# On Finding Planted Cliques in Random Graphs

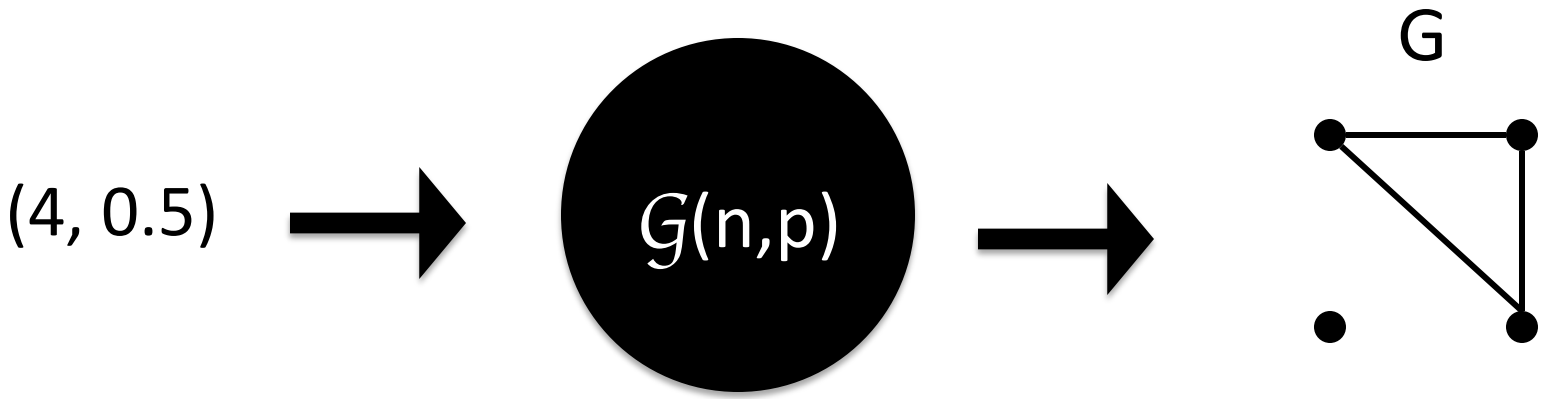
GraphEx 2014

Lev Reyzin
UIC

joint with Feldman, Grigorescu, Vempala, & Xiao in STOC ’13

# Erdős-Rényi Random Graphs

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

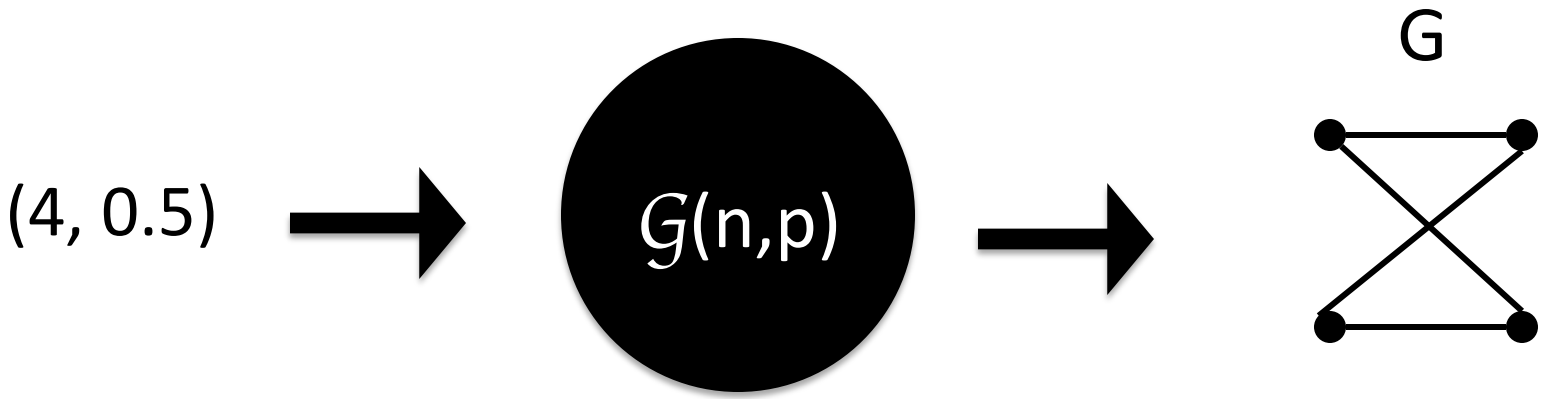


# Erdős-Rényi Random Graphs

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

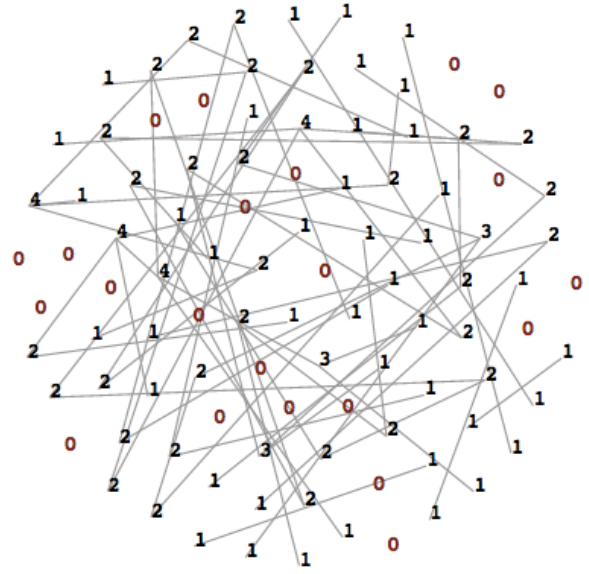
# Erdős-Rényi Random Graphs

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

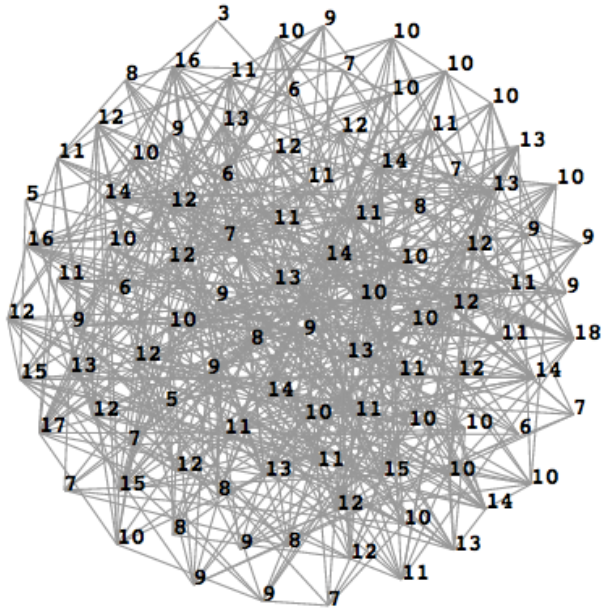


# Typical Examples

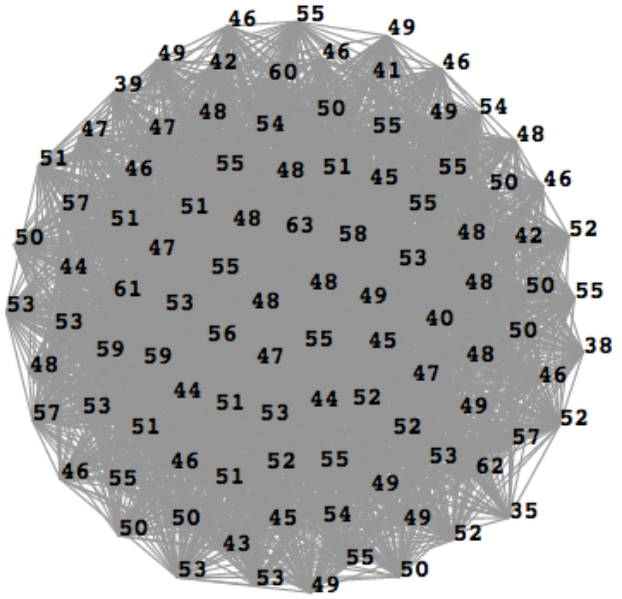
n = 100, p = 0.01



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 $k(n) \approx 2\lg_2(n)$

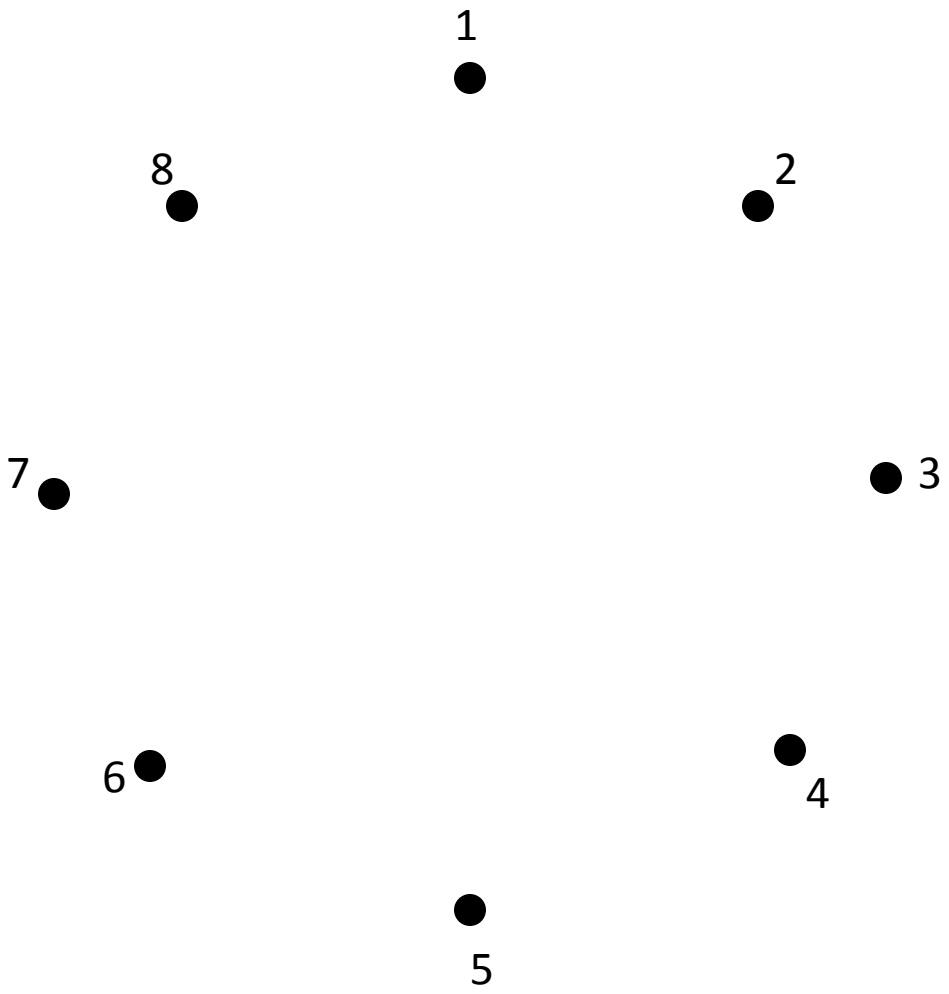
# w.h.p. for G ~ $\mathcal{G}$(n,½), k(n) ≈ $2\lg_2(n)$

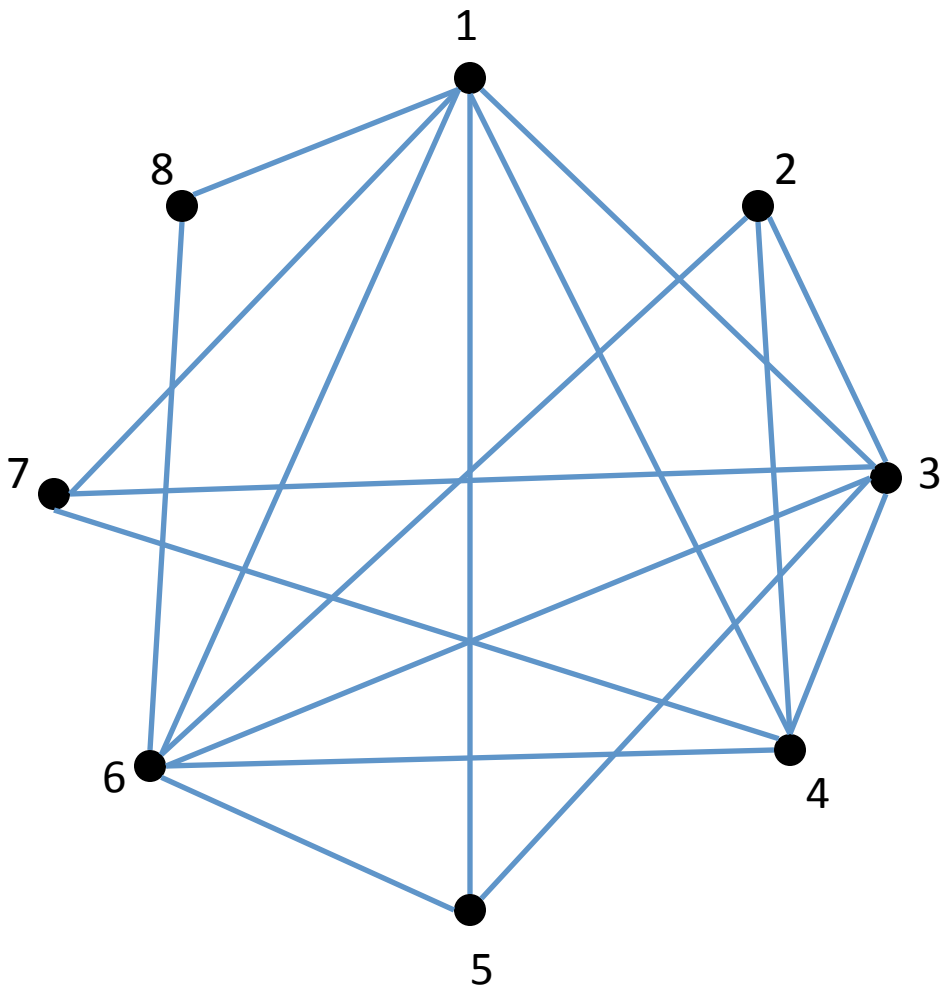
*why?*

- let $X_k$ be the number of cliques in G ~ $\mathcal{G}$(n,.5)

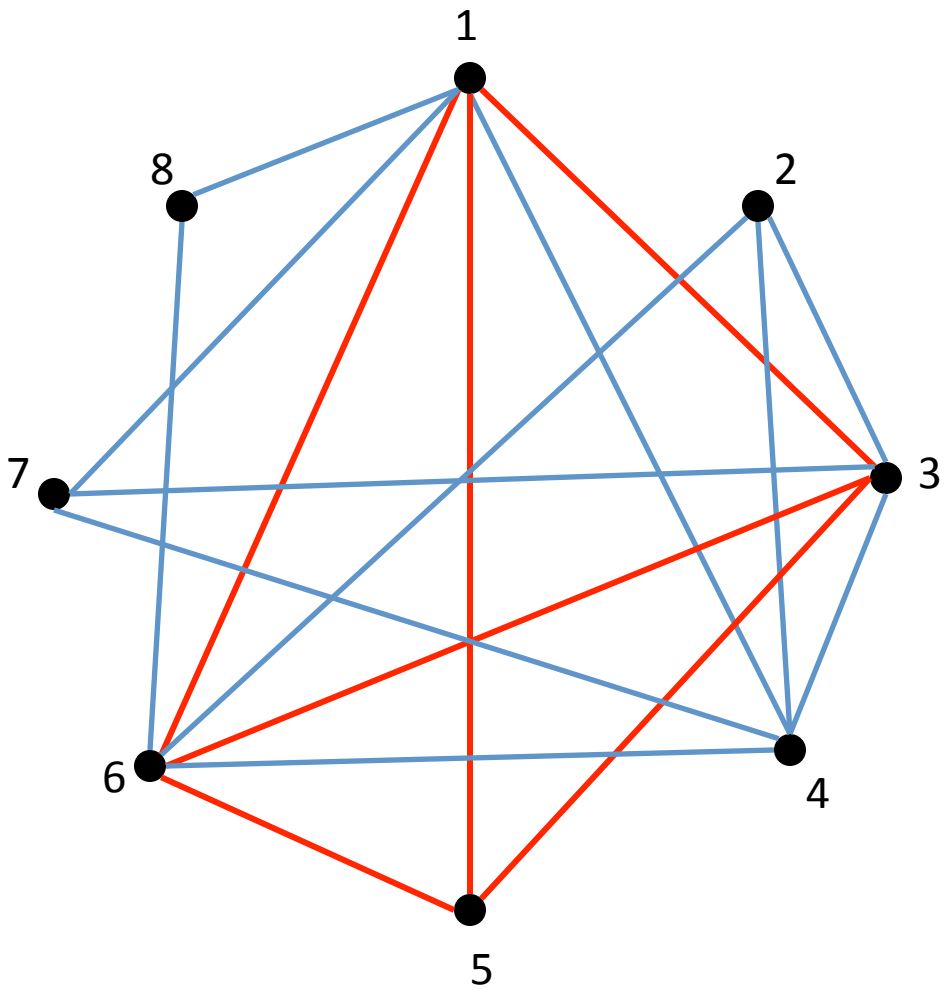
- $\mathrm{E}[X_K] = \binom{n}{k} 2^{-\binom{k}{2}}$ < 1 for k > ≈ $2\lg_2 n$

- 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]


1
2
3
4
5
6
7
8




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.

W[1]-Hard

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

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

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

- Simply 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)

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

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

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

# 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 Clique

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

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

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

# 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)

Obvious relationship to community detection and other applications.

# 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+2
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 $k \leq 2\lg_2 n + 1$.

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+2
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]

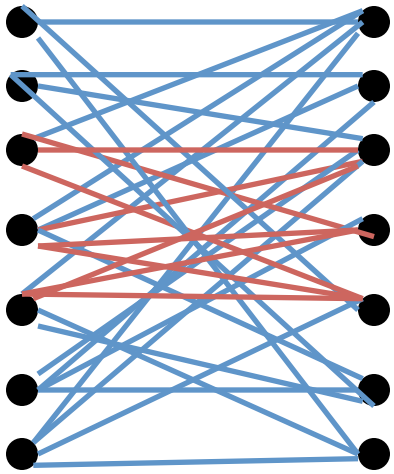
---

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

1) find 2nd 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 $\geq \frac{3}{4}k$ 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]



In fact, (bipartite) planted clique was recently used as an 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)

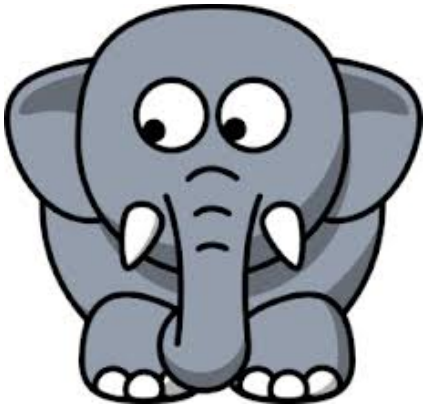
- $k = c\,n^{1/2}$ is best so far. [Alon-Krivelevich-Sudokov '98]

---

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

# Statistical Query Learning

[Kearns ’93]



$(x_1, y_1)$

$(x_2, y_2)$

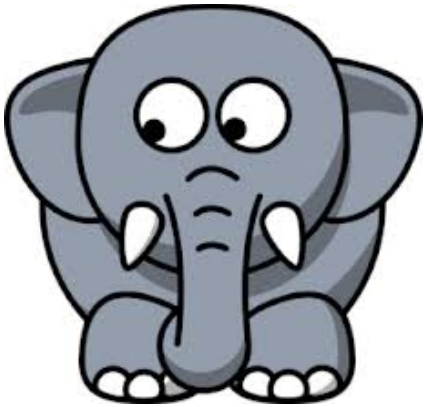
$(x_3, y_3)$

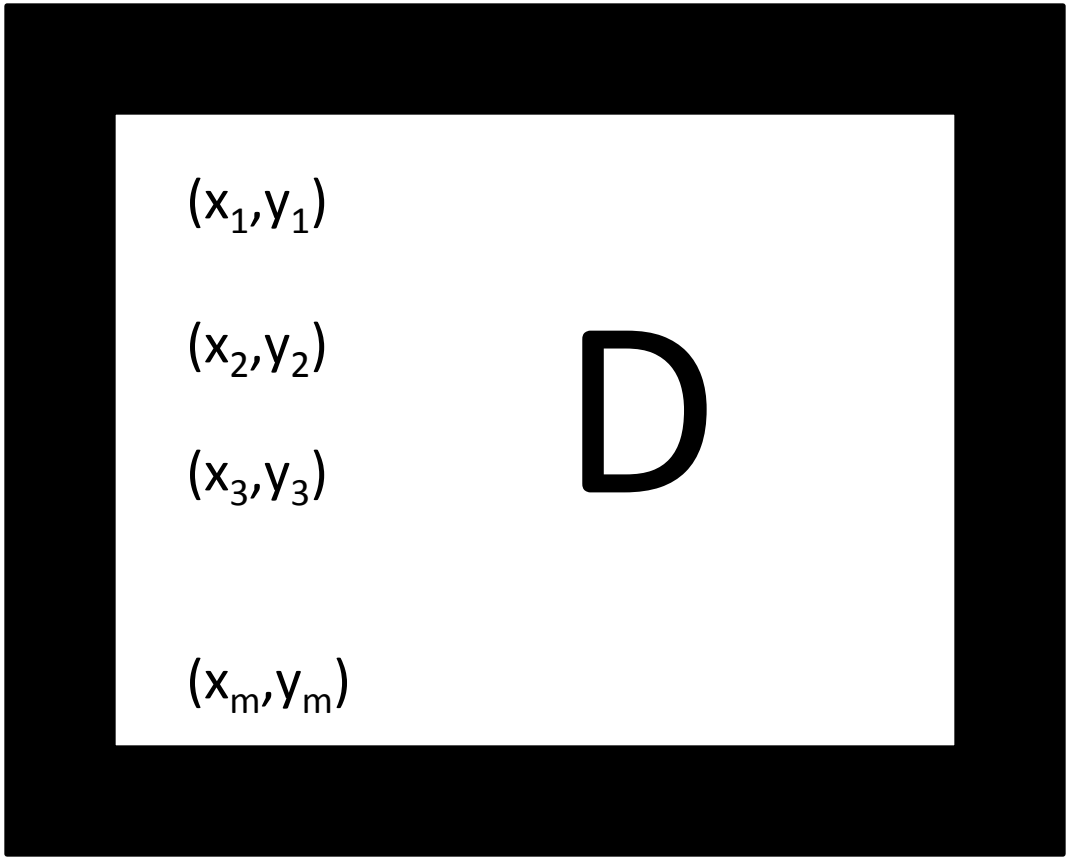
D

$(x_m, y_m)$

# Statistical Query Learning

[Kearns ’93]





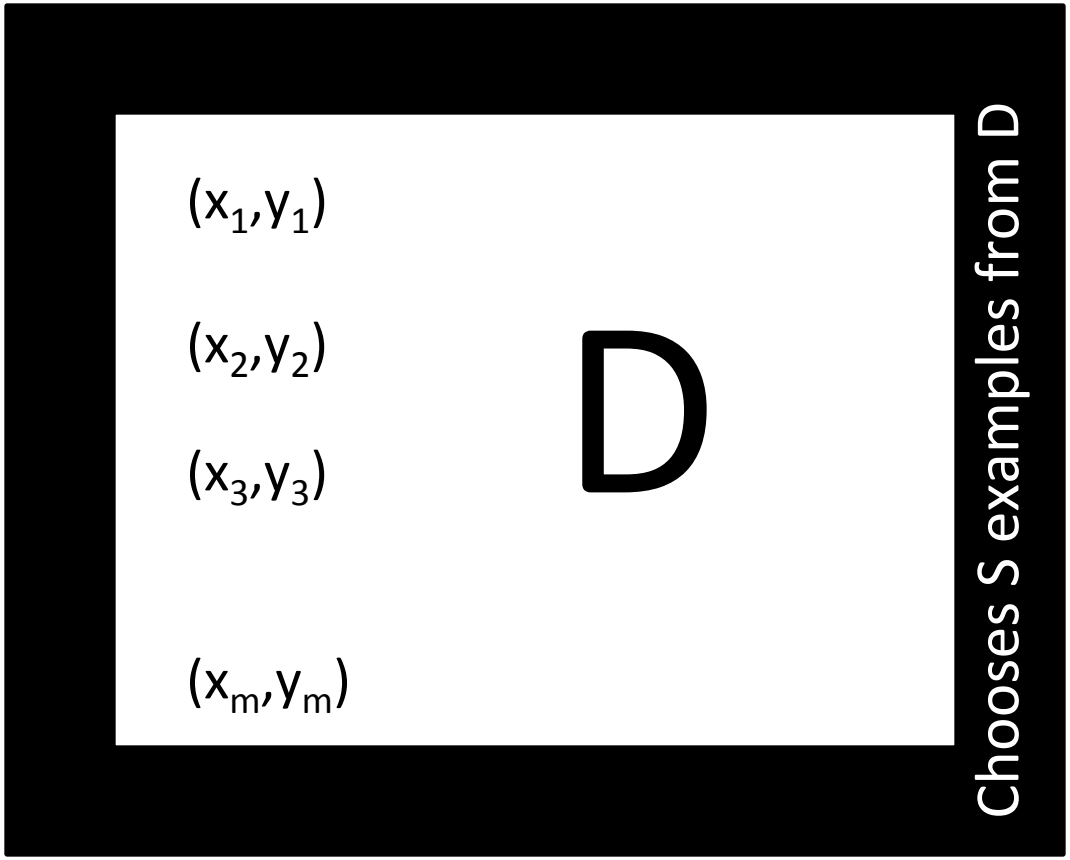

$q(x, f(x))$, $S = poly$

where $x \in X$

$q: X \times \{0,1\} \rightarrow \{0,1\}$

# Statistical Query Learning

[Kearns '93]




$q(x, f(x))$, $S$ = poly

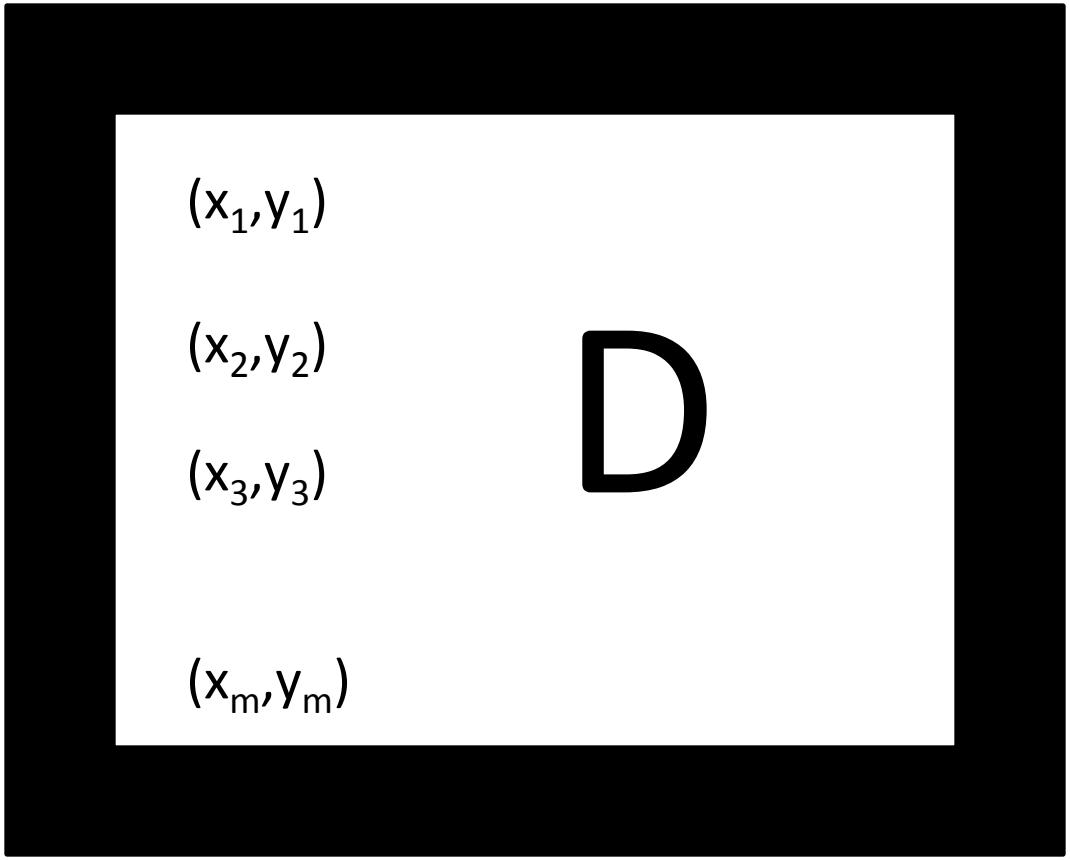
where $x \in X$

$q: X \times \{0,1\} \rightarrow \{0,1\}$

$\approx$ avg $q(x, f(x))$ over S examples

# Statistical Query Learning

[Kearns '93]

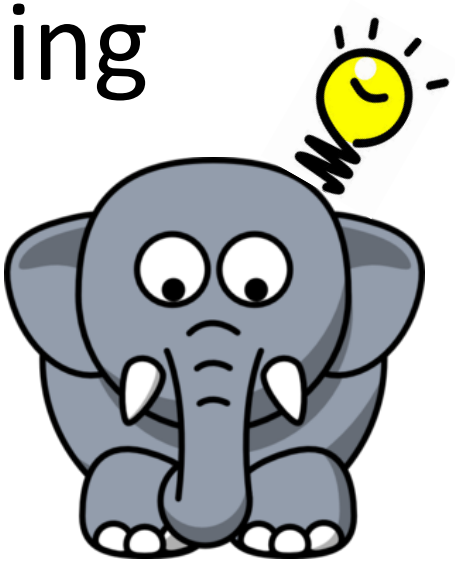


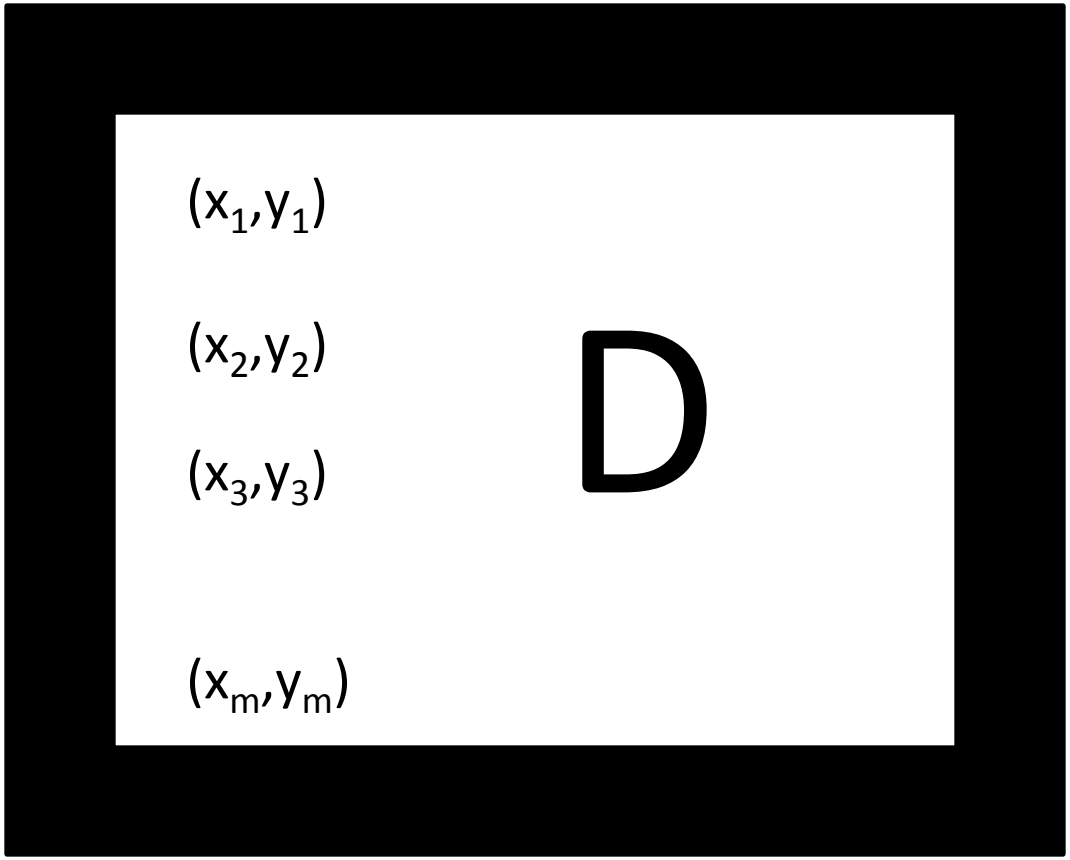

q

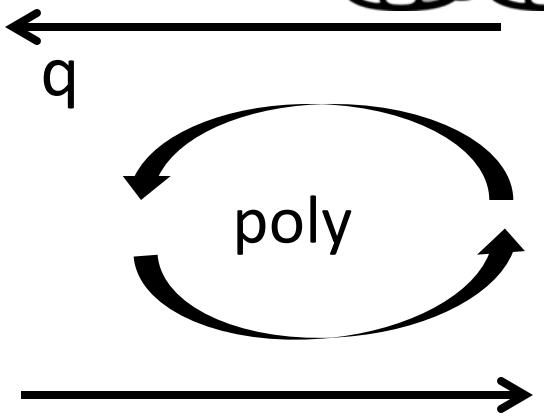
poly

# 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 normal "PAC" sense) with noise!
- Theorem [Kearns '93]: parity functions are not learnable with statistical queries.

  *proof idea*: *b/c the parity functions are orthogonal under U, queries are either uninformative or "eliminate" one wrong parity 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 normal "PAC" sense) with noise!
- Theorem [Kearns '93]: parity functions 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). Parity 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 normal "PAC" sense) with noise!
- Theorem [Kearns '93]: parity functions 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

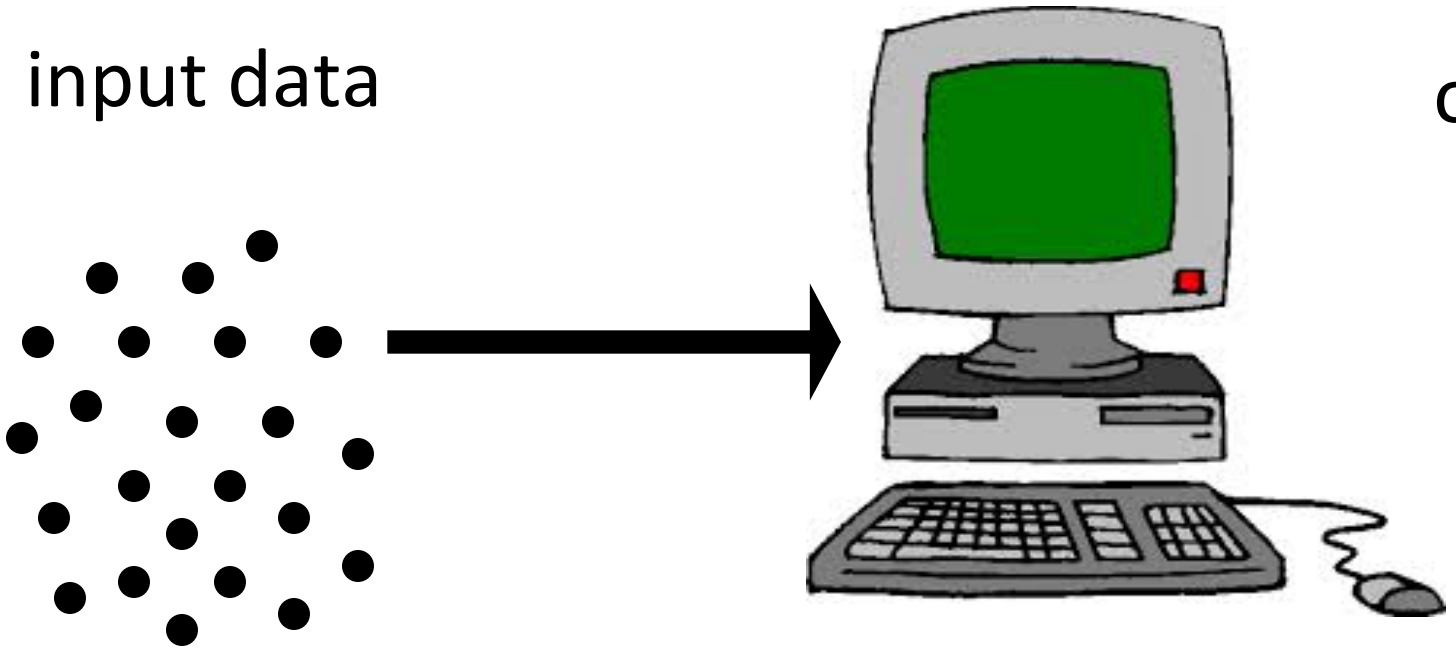
Statistical queries and statistical dimension from learning theory are an explanation as to why certain classes are hard to learn.

(almost all our learning algorithms are statistical)

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

# Traditional Algorithms

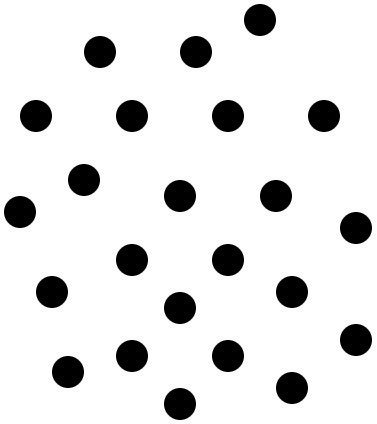
input data

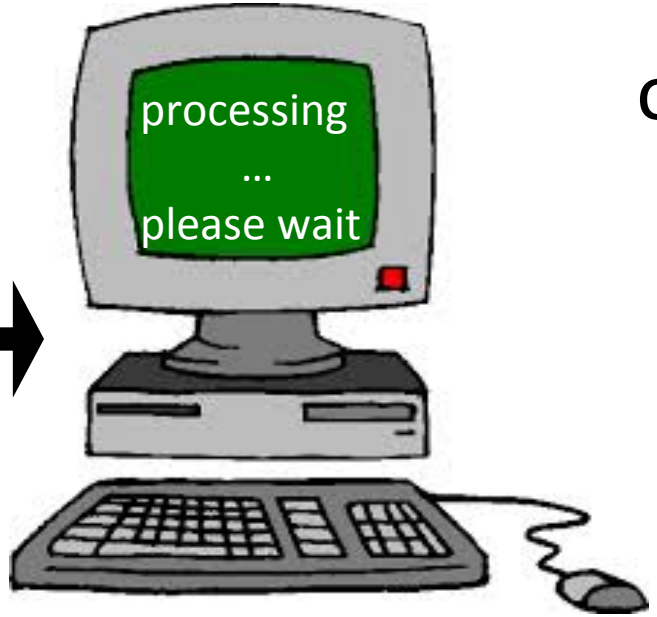


output

# Traditional Algorithms

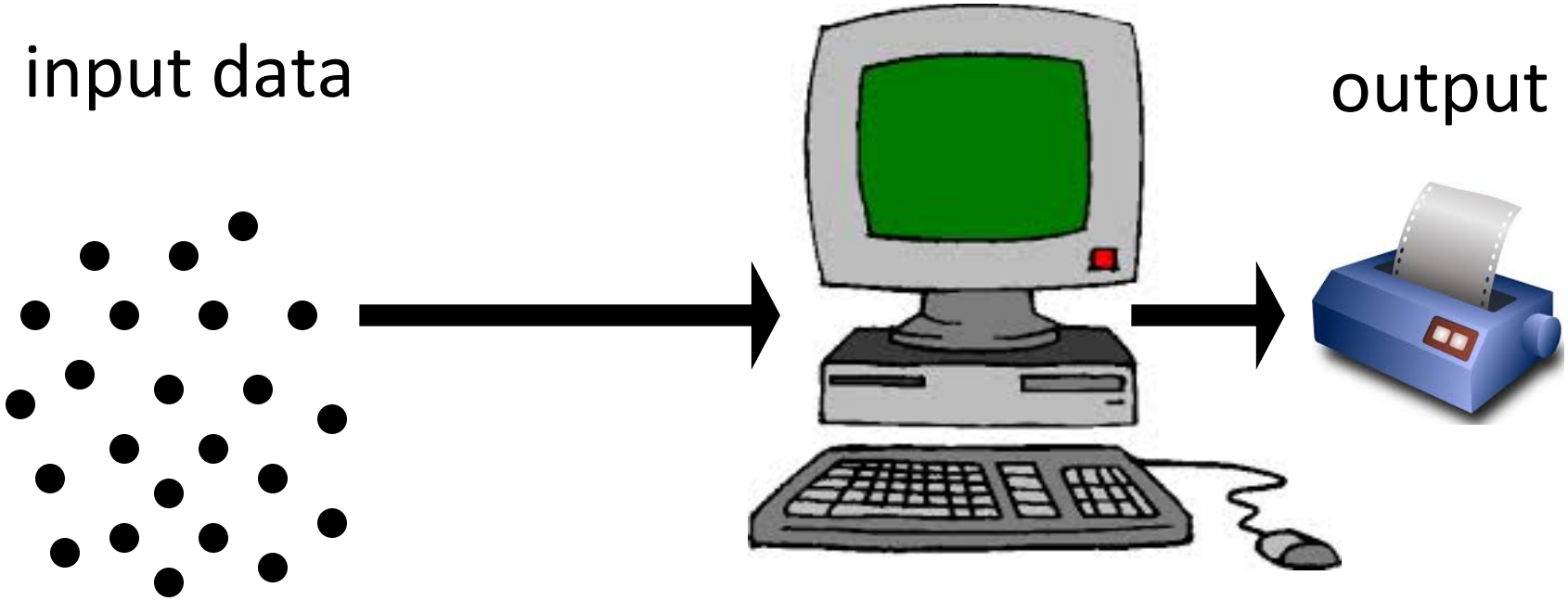
input data





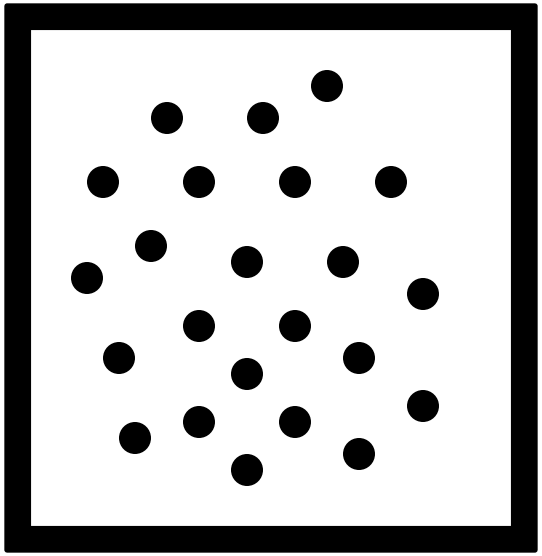

output

# Traditional Algorithms



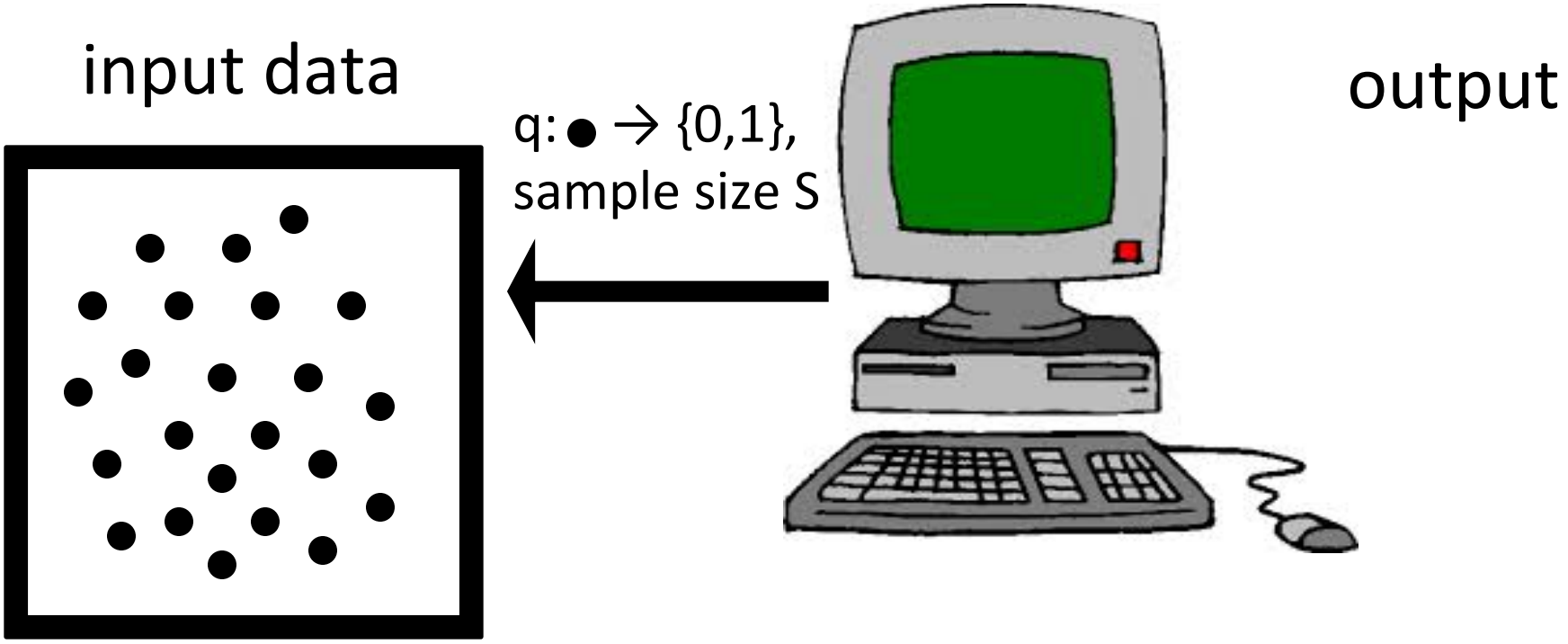
# Statistical Algorithms

input data

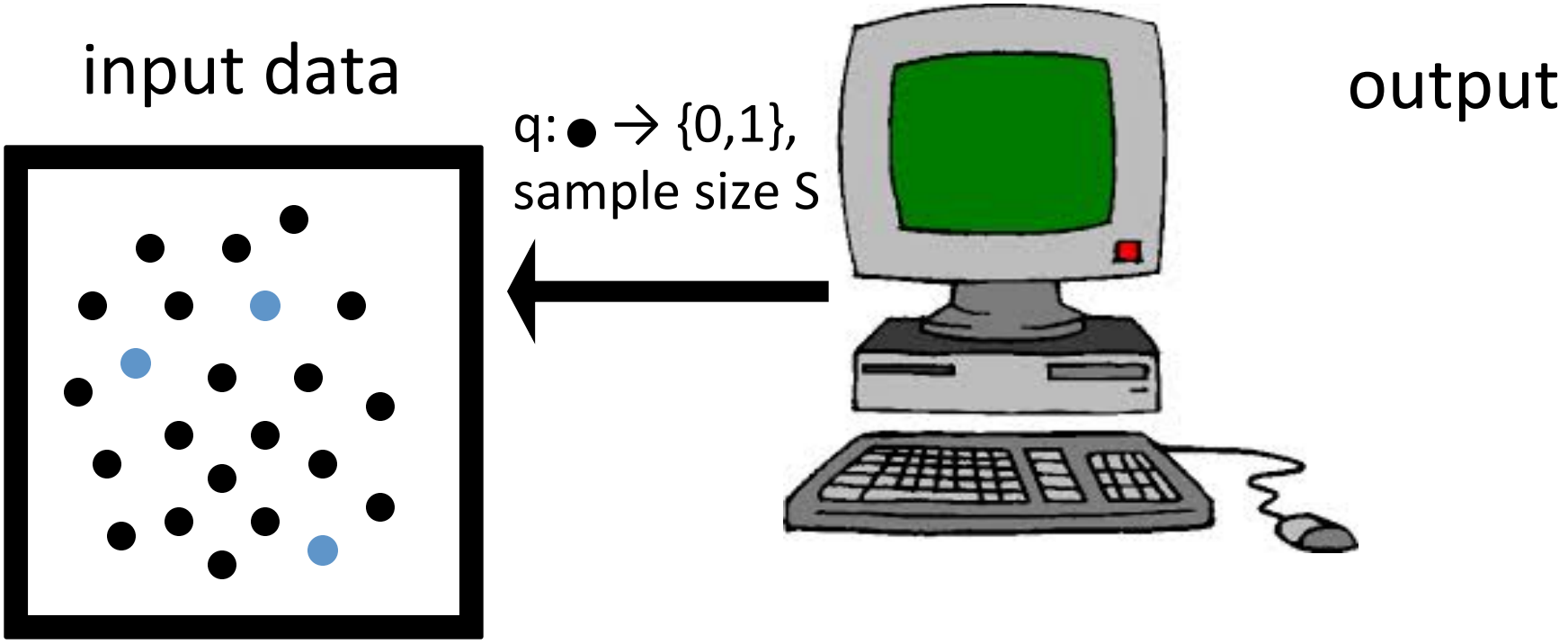


output

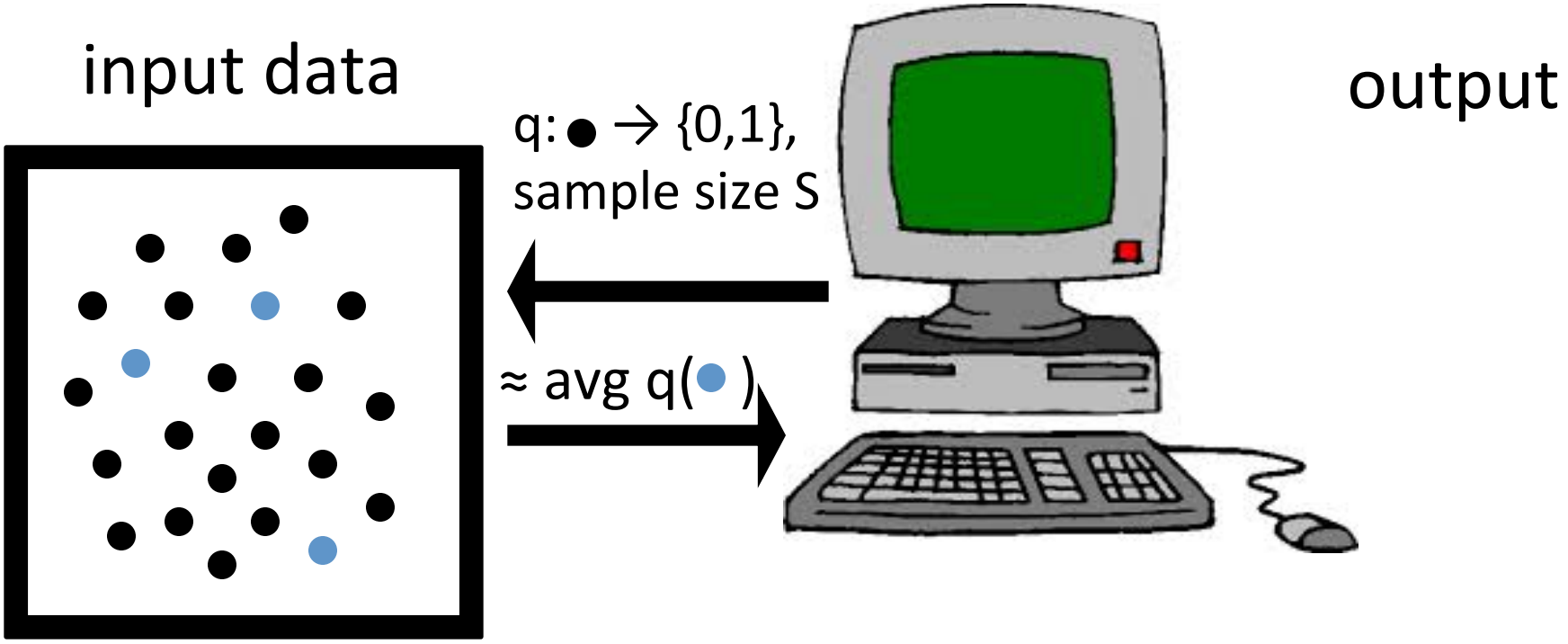
# Statistical Algorithms



# Statistical Algorithms

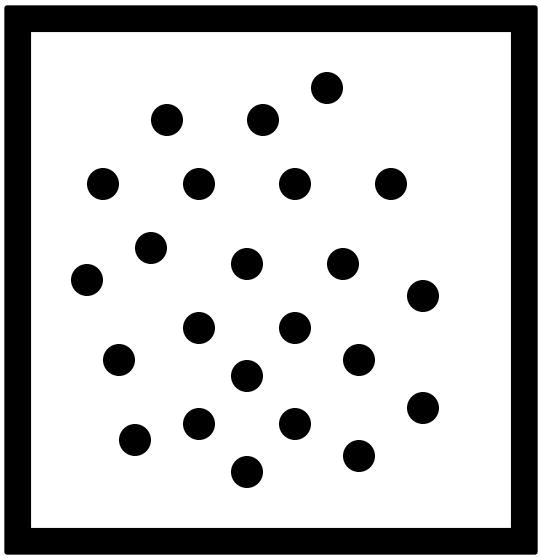


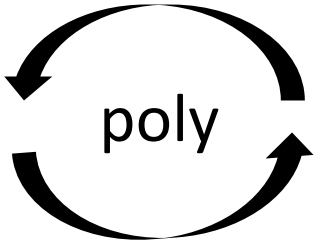
# Statistical Algorithms



# Statistical Algorithms

input data

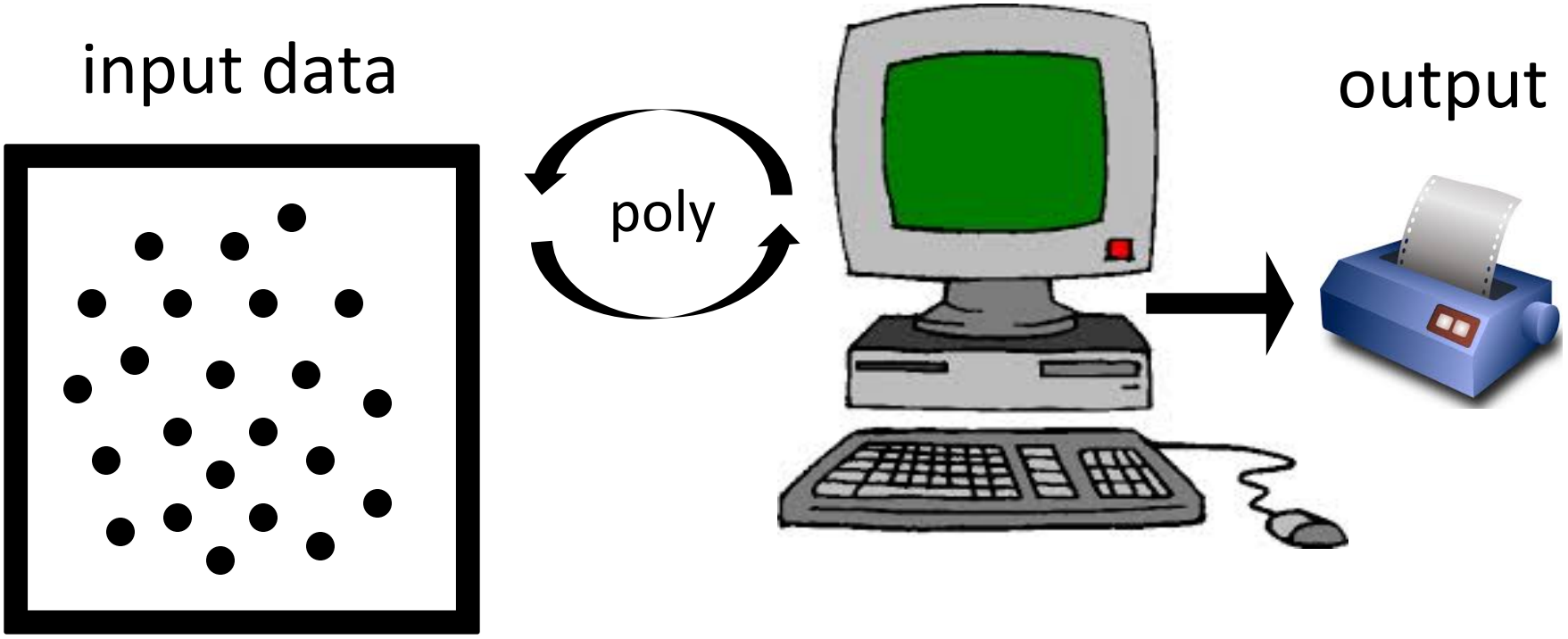




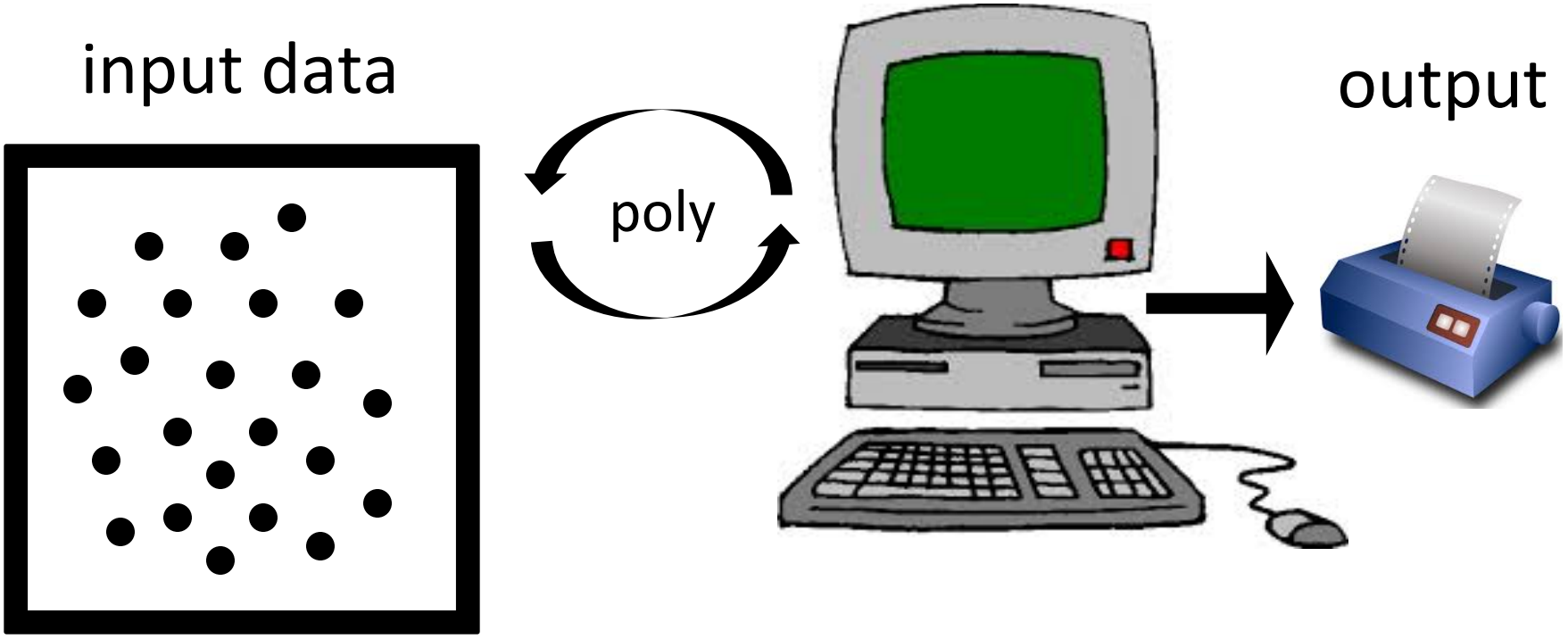


output

# Statistical Algorithms

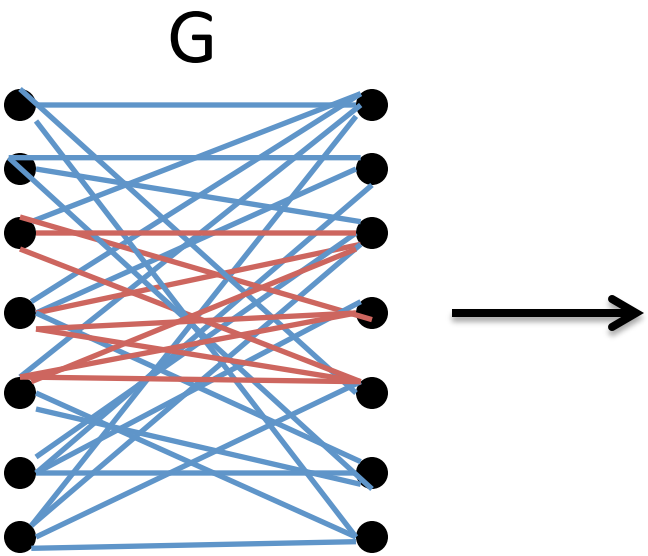


# Statistical Algorithms




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

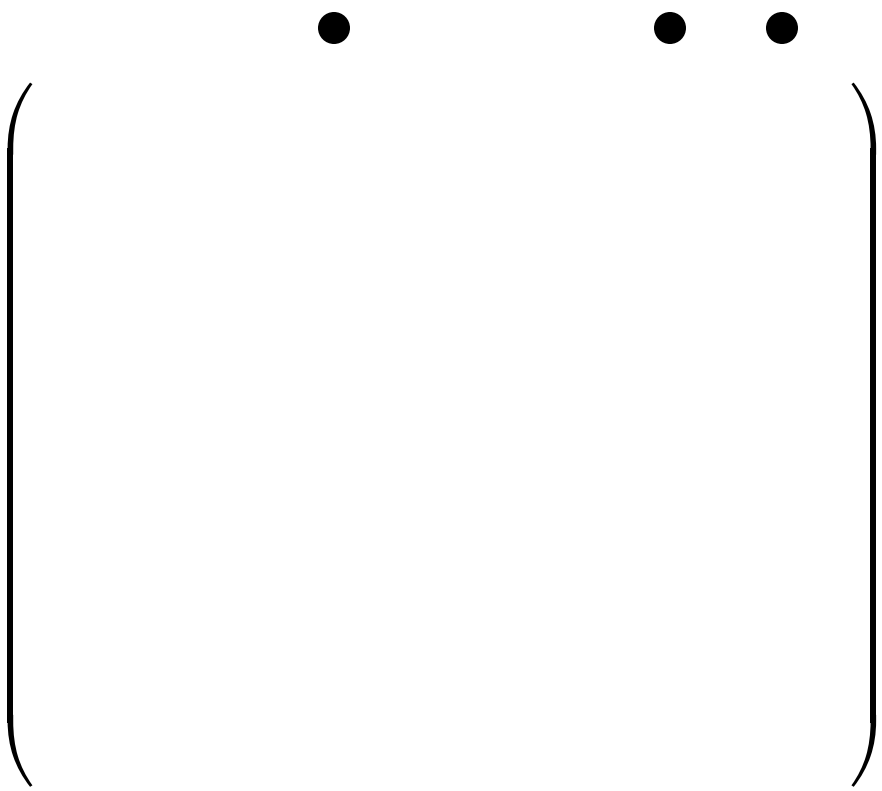
# Bipartite Planted Clique




A(G)

# Bipartite Planted Clique

| each row | |
|---|---|
| w.p. (n-k)/n | is random |
| w.p. k/n | is random, except in "plant" coordinates |



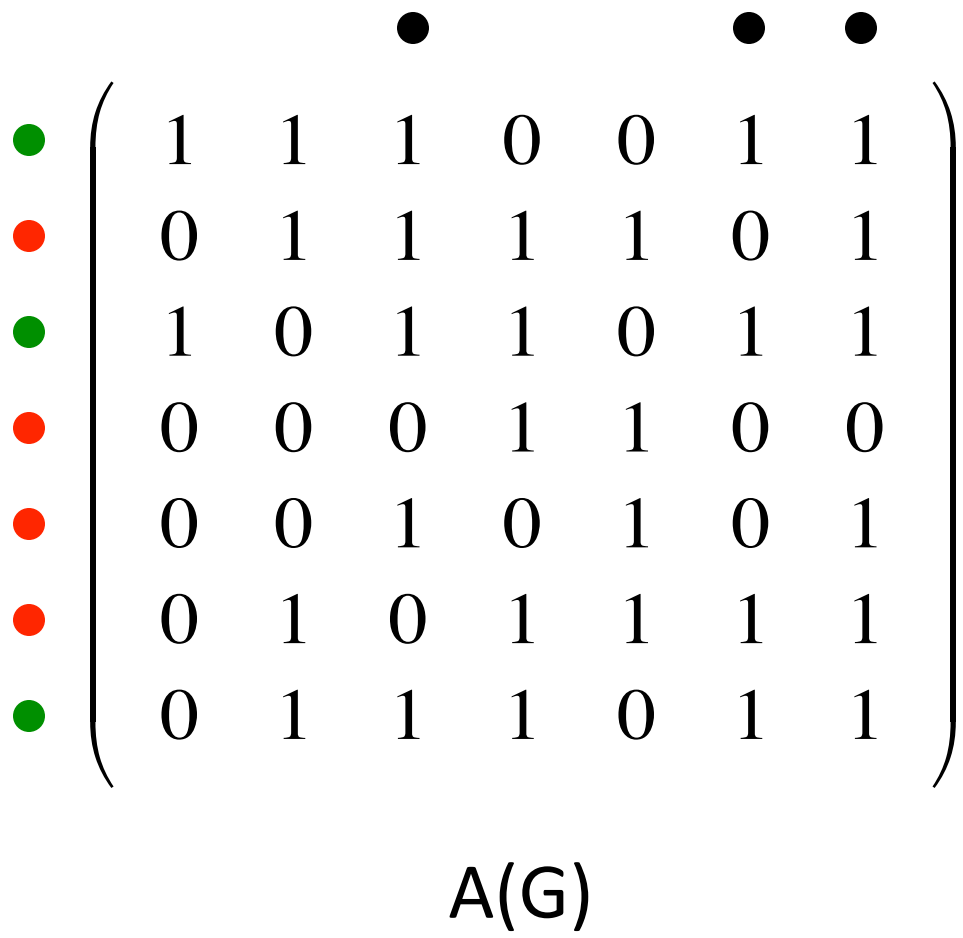
A(G)

# Bipartite Planted Clique

each row

w.p. (n-k)/n is random

w.p. k/n is random, except in "plant" coordinates




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

each row

w.p. (n-k)/n is random

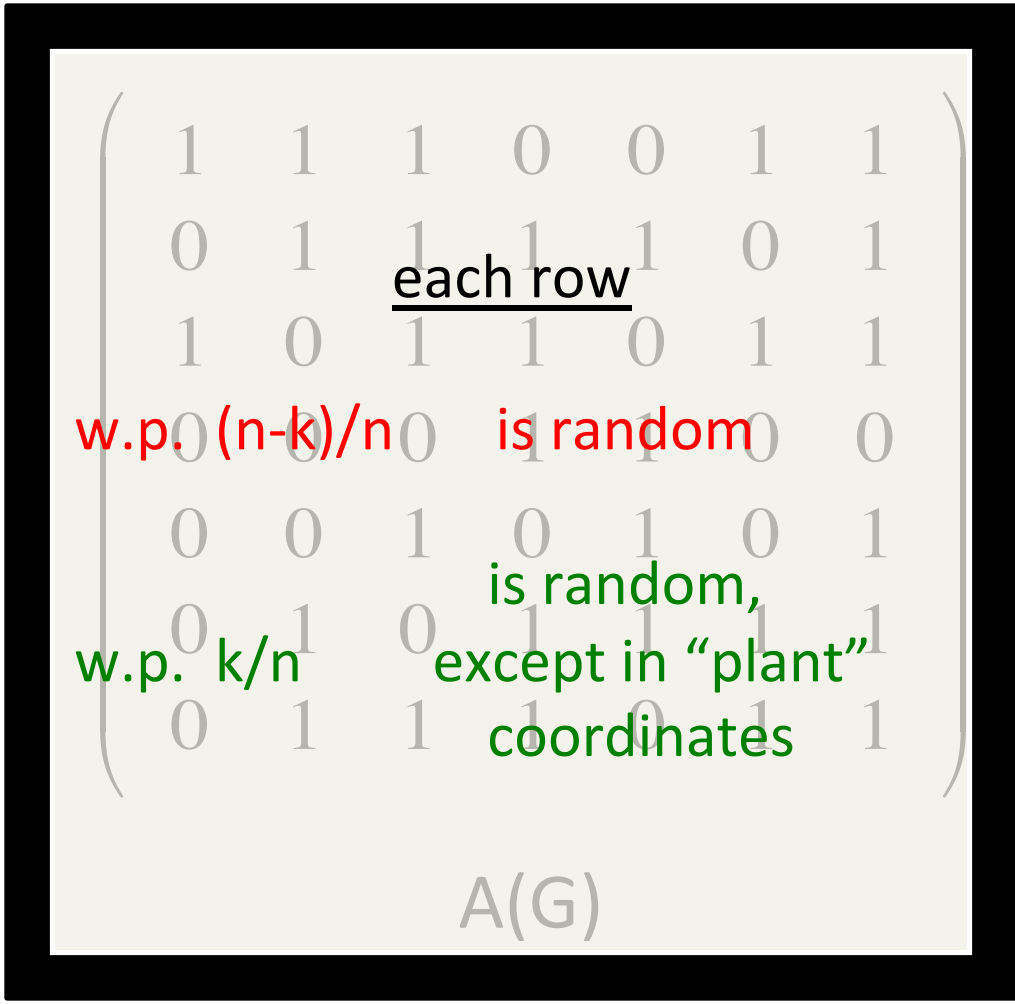
is random,

w.p. k/n except in "plant" coordinates

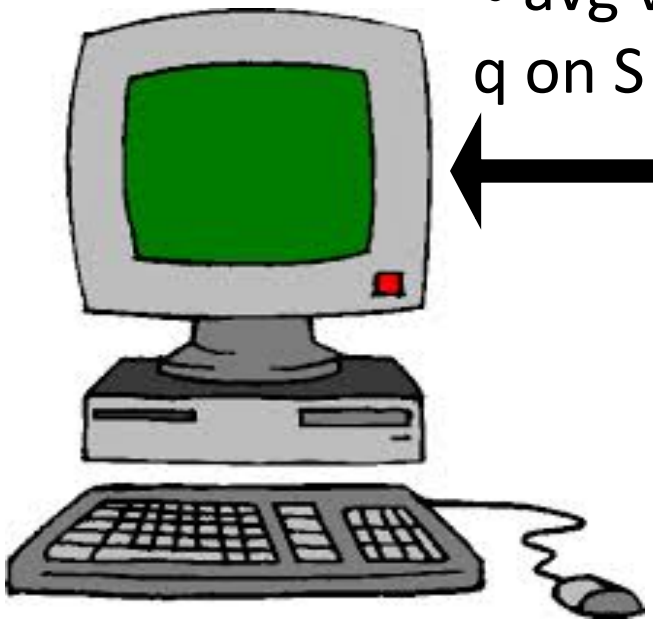
A(G)

# Statistical Algorithms for BPC

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

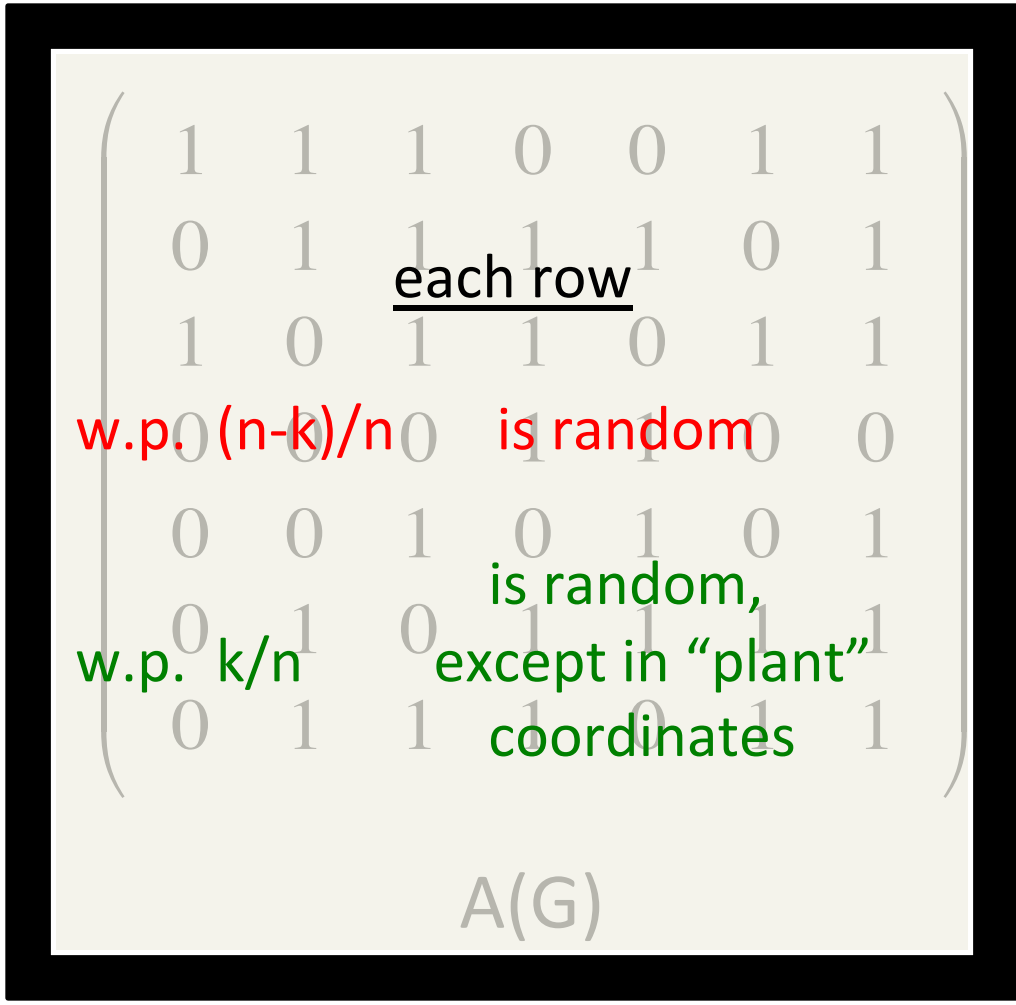


# Statistical Algorithms for BPC

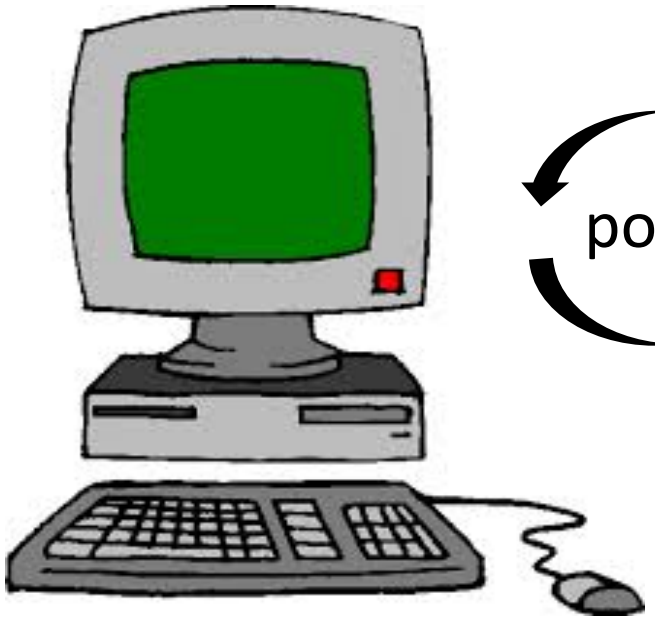


≈ avg value of
q on S samples

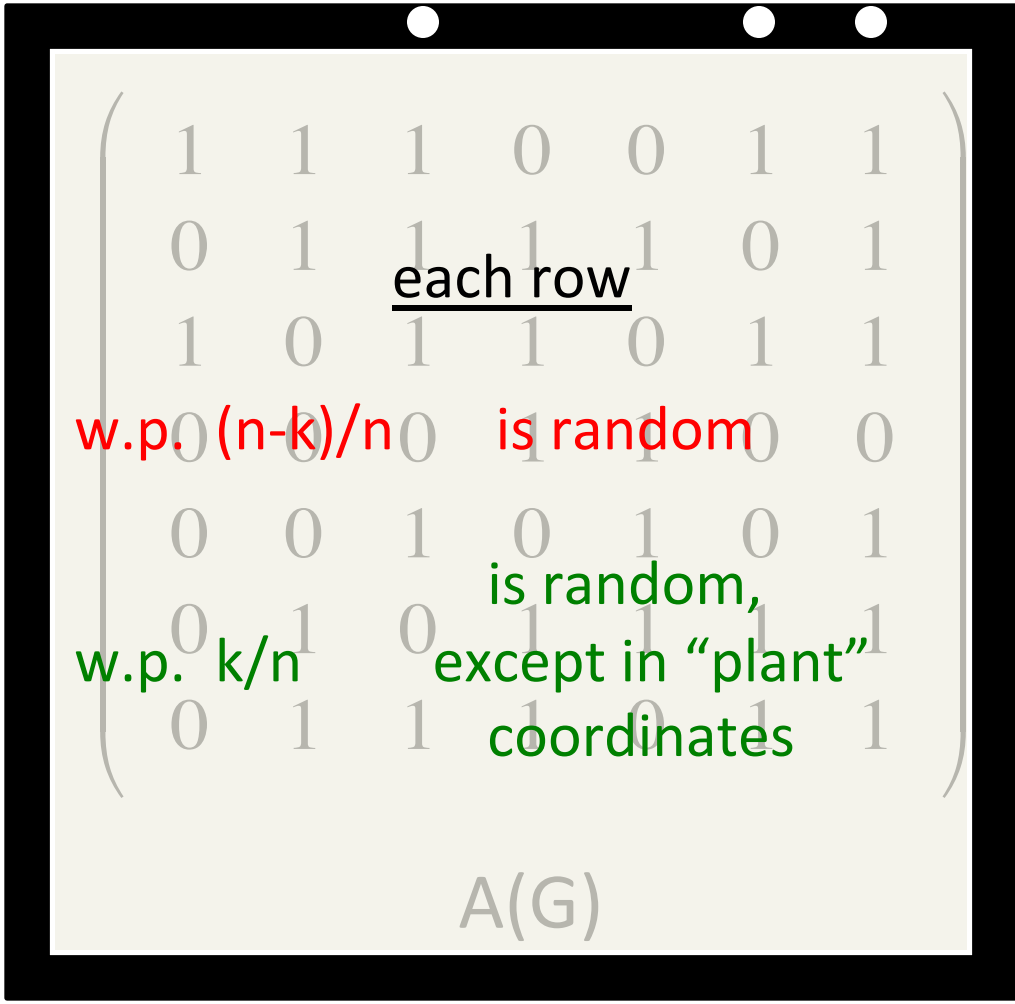
q: $\{0,1\}^n \rightarrow \{0,1\}$, S



# Statistical Algorithms for BPC



poly(n)



# 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): for any $\varepsilon>0$, no poly time statistical alg. making queries with sample sizes $o(n^2/k^2)$, can find planted cliques of size $n^{1/2-\varepsilon}$.
  - *intuition*: $\exists$ 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.

# Results

- Main result (almost): for any $\varepsilon>0$, no poly time statistical alg. making queries with sample sizes $o(n^2/k^2)$, can find planted cliques of size $n^{1/2-\varepsilon}$.
  - *intuition*: $\exists$ 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.
- Since this work, an integrality gap of $\approx n^{1/2}$ was shown for planted clique, giving further evidence for its hardness. [Meka-Wigderson '13]

# Any Questions?