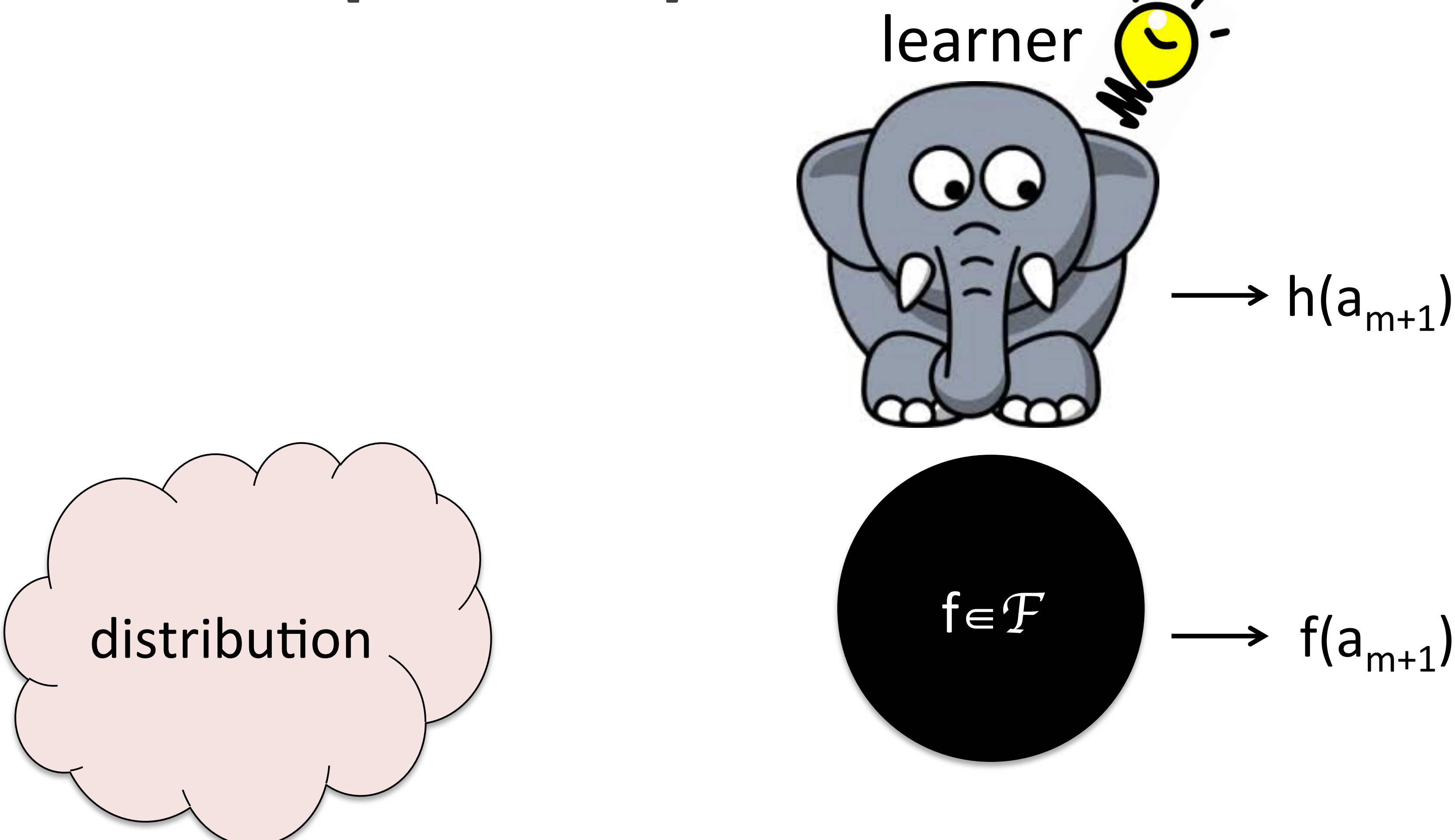
PAC Learning, in One Slide

[Valiant '84]



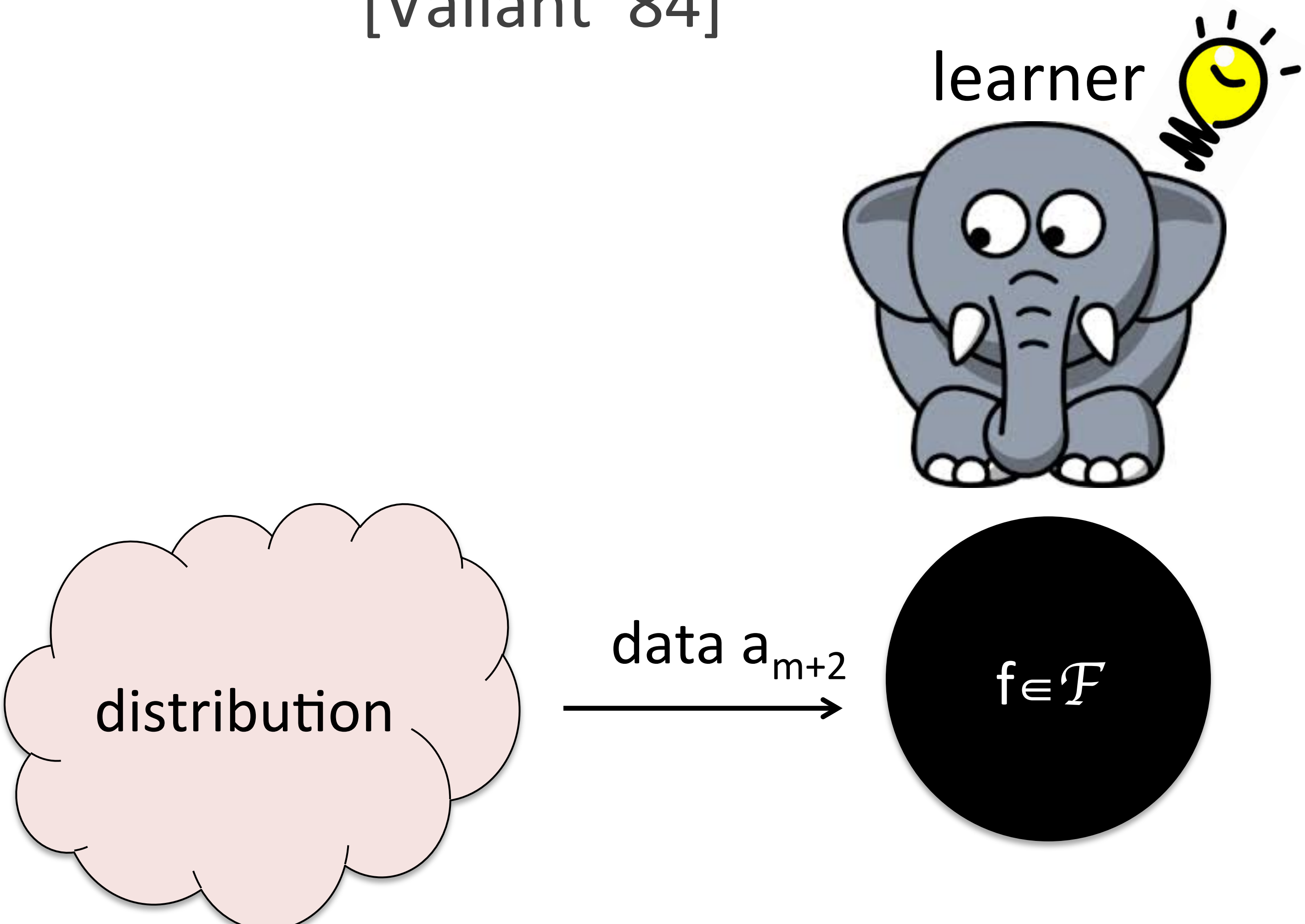
PAC Learning, in One Slide

[Valiant ’84]



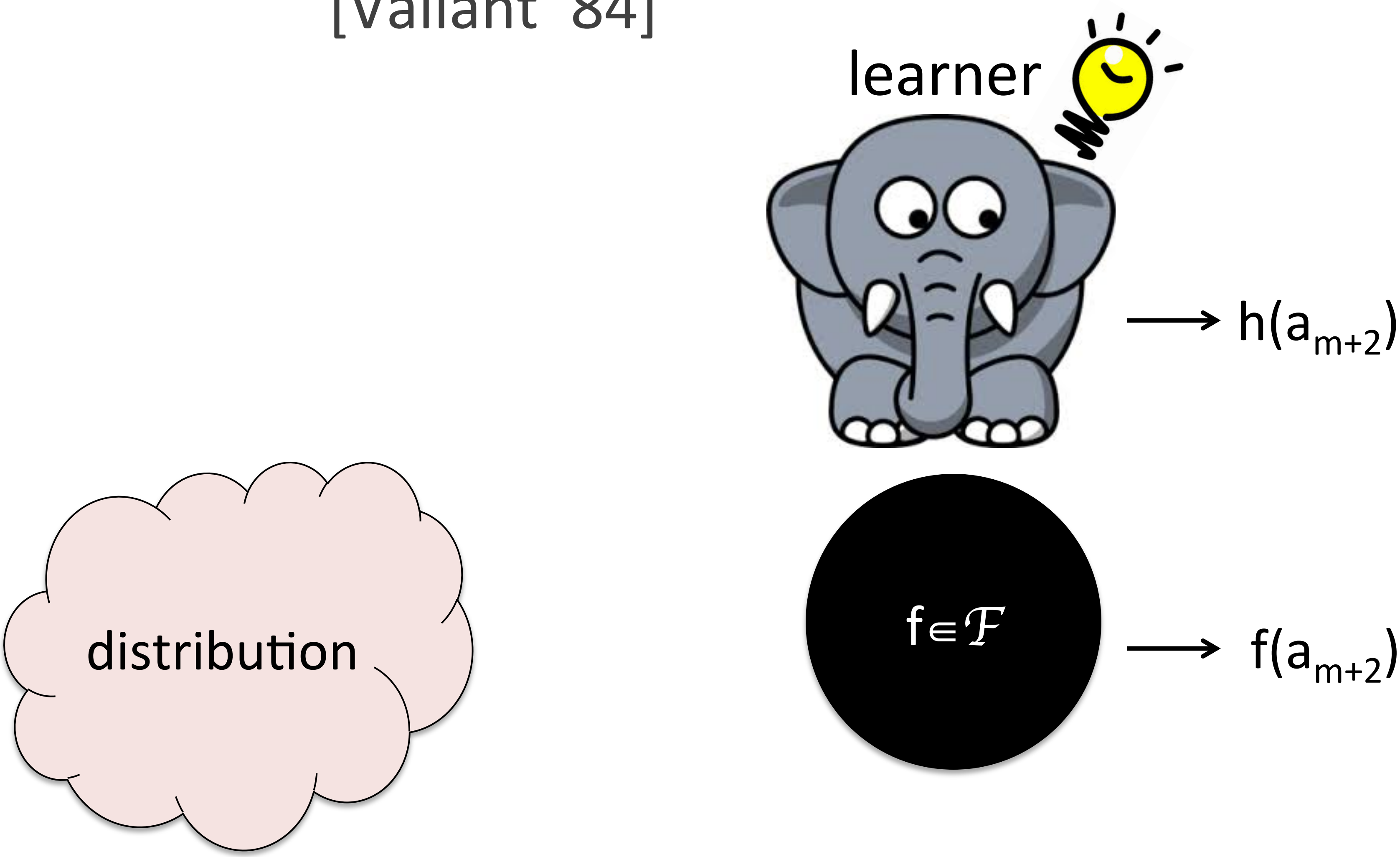
PAC Learning, in One Slide

[Valiant ’84]



PAC Learning, in One Slide

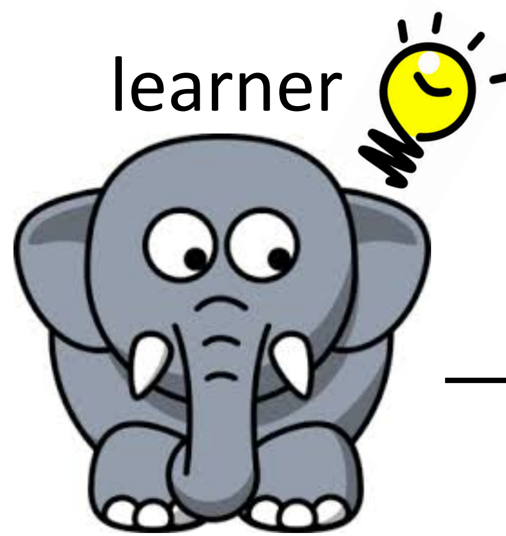
[Valiant ’84]



PAC Learning, in One Slide

[Valiant '84]

learner



→ $h(a_{m+2})$

≈

distribution

$f \in \mathcal{F}$ → $f(a_{m+2})$

Learning Linear Functions (mod 2)

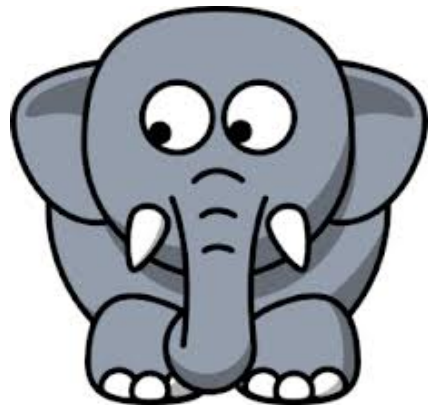
n variables

m equations

$$U \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \\ \\ \end{pmatrix}$$

m results

Learning Linear Functions (mod 2)



n variables

m equations

$$\begin{pmatrix} 0 & 1 & 0 \\ & & \\ & U & \\ & & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \\ \\ \\ \end{pmatrix}$$

m results

Learning Linear Functions (mod 2)

n variables

m equations

$$\begin{pmatrix} 0 & 1 & 0 \\ & & \\ & \mathrm{U} & \\ & & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ \\ \\ \\ \end{pmatrix}$$

m results

Learning Linear Functions (mod 2)

n variables

m equations

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ & \cup & \\ & & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ \\ \\ \end{pmatrix}$$

m results

Learning Linear Functions (mod 2)

n variables

m equations

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ & \cup & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \end{pmatrix}$$

m results

Learning Linear Functions (mod 2)

Gaussian elimination

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix}$$

n variables

m equations

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

m results

Learning Linear Functions (mod 2)

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \qquad \begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix}$$

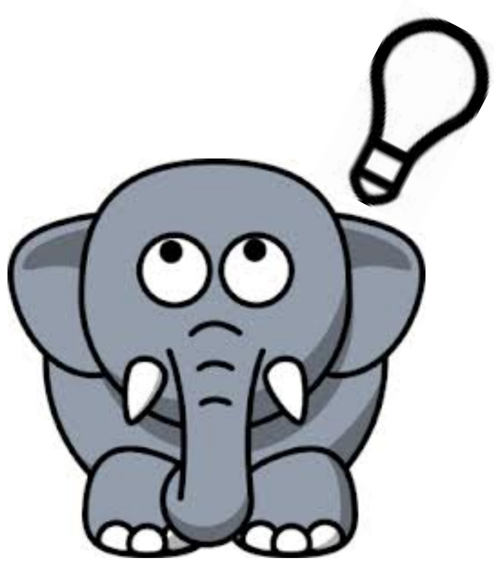
target f hypothesis h

Remember the coefficients of the equations are generated uniformly at random from $\{0,1\}^n$.

So, if $\exists$ i s.t. $x_i \neq x'_i$, then f and h will disagree ½ of the time. Hence, 2^n different orthogonal functions.

form the Fourier basis in DFA

When there's noise…



n variables

m equations

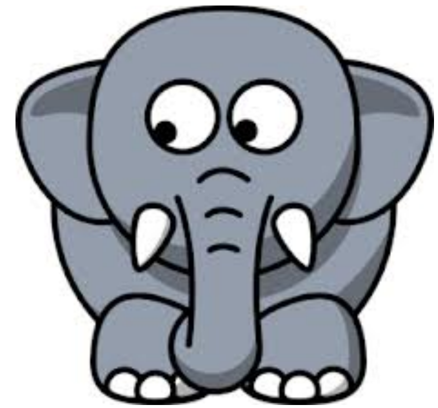
$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \cancel{1}\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

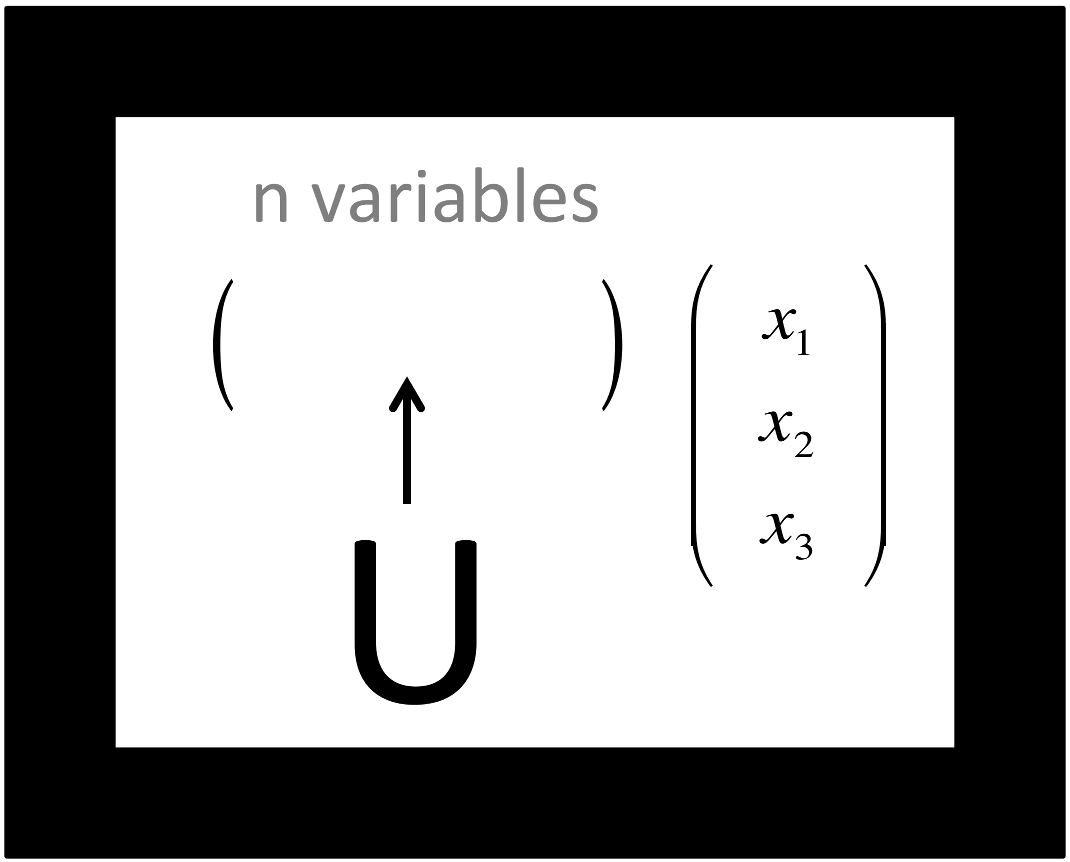
m results

statistical queries

Statistical Query Learning

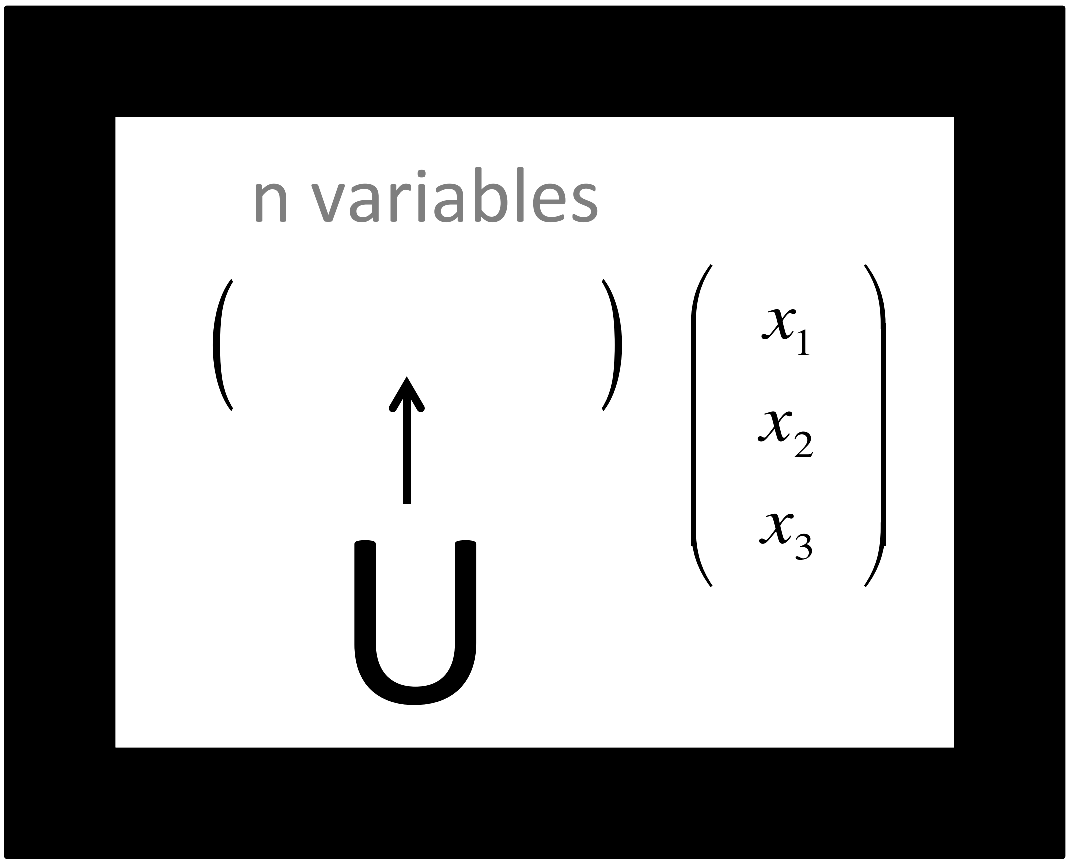
[Kearns ’93]

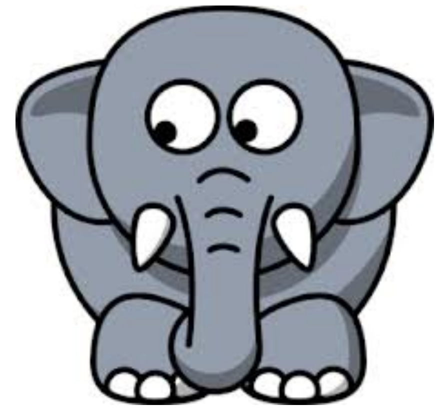




Statistical Query Learning

[Kearns '93]





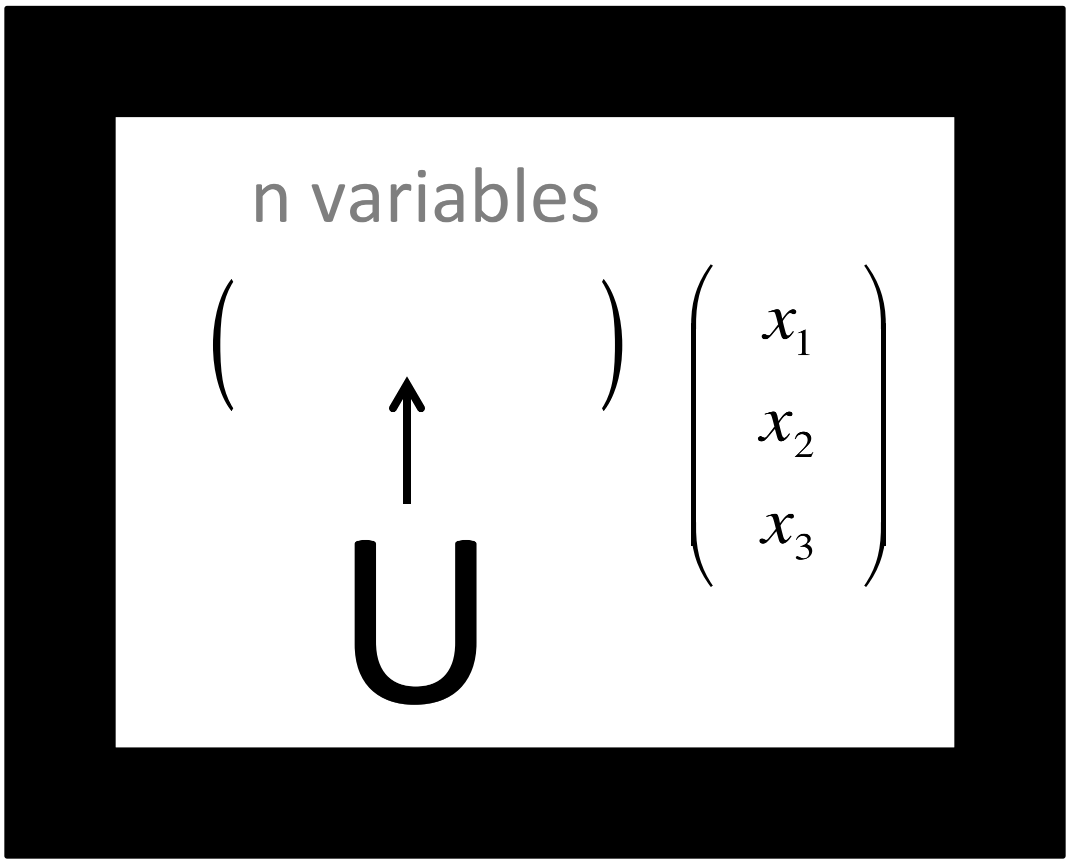
q(a, f(a))

q: $\{1,0\}^n \times \{0,1\} \rightarrow \{0,1\}$

and sample size S

Statistical Query Learning

[Kearns ’93]



$q(a, f(a))$

$q: \{1,0\}^n \times \{0,1\} \rightarrow \{0,1\}$

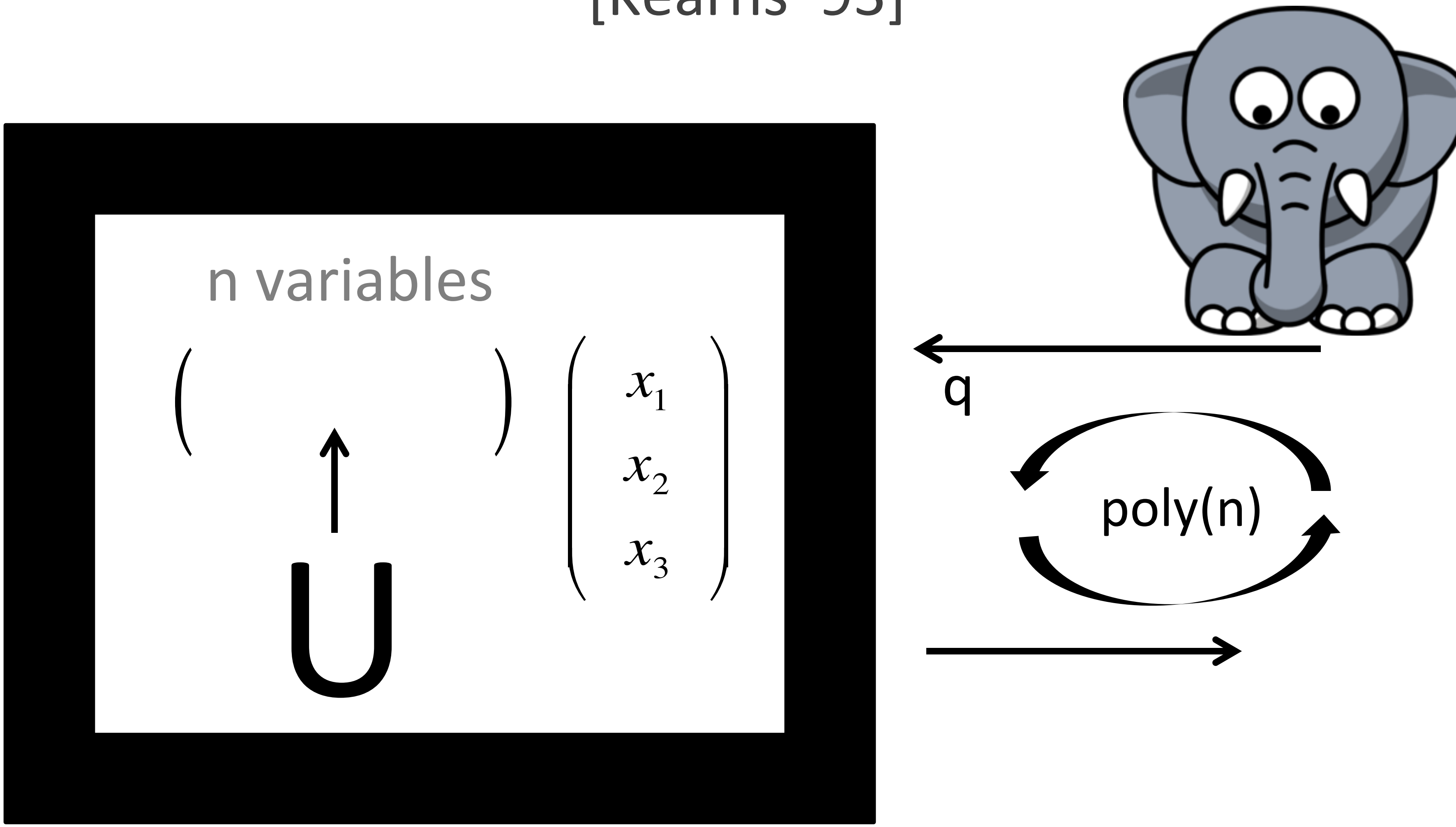
and sample size S

something like

$E_U[q(a, f(a))] \pm 1/S^{1/2}$

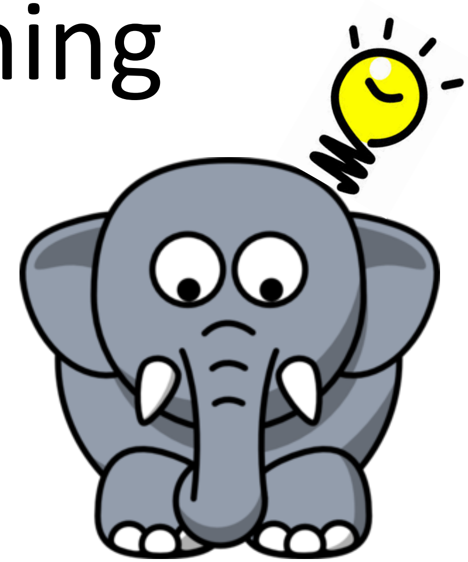
Statistical Query Learning

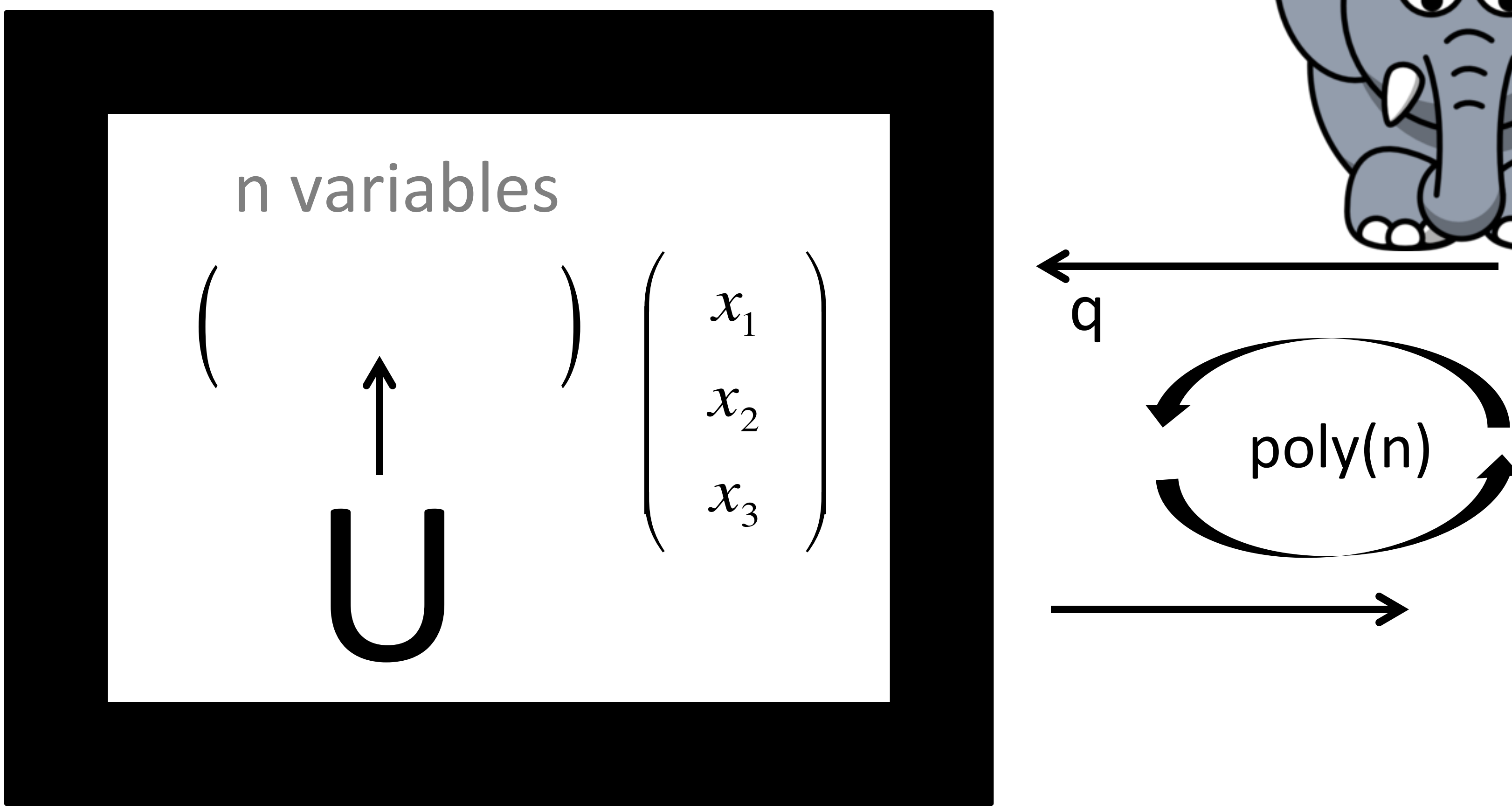
[Kearns '93]



Statistical Query Learning

[Kearns '93]





Statistical Queries

- Theorem [Kearns '93]: If a family of functions is learnable with statistical queries, then it is learnable (in the original model) with noise!

- Theorem [Kearns '93]: Linear functions (mod 2) are not learnable with statistical queries.

  proof idea: *b/c the linear functions are orthogonal under U, queries are either uninformative or "eliminate" one wrong linear function at a time (and there are 2^n)*

Statistical Queries

- Theorem [Kearns '93]: If a family of functions is learnable with statistical queries, then it is learnable (in the original model) with noise!
- Theorem [Kearns '93]: Linear functions (mod 2) are not learnable with statistical queries.
- Theorem [Blum et al '94], when a family of functions has exponentially high "SQ dim" it is not learnable with statistical queries.
  - SQ dim is roughly the number of nearly-orthogonal functions (wrt a reference distribution). Linear functions have SQ dimension = 2^n.

Statistical Queries

- Theorem [Kearns '93]: If a family of functions is learnable with statistical queries, then it is learnable (in the original model) with noise!
- Theorem [Kearns '93]: Linear functions (mod 2) are not learnable with statistical queries.
- Theorem [Blum et al '94], when a family of functions has exponentially high "SQ dim" it is not learnable with statistical queries.
- Shockingly, almost all learning algorithms can be implemented w/ statistical queries! So high SQ dim is a serious barrier to learning, especially under noise.

Summary

- Linear equations with errors seem hard to solve (Noisy parity functions seem hard to "learn")
- Statistical queries and statistical dimension from learning theory are an explanation as to why.

  (almost all our learning algorithms are statistical)

Summary

- Linear equations with errors seem hard to solve (Noisy parity functions seem hard to “learn”)

- Statistical queries and statistical dimension from learning theory are an explanation as to why.

  (almost all our learning algorithms are statistical)

Idea: extend this framework to optimization problems and use it to explain the hardness of planted clique!

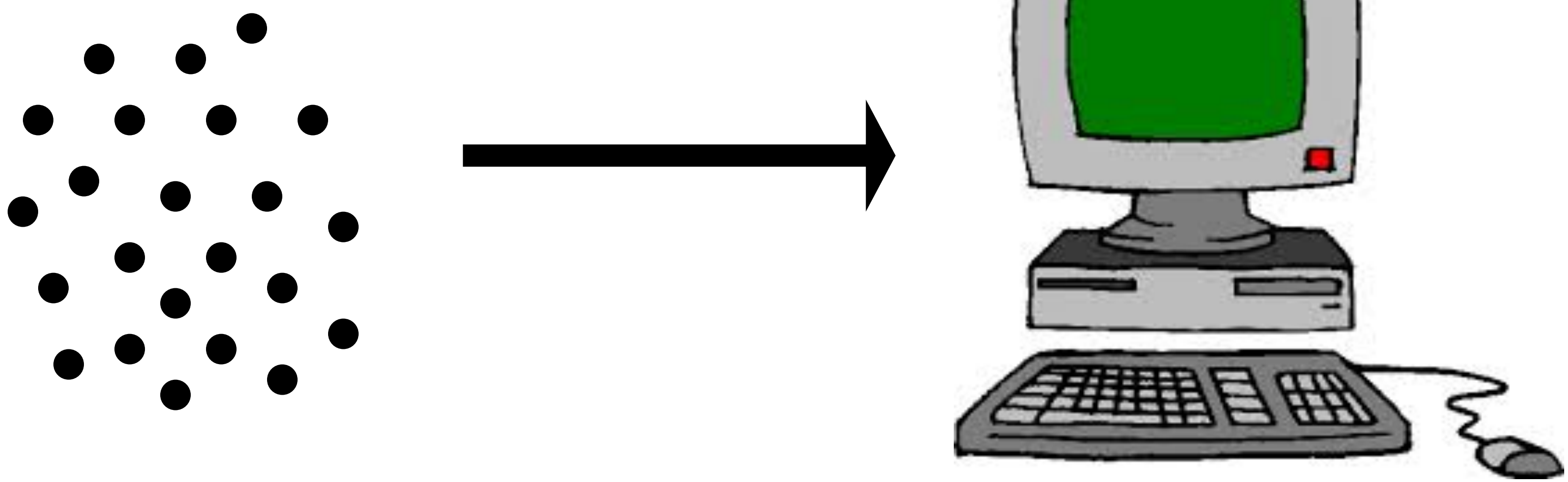
statistical algorithms

[FGRVX ’13]

Traditional Algorithms

input data

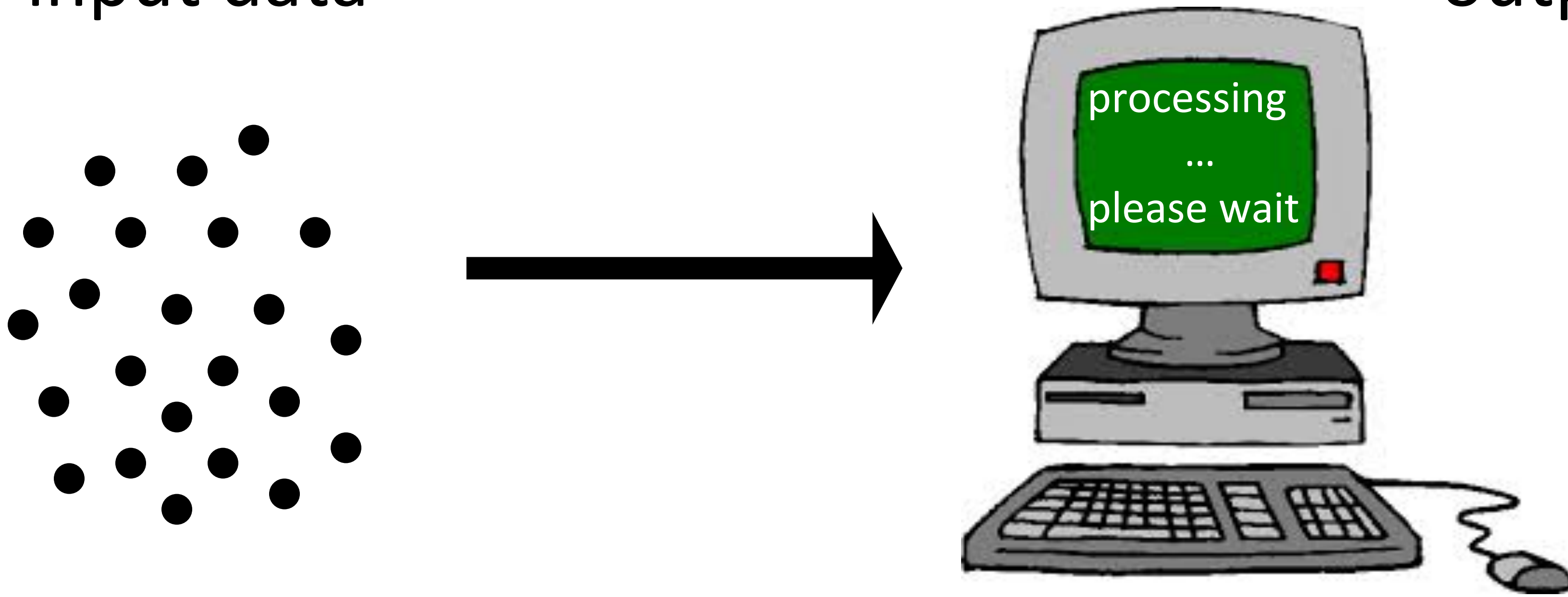
output



Traditional Algorithms

input data

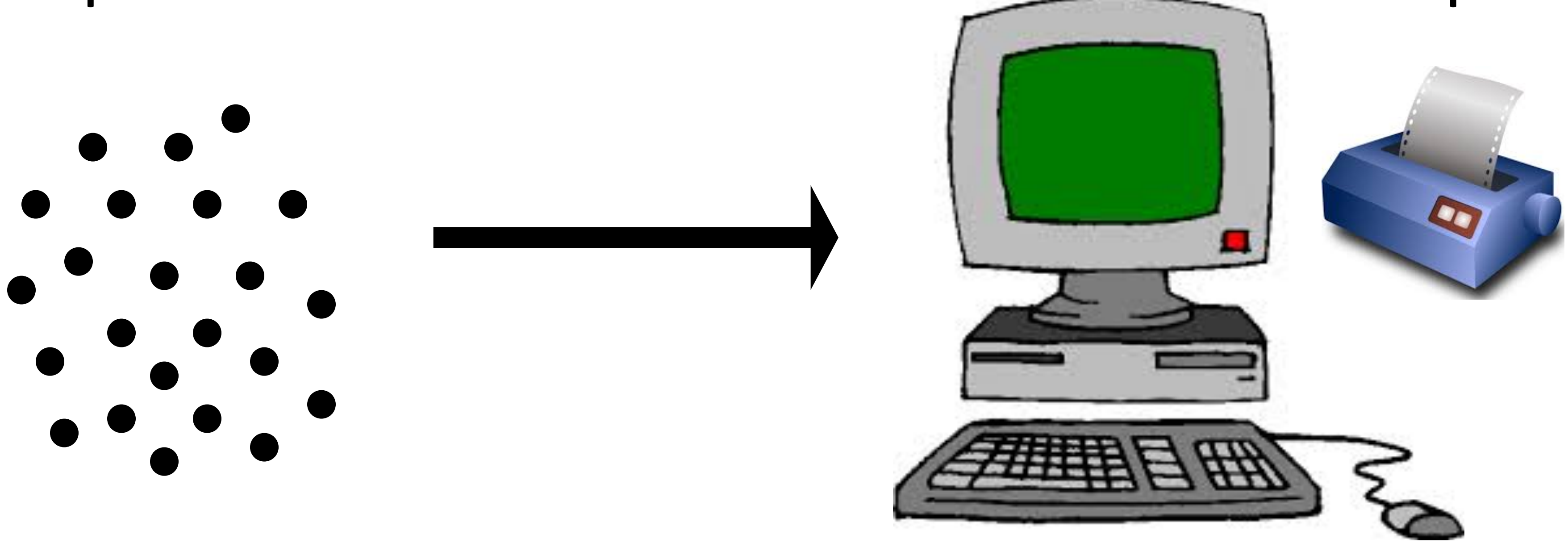
output



Traditional Algorithms

input data

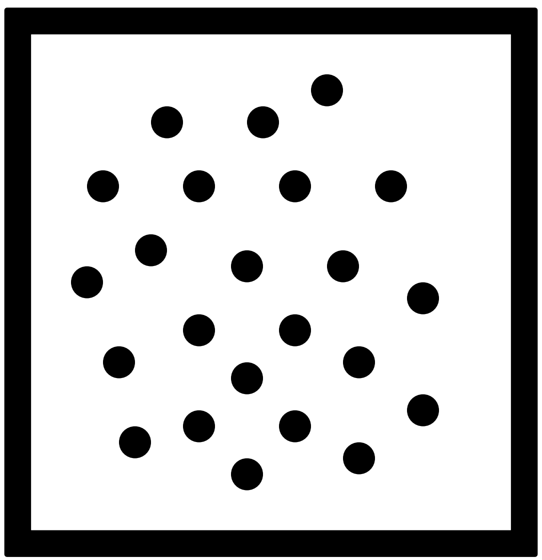
output



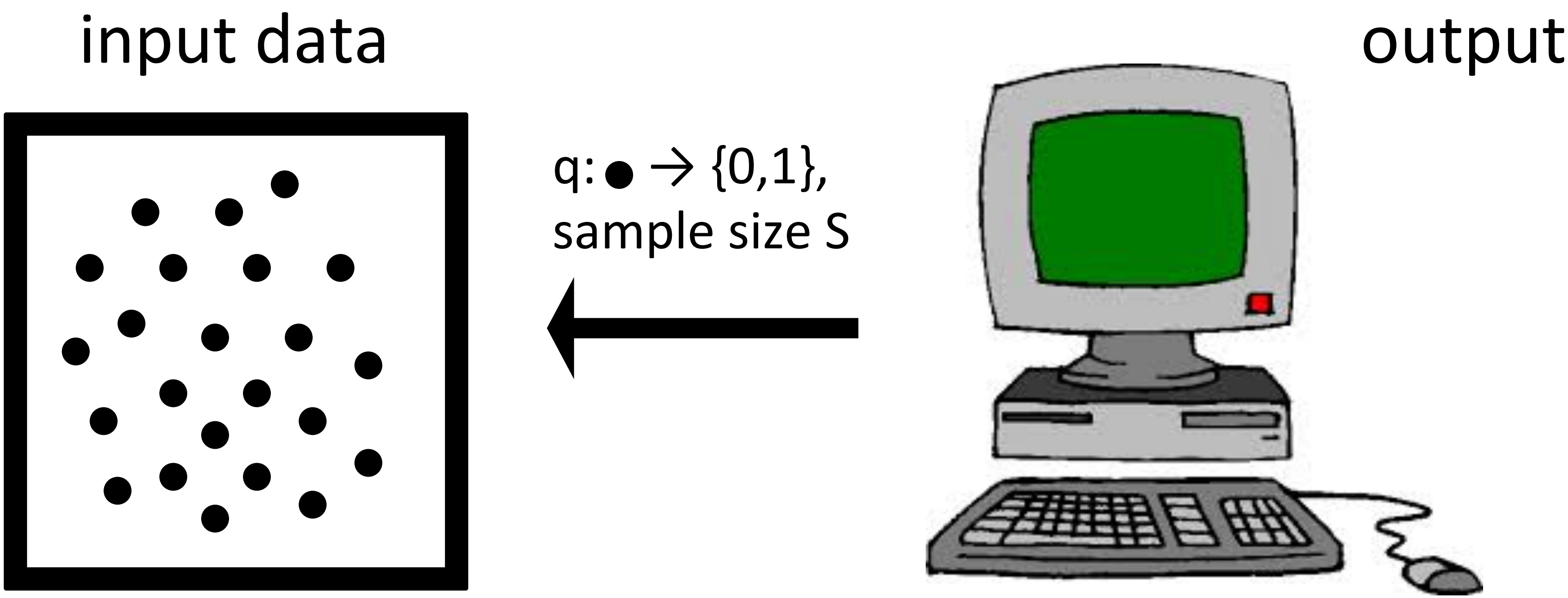
Statistical Algorithms

input data

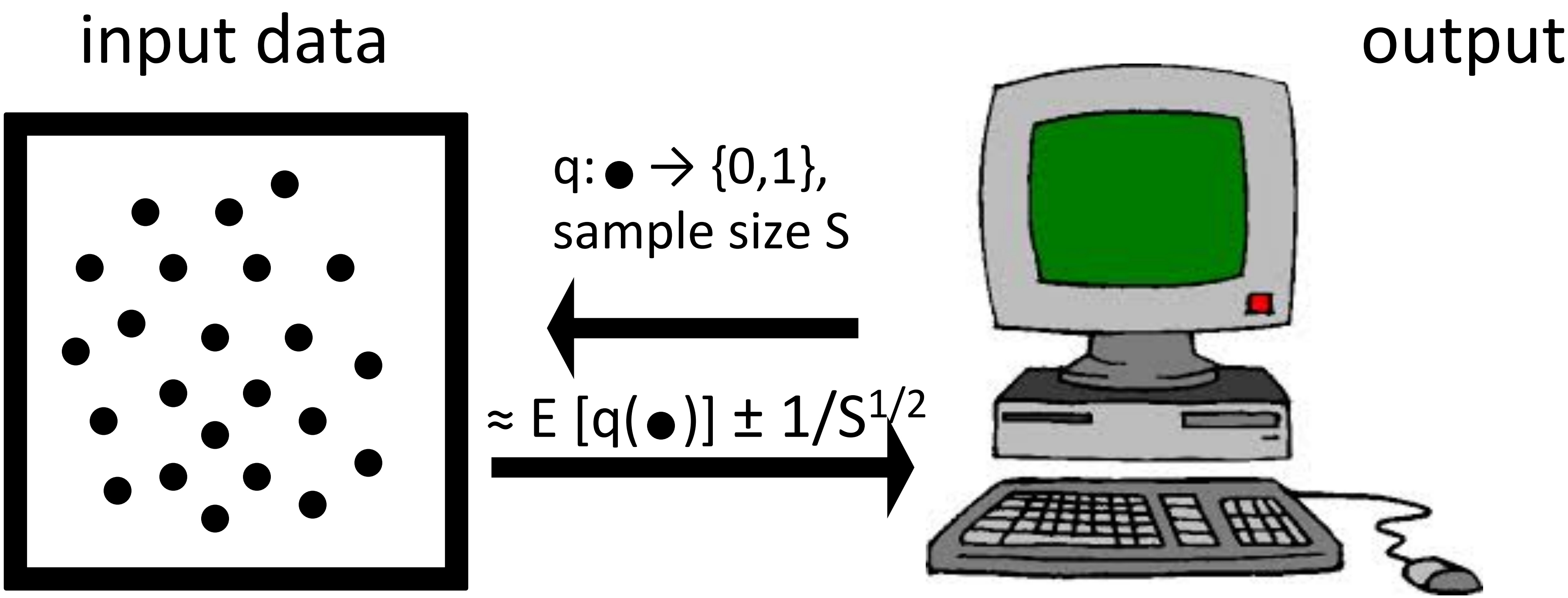
output



Statistical Algorithms



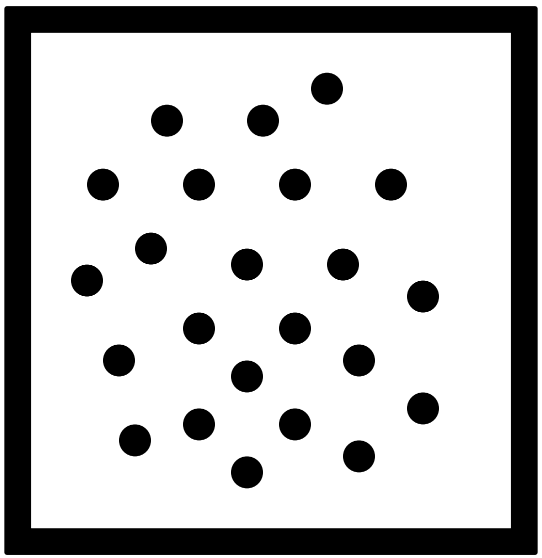
Statistical Algorithms

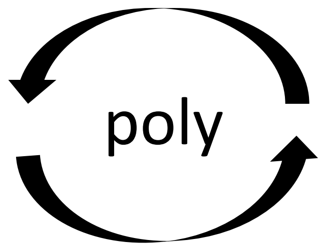


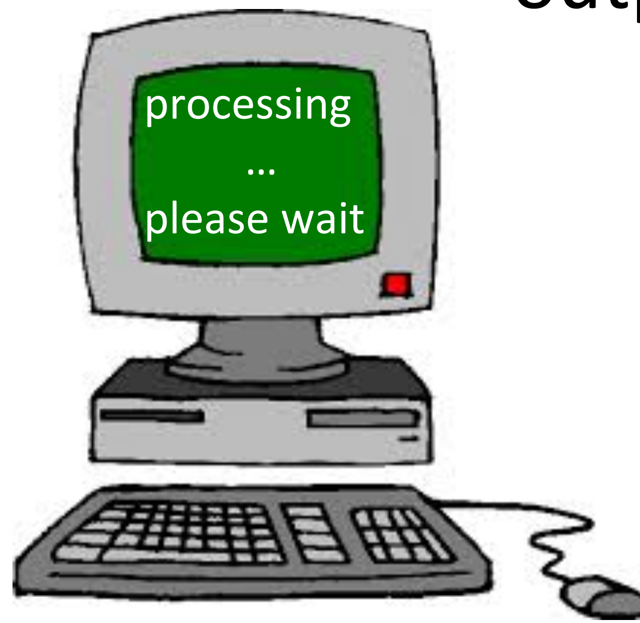
Statistical Algorithms

input data

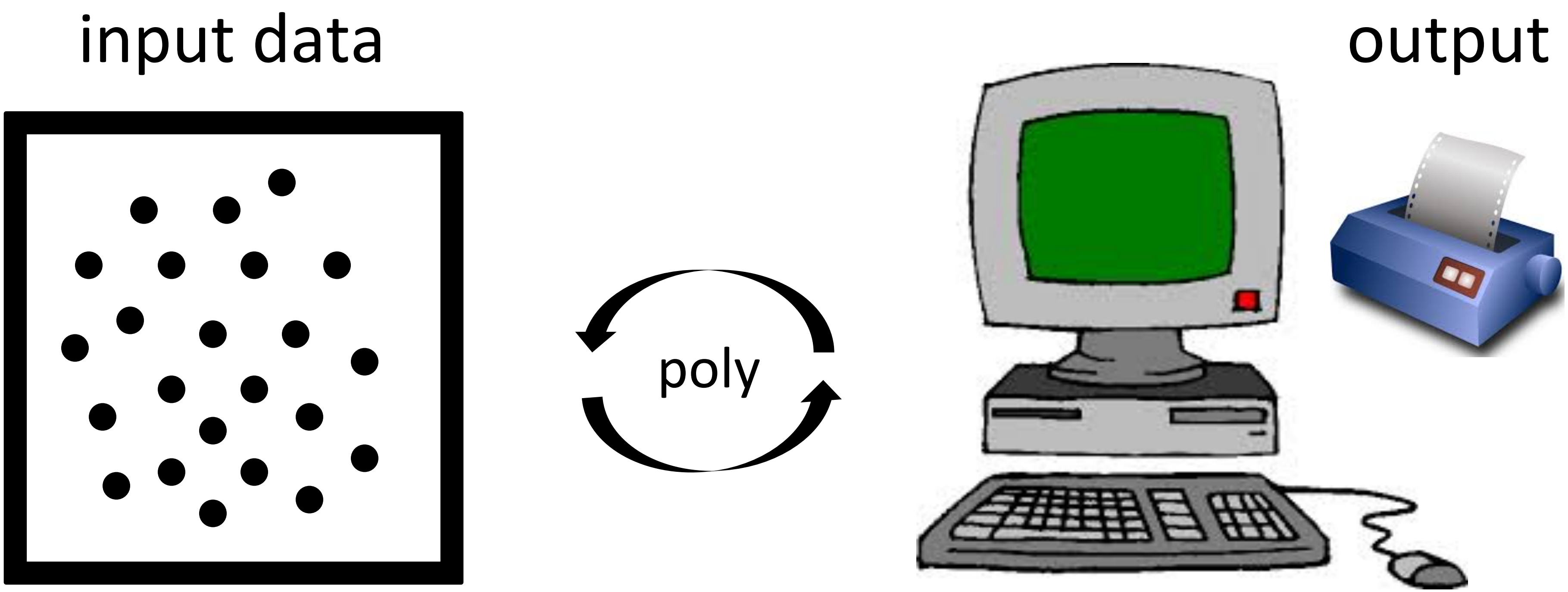
output



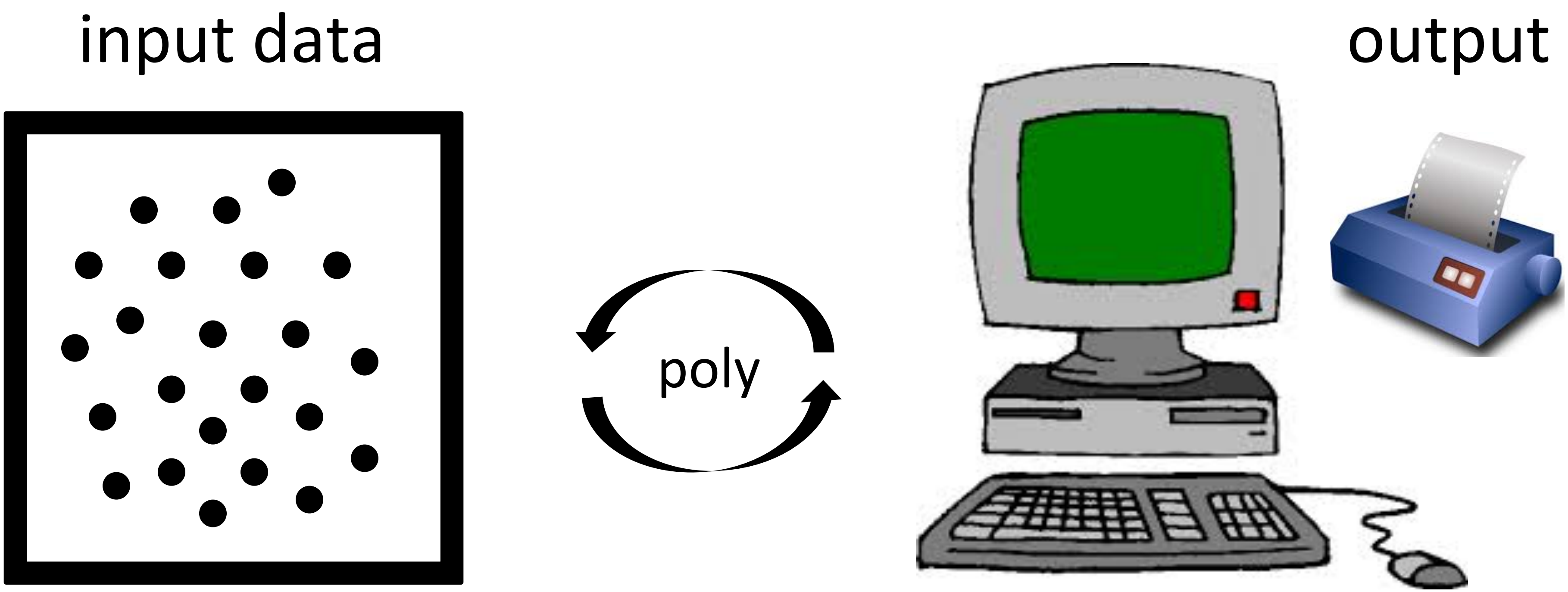




Statistical Algorithms

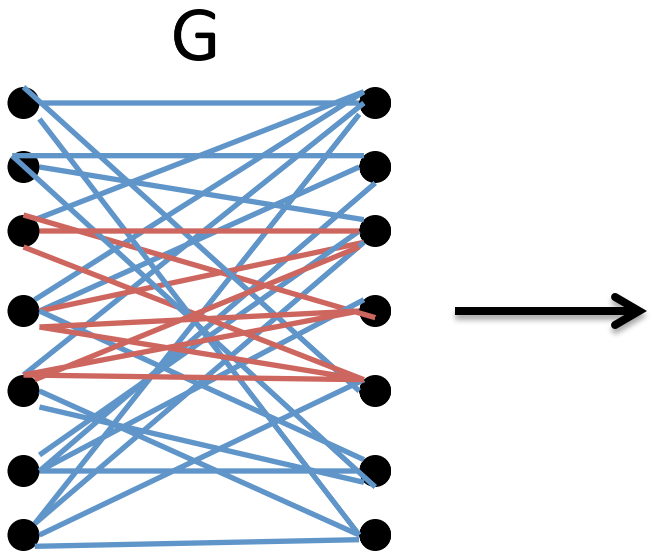


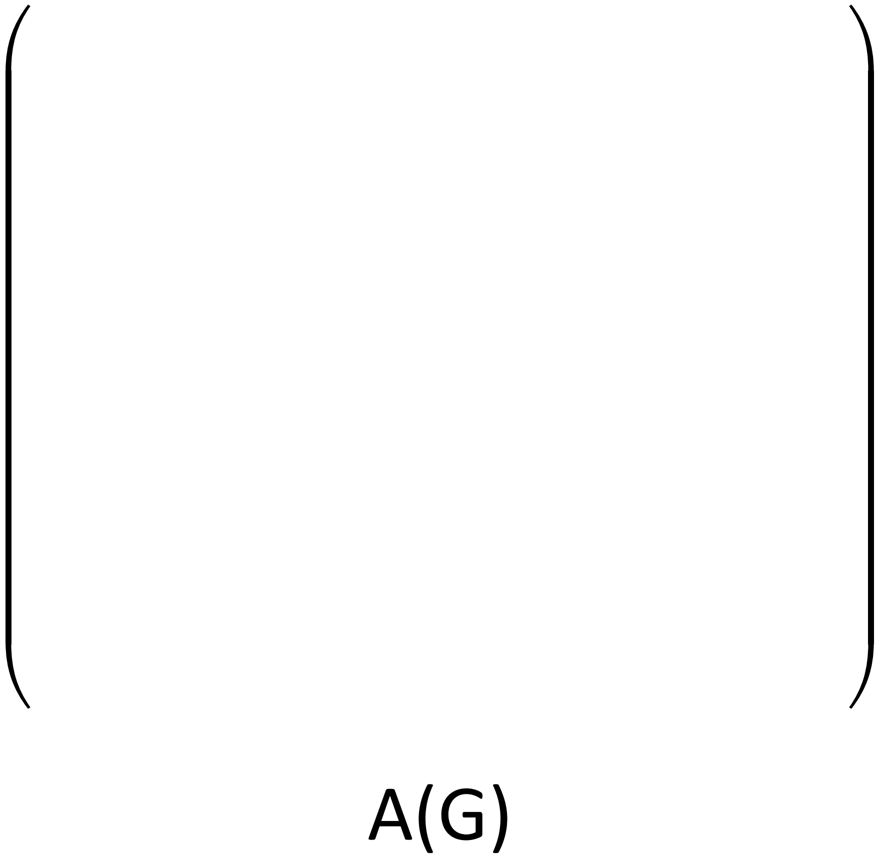
Statistical Algorithms



Turns out most (all?) current optimization algorithms have statistical analogues!

Bipartite Planted Clique

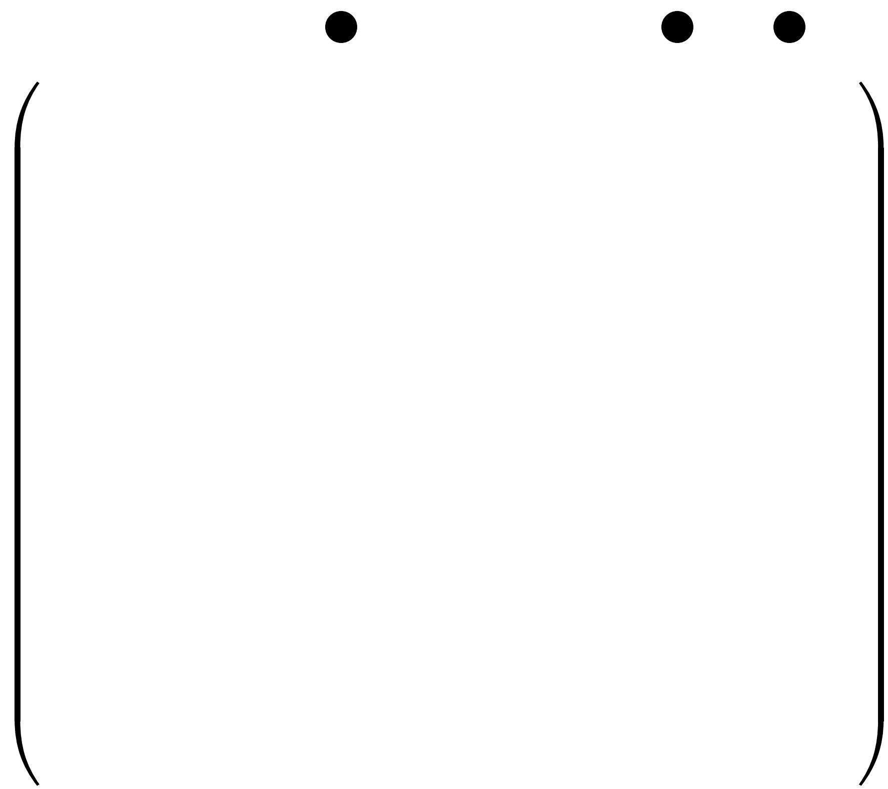




Bipartite Planted Clique

<u>each row</u>

w.p. (n-k)/n	is random
w.p. k/n	is random, except in “plant” coordinates



A(G)

Bipartite Planted Clique

<u>each row</u>

w.p. (n-k)/n	is random
w.p. k/n	is random, except in “plant” coordinates

			●			●	●
●	1	1	1	0	0	1	1
●	0	1	1	1	1	0	1
●	1	0	1	1	0	1	1
●	0	0	0	1	1	0	0
●	0	0	1	0	1	0	1
●	0	1	0	1	1	1	1
●	0	1	1	1	0	1	1

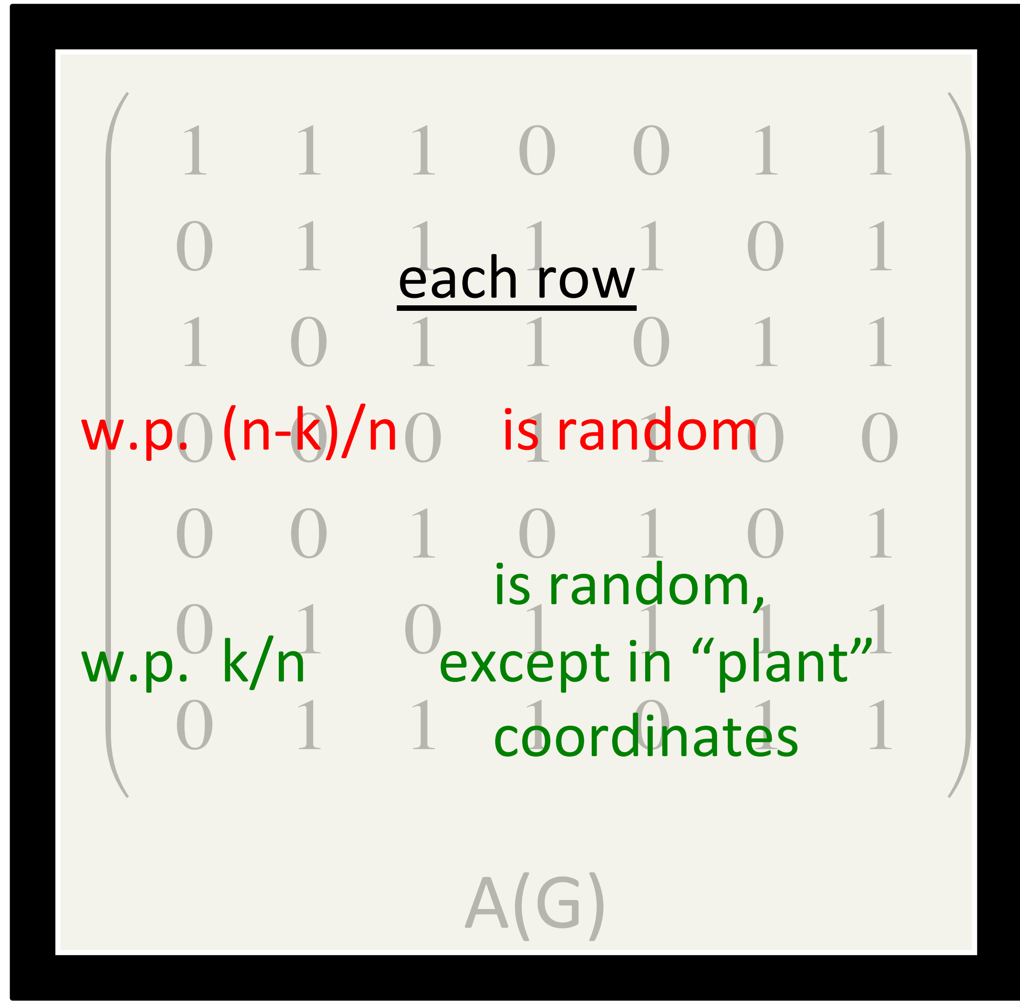
A(G)

Statistical Algorithms for BPC

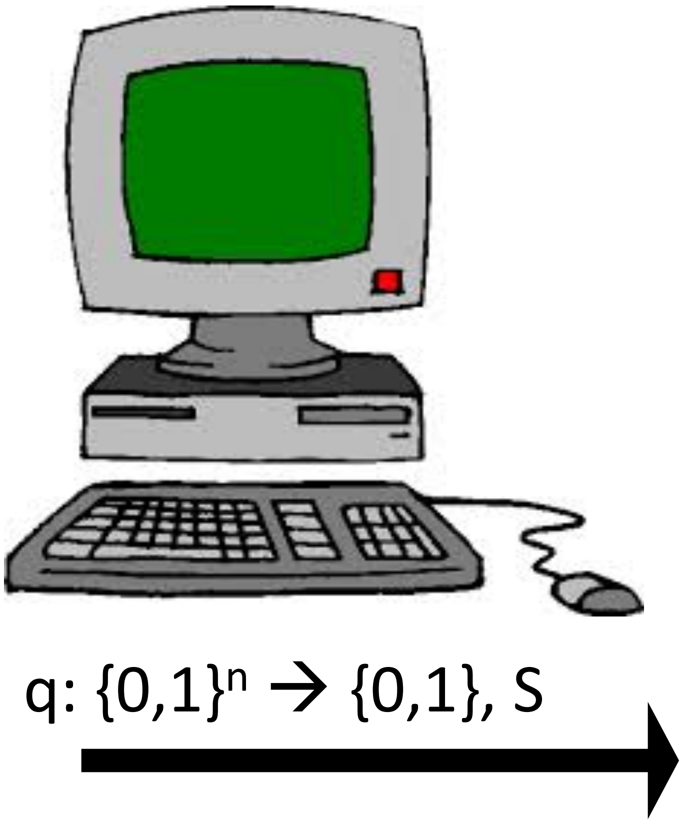
$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

A(G)

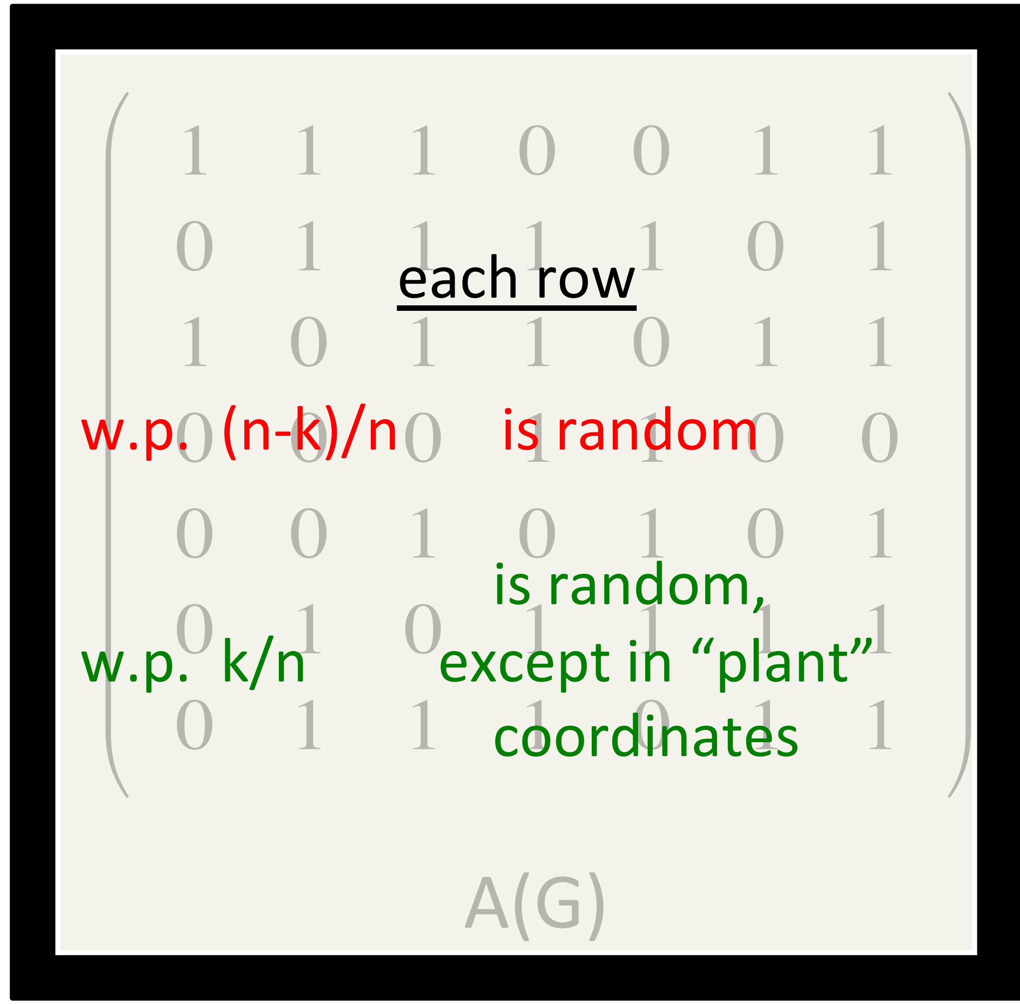
Statistical Algorithms for BPC



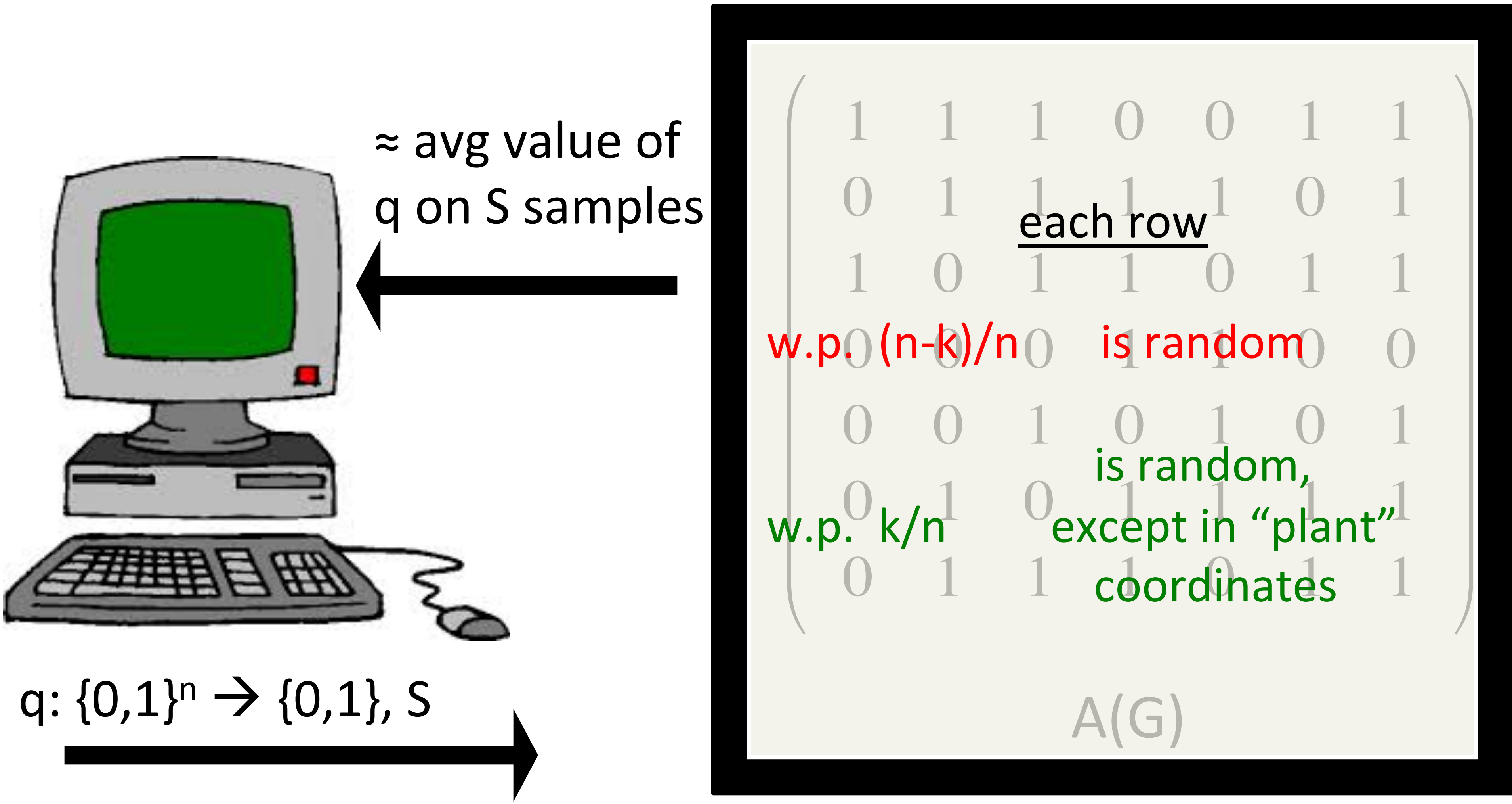
Statistical Algorithms for BPC



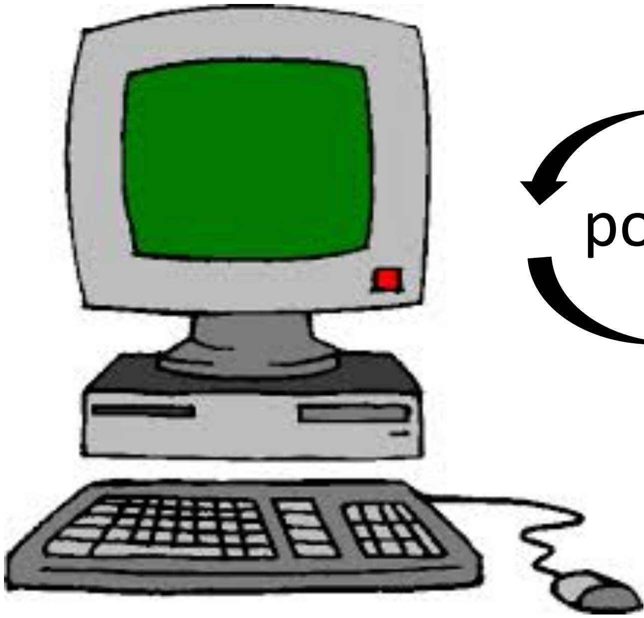
q: $\{0,1\}^n \rightarrow \{0,1\}$, S

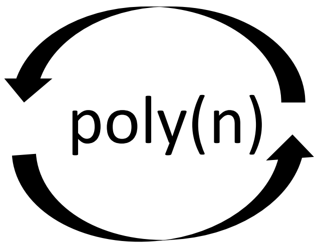


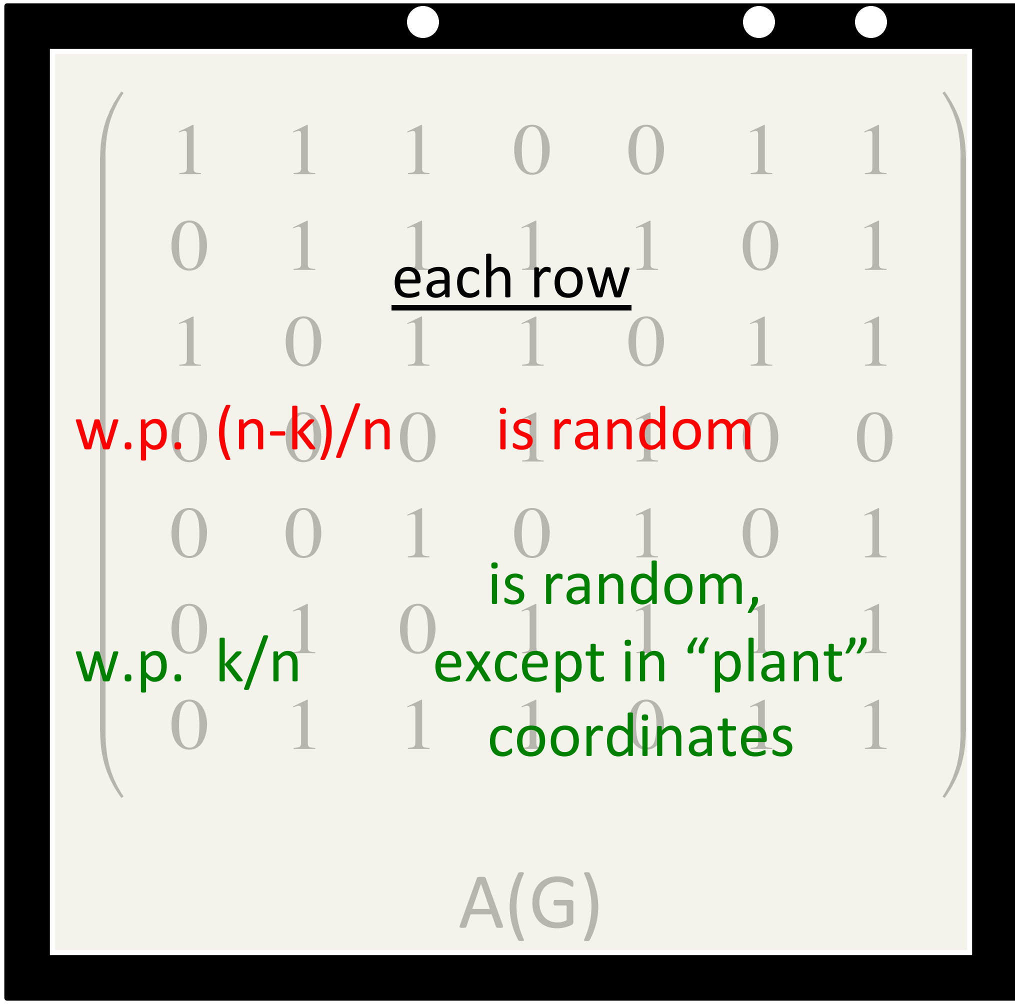
Statistical Algorithms for BPC



Statistical Algorithms for BPC







Results

- Extension of statistical query model to optimization.
- Proving tighter, more general, lower bounds, which apply to learning also.

Gives a new tool for showing problems are difficult.

Results

- Main result (almost): No statistical algorithm making a polynomial number of queries with sample sizes $o(n^2/k^2)$, can find planted cliques of size k.
  - *intuition*: ∃ many planted clique distributions with small "overlap" (nearly orthogonal in some sense), which are hard to tell from normal E-R graphs.
  - Implies that many ideas will fail to work, including Markov chain approaches [Frieze-Kannan '03] for our version of the problem.

Overview

Statistical oracles are a new lens through which we can study existing algorithms.

Statistical lower bounds can help explain why certain problems appear intractable.

This idea gives the first general lower bound for the notorious planted clique problem.

Any Questions?