Lecture 6: Statistical vs Computational notions of tractability

Lecturer: Jay Mardia

April 14, 2021

Recap

Comparison of Models

For two statistical models $\mathcal{M} = (\mathcal{X}, (P_\theta)_{\theta\in\Theta})$ and $\mathcal{N} = (\mathcal{Y}, (Q_\theta)_{\theta\in\Theta})$, when can we say model $\mathcal{M}$ is stronger than model $\mathcal{N}$? How can we translate a solution to model $\mathcal{N}$ to a solution to model $\mathcal{M}$?

Definition (Deficiency; Le Cam (1964))

For two statistical models $\mathcal{M} = (\mathcal{X}, (P_\theta)_{\theta\in\Theta})$ and $\mathcal{N} = (\mathcal{Y}, (Q_\theta)_{\theta\in\Theta})$, we say $\mathcal{M}$ is ε-deficient relative to $\mathcal{N}$ if

- for any finite subset $\Theta_0 \subseteq \Theta$;
- for any finite action space $\mathcal{A}$;
- for any loss function $L : \Theta \times \mathcal{A} \to [0, 1]$;
- for any decision rule $\delta_\mathcal{N}$ for model $\mathcal{N}$;

there exists a decision rule $\delta_\mathcal{M}$ for model $\mathcal{M}$ such that

$$R_\theta(\delta_\mathcal{M}) \le R_\theta(\delta_\mathcal{N}) + \varepsilon, \quad \forall\theta \in \Theta_0.$$

Recap: Randomization of statistical models

Definition (Randomization)

For two statistical models $\mathcal{M} = (\mathcal{X}, (P_\theta)_{\theta \in \Theta})$ and $\mathcal{N} = (\mathcal{Y}, (Q_\theta)_{\theta \in \Theta})$, we say $\mathcal{N}$ is a randomization of $\mathcal{M}$ if there exists a stochastic kernel $\mathsf{K} : \mathcal{X} \to \mathcal{Y}$ such that $Q_\theta = \mathsf{K} P_\theta$ for all $\theta \in \Theta$, i.e.

$$Q_\theta(y) = \sum_{x \in \mathcal{X}} P_\theta(x) \mathsf{K}(y \mid x), \quad \forall y \in \mathcal{Y}, \theta \in \Theta.$$

$\mathcal{N}$ is a randomization of $\mathcal{M} \implies \mathcal{M}$ is 0-deficient relative to $\mathcal{N}$

Theorem

Model $\mathcal{M}$ is ε-deficient relative to $\mathcal{N}$ if and only if there exists a stochastic kernel $\mathsf{K} : \mathcal{X} \to \mathcal{Y}$ such that

$$\sup_{\theta \in \Theta} \|Q_\theta - \mathsf{K} P_\theta\|_{\mathrm{TV}} \leq \varepsilon.$$

showing deficiency results $\iff$ showing randomization results

Recap: Sufficiency

- statistical model $\mathcal{M} = (\mathcal{X}, (P_\theta)_{\theta\in\Theta})$, with $X \sim P_\theta$
- let $T = T(X)$ be a function of X
- T-induced model $\mathcal{N} = (\mathcal{T}, (P_\theta \circ T^{-1})_{\theta\in\Theta})$
- when do we have $\Delta(\mathcal{M}, \mathcal{N}) = 0$?

Theorem

$\Delta(\mathcal{M}, \mathcal{N}) = 0$ if and only if T is a sufficient statistic, i.e. $\theta - T - X$ forms a Markov chain.

A puzzle

RSA (Rivest-Shamir-Adleman) is a public-key cryptosystem that is widely used for secure data transmission.

The security of RSA relies on the practical difficulty of factoring the product of two large prime numbers, the “factoring problem”.

- $\mathcal{M}$: $X \sim$ uniform distribution on all positive integers smaller than N.
- $\mathcal{N}$: $Y \sim$ distribution on prime factors induced by X.

$$\text{By sufficiency, } \Delta(\mathcal{M}, \mathcal{N}) = 0$$

So factoring should be easy...? There is clearly something more going on.

Today's plan

- What is computation? What is efficient computation?
- Two notions of tractability, do they coincide? Planted Clique as a case study.
- Example of problems with statistical-computational gaps
- How to reason about and predict stat-comp gaps?

What is computation?

A **finitely describable** set of instructions that takes an **arbitrarily large** input, and produce an answer.

```
Eg: Algorithm to compute mean of $X_1, .., X_n$
$N \leftarrow$ size of input
$Y \leftarrow 0$
For $i = 1, 2, ..., N$, $Y \leftarrow Y + \frac{X_i}{N}$
Output $Y$.
```

- Can formalize definition of 'computer' in various ways. Turing machines are the theoretical workhorse. Equivalent to programming languages like C++, python.
- "Time" = number of steps taken by algorithm. Good proxy for resource / power / effort required to solve the problem.

How do we define computational efficiency?

If size (number of bits) of input is n, an algorithm running in time poly(n) is called efficient.

Reasons:

- Corresponds to practical reality.
- Agnostic to precise low level definition of ‘algorithm’.
- Allows composition of efficient algorithms.

Factoring is believed to not be solvable in poly time. So computational notions might not be well captured by TV distance or Le Cam distance. Questions?

Do statistical and computational tractability coincide?

Statistical Tractability (Shorthand): Usually controlled by some “signal to noise ratio” (SNR). As SNR increases, problem becomes “easier”
Eg: n/σ^2 is SNR for gaussian mean estimation.

Computational tractability: Running time of algorithm. As runtime decreases, problem becomes easier.

- Need both notions of tractability to solve a real-world learning problem.
- If SNR is large enough to let us solve the problem, does there always exist a computationally efficient algorithm?

Lets look at a concrete example problem.

Example 1: Planted Clique

$G = (V, E)$, a graph, is a collection of vertices V and edges E.
$G(n, 1/2)$ is a graph on n vertices where each possible edge is included iid with probability $1/2$.
Represented via edge indicators $X_{ij} \sim \text{Bern}(1/2)$ (i, j are vertices)

Planted clique detection problem:

$$H_0 : G(n, 1/2) \text{ vs } H_1 : G(n, 1/2, k) = G(n, 1/2) + K$$

K denotes planted clique and is a uniformly random subset of size k.

As k, the SNR increases, problem becomes easier. Fix these two parameters in your mind.

How to solve this?

Idea: How many cliques of size m do we expect to see in $G(n, 1/2)$?

$$\mathit{Prob}(\exists m\text{-clique in } G(n,1/2)) \leq \mathbb{E}[\text{number of cliques of size } m] = \frac{\binom{n}{m}}{2^{\binom{m}{2}}}.$$

- For $m \geq (2+\epsilon)\log n$, this number is $o(1)$.
- So if $k > (2+\epsilon)\log n$, can distinguish between the two by enumerating over all subsets of size $(2+\epsilon)\log n$ and checking for a clique. Takes time $n^{O(\log n)}$.

Is there a (possibly inefficient) algorithm that works for smaller clique sizes k?

Recap: Le Cam's Lemma

Basic technique: For a hypothesis testing problem,

$$H_0 : P \text{ vs } H_1 : Q$$

1. Show $TV(P, Q)$ is small. Then distinguishing between H_0 and H_1 must incur large probability of error.
2. TV is not always easy to calculate. Upper bound it with various divergences that are easier to calculate.

In this course we will also see much fancier variations on this basic scheme.

Impossibility result for planted clique

Cliques are the main helpful feature, and we know $\mathbb{E}[\text{number of cliques of size } m] = \frac{\binom{n}{m}}{2^{\binom{m}{2}}}$. Since this is $\omega(1)$ for $m < (2-\epsilon)\log n$, we expect our planted clique to get lost in the crowd. How to argue this more formally?

$$TV(P,Q) \le \sqrt{\frac{KL(P\|Q)}{2}} \le \sqrt{\frac{\log(1+\chi^2(P,Q))}{2}} \le \sqrt{\frac{\chi^2(P,Q)}{2}}$$

As we saw in HW1 Q2d, chi-square is useful to analyse mixture distributions.

$$\chi^2(G(n,1/2,k), G(n,1/2)) = \mathbb{E}_{K,K'}\left[\sum_{g} \frac{Prob_K(g)Prob_{K'}(g)}{Prob_{Null}(g)}\right] - 1$$

Analyse, bound binomial coefficients, can see that it is impossible to distinguish for k much smaller than $(2-\epsilon)\log n$.

Efficient algorithm?

Idea: What is k is very large?
Idea: How many edges does each graph have?

- Number of edges in $G(n, 1/2, k) \geq \frac{\binom{n}{2}}{2} + \Omega(k^2)$.
- Number of edges in $G(n, 1/2) = \frac{\binom{n}{2}}{2} \pm \Theta(n)$.

If $k \gg \sqrt{n}$, simply counting the edges in the graph (which takes linear time) is good enough.
Is there an efficient algorithm that does better?

What about a spectral algorithm?

Incorporates far more global / long range information than simply edge density, which just looks in a very local neighbourhood. Can it do better? Let A denote the centered adjacency matrix of the graph.

- $A \sim G(n, 1/2, k)$. There exists $x \in \mathbb{R}^n$ such that $\frac{x^T A x}{x^T x} \geq \Omega(k)$.
- $A \sim G(n, 1/2)$. Whp, largest eigenvalue is A are $\Theta(\sqrt{n})$.

The same $\sqrt{n}$ threshold shows up again mysteriously. Interestingly, this algorithm works to recover the planted clique too.

Current state of the art for planted clique

1. $k \ll 2\log n$: Problem is statistically unsolvable.
2. $2\log n \ll k \ll \sqrt{n}$: No known efficient polynomial time algorithm. Known $n^{O(\log n)}$ time algorithm.
3. $\sqrt{n} \ll k$: Efficient algorithm

- Many far more sophisticated algorithmic techniques break down at the same $\Theta(\sqrt{n})$ threshold. $:-($
- Arose from trying to make random graph clique finding easier.
- 3 regimes. Prototypical example of statistical-computational gap. Many more examples.
- How to explain this situation? What tools do we have? Why do all algorithms break down at same threshold? Is it just a coincidence?

Questions?

Example 2: Sparse PCA / sparse spiked covariance model

PCA is one of the most widely used dimension reduction techniques. What if you had additional knowledge about your data?
Data: $X = X_1, X_2, ..., X_n \in \mathbb{R}^d$. v is a k-sparse unit vector, θ is SNR.

$$H_0 : X \sim N(0, I_d)^{\otimes n} \text{ and } H_1 : X \sim N(0, I_d + \theta v v^T)^{\otimes n}$$

- IT limit: $\theta \gg \sqrt{\frac{k \log d}{n}}$.
- Known efficient algorithms require: $\theta \gg \sqrt{\frac{k^2}{n}}$
- Simply adding additional knowledge like sparsity can introduce a stat-comp gap. We do not know how to efficiently exploit this extra structure.

Example 3: Robust Sparse Mean Estimation

Gaussian mean estimation, but

1. Mean is known to be sparse.
2. Robustness is desired. Some data points may be adversarially corrupted.

$X_1, \ldots, X_n$ are n iid samples in d dimensions distributed as $\mathcal{N}(\mu, I_d)$.
μ has at most k non-zero entries. ϵn samples are adversarially corrupted for some $0 < \epsilon < 1$.
IT threshold: $n \gg \frac{k \log d}{\epsilon^2}$.
Computational threshold: $n \gg \frac{k^2 \log d}{\epsilon^2}$ (alg is much more complicated than the simple edge counting test that worked for Planted Clique)

- Robustness requirement can turn gapless problems into problems with a stat-comp gap. Empirical mean stops working, even though it worked with sparsity.
- This is different from previous example.
- In both examples though, it seems we need some combinatorial search to do things statistically optimally.

Example 4: Planted Dense Submatrix detection

Trying to capture dense communities in networks of individuals. A is a $n \times n$ symmetric matrix with 0 on the diagonals. Let K be a random subset of $[n]$ of size k. $A_{ij} \sim P$ if $i, j \in K$. $A_{ij} \in Q$ otherwise.

1. Bernoulli version: $P = Bern(p)$ and $Q = Bern(q)$ with $p = 2q$.
2. Gaussian version: $P = N(\mu, 1)$ and $Q = N(0, 1)$ with $\mu > 0$.

Note how this is slightly similar to PC. Let us further parametrize $k = n^{\alpha}$ and $\mu = \Theta(\sqrt{q}) = n^{-\beta}$.

- 2 parameters govern the easiness / hardness, rather than just 1. Gives rise to a more complicated landscape.
- IT threshold: $\beta \leq \max\{\frac{\alpha}{2}, 2\alpha - 1\}$
- Efficient algorithms threshold: $\beta \leq \max\{0, 2\alpha - 1\}$

What tools do we have to understand computational aspects of statistical problems?

Purely statistical tools fail to always predict computational thresholds.

1. Analysis against restricted classes of algorithms.
2. **Efficient** reductions to other problems believed to be hard (bread and butter of complexity theory).
3. (1.) & (2.): reductions between algorithmic classes. (Tselil Schramm stats dept)

Restricted algorithmic classes

Consider the planted clique problem. All the below listed classes of algorithms provably fail to give polynomial running times for $k = o(\sqrt{n})$.

- Markov chain Monte Carlo (sampling technique) [Jerrum 92]
- Lovasz-Schrijver hierarchy [Feige Krauthgamer 03]
- Sum of Squares hierarchy [Barak et al. 16]
- Statistical Query algorithms [Feldman et al. 17]

Key hope – these classes should capture all / most of our known algorithmic ideas. This builds up evidence that the problem is indeed hard.

Eg: Low Degree Polynomials

Input: $X_1, X_2, ..., X_n$.
Class of algorithms / tests: All multivariate polynomials of the input of "low degree".

- Captures both the edge counting test for PC, the gaussian mean estimation algorithm.
- Computing larger degree polynomials intuitively aligns with more computational resources required.

Let p be **any** such low degree polynomial.

$$H_0 : p(G(n, 1/2)) \text{ vs } H_1 : p(G(n, 1/2, k)).$$

We can now use well developed theory of statistical lower bounds to reason about this class of algorithms. Can use TV distance, other divergences to reason about hardness of this statistical problem.

Reductions - workhorse of worst-case complexity theory

A **reduction** is an algorithm for transforming one problem into another problem. A sufficiently efficient reduction from one problem to another may be used to show that the second problem is at least as difficult as the first.

Squaring vs Multiplying integers.

If we can multiply two numbers, we can easily square a number. Is it possible that squaring is “easier” than multiplying?

$$a \times b = \frac{(a+b)^2 - a^2 - b^2}{2}$$

Usually, we care about reductions between hard problems, not easy ones.

Reductions - workhorse of worst-case complexity theory

A **reduction** is an algorithm for transforming one problem into another problem. A sufficiently **efficient** reduction from one problem to another may be used to show that the second problem is at least as difficult as the first.

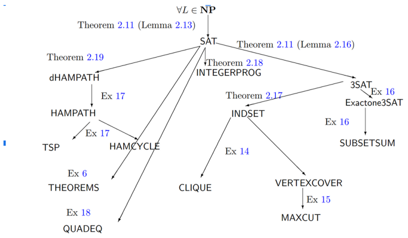
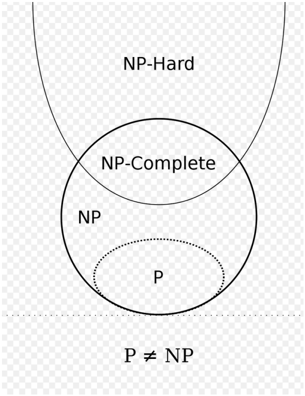


Figure 2.4: Web of reductions between the **NP**-completeness problems described in this chapter and the exercises. Thousands more are known.



Images from https://users.cs.duke.edu/ reif/courses/complectures/books/AB/NPchap.pdf, Wikipedia

Allows classification of problems. Can we do this for statistical problems?

Reductions - some remarks

- Reductions are so useful because they let us make progress even when we are ignorant. Very few unconditional computational hardness results, but a very rich conjectural landscape exists.
- Reductions between statistical problems are much more delicate and difficult to pull off as compared to (traditional) worst-case complexity.

Can we create a web of reductions for statistical problems? What, precisely, would such a thing even mean?
- **Efficient** reduction in **total variation distance**

Comparison of Models

For two statistical models $\mathcal{M} = (\mathcal{X}, (P_\theta)_{\theta\in\Theta})$ and $\mathcal{N} = (\mathcal{Y}, (Q_\theta)_{\theta\in\Theta})$, when can we say model $\mathcal{M}$ is stronger than model $\mathcal{N}$? How can we translate a solution to model $\mathcal{N}$ to a solution to model $\mathcal{M}$?

Easy direction: randomization $\Rightarrow$ deficiency

Theorem

Model $\mathcal{M}$ is ε-deficient relative to $\mathcal{N}$ if (**don't need other direction**) there exists a stochastic kernel $\mathsf{K} : \mathcal{X} \to \mathcal{Y}$ such that

$$\sup_{\theta \in \Theta} \|Q_\theta - \mathsf{K} P_\theta\|_{\mathsf{TV}} \le \varepsilon.$$

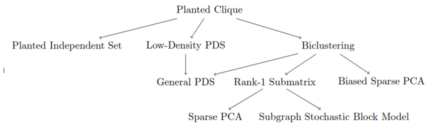
- choose any action space $\mathcal{A}$ and loss L.
- fix any **poly-time computable** decision rule $\delta_{\mathcal{N}}$ for model $\mathcal{N}$.
- choose $\delta_{\mathcal{M}} = \delta_{\mathcal{N}} \circ \mathsf{K}$. If K is **poly-time computable**, $\delta_{\mathcal{M}}$ is **poly-time computable**

$$\begin{aligned}
&R_\theta(\delta_{\mathcal{M}}) - R_\theta(\delta_{\mathcal{N}}) \\
&= \mathbb{E}_{a \sim \delta_{\mathcal{M}}(\cdot|X)} \mathbb{E}_{X \sim P_\theta}[L(\theta, a)] - \mathbb{E}_{a \sim \delta_{\mathcal{N}}(\cdot|Y)} \mathbb{E}_{Y \sim Q_\theta}[L(\theta, a)] \\
&= \mathbb{E}_{a \sim \delta_{\mathcal{N}}(\cdot|Y)} \mathbb{E}_{Y \sim \mathsf{K} P_\theta}[L(\theta, a)] - \mathbb{E}_{a \sim \delta_{\mathcal{N}}(\cdot|Y)} \mathbb{E}_{Y \sim Q_\theta}[L(\theta, a)] \\
&= \mathbb{E}_{a \sim \delta_{\mathcal{N}}(\cdot|Y)} (\mathbb{E}_{Y \sim \mathsf{K} P_\theta}[L(\theta, a)] - \mathbb{E}_{Y \sim Q_\theta}[L(\theta, a)]) \\
&\le \|Q_\theta - \mathsf{K} P_\theta\|_{\mathsf{TV}}
\end{aligned}$$

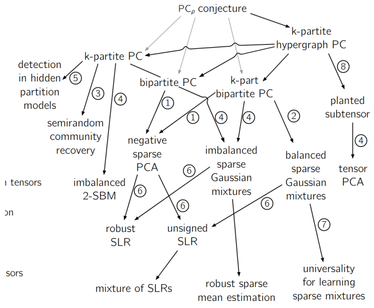
Web of average case reductions. Images from BBH18, BB20. Many other reductions are also known.

1. Show evidence that Planted Clique is hard by studying restricted but powerful classes of algorithms.
2. Show other problems are hard by giving **efficient** reductions in **total variation distance** from Planted Clique in the hard regime.

Planted Clique has emerged as central problem where such a strategy has been carried out.



igure 1: Graph of average-case reductions for detection problems showing tight statistical-computational aps given the planted clique conjecture.



Recall: Planted Dense Submatrix detection

Trying to capture dense communities in networks of individuals. A is a $n \times n$ symmetric matrix with 0 on the diagonals. Let K be a random subset of $[n]$ of size k. $A_{ij} \sim P$ if $i, j \in K$. $A_{ij} \in Q$ otherwise.

1. Bernoulli version: $P = Bern(p)$ and $Q = Bern(q)$ with $p = 2q$.
2. Gaussian version: $P = N(\mu, 1)$ and $Q = N(0, 1)$ with $\mu > 0$.

Note how this is slightly similar to PC. Let us further parametrize $k = n^{\alpha}$ and $\mu = \Theta(\sqrt{q}) = n^{-\beta}$.

- 2 parameters govern the easiness / hardness, rather than just 1. Gives rise to a more complicated landscape.
- IT threshold: $\beta \leq \max\{\frac{\alpha}{2}, 2\alpha - 1\}$
- Efficient algorithms threshold: $\beta \leq \max\{0, 2\alpha - 1\}$

Zooming in on Planted Dense Submatrix

Image from WX18.

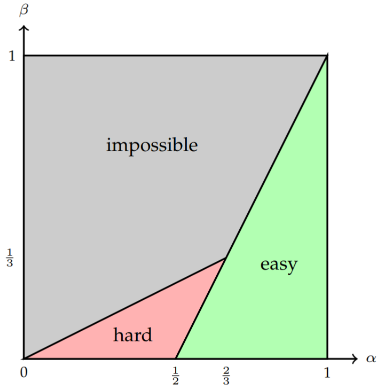


Figure 1: Computational versus statistical limits. For the submatrix detection problem, the size of the submatrix is $K = N^{\alpha}$ and the elevated mean is $\mu = N^{-\beta}$. For the community detection problem, the cluster size is $K = N^{\alpha}$, and the in-cluster and inter-cluster edge probability p and q are both on the order of $N^{-2\beta}$.

Questions?

Reduction from PC to PDS

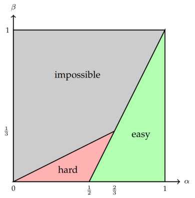


Figure 1: Computational versus statistical limits. For the submatrix detection problem, the size of the submatrix is $K = N^{\alpha}$ and the elevated mean is $\mu = N^{-\beta}$. For the community detection problem, the cluster size is $K = N^{\alpha}$, and the in-cluster and inter-cluster edge probability p and q are both on the order of $N^{-2\beta}$.

- We take assumptions on lower line and transfer it to fill out the whole picture.
- Trivial to get inefficient reduction in hard regime simply by solving the problem.
- Today we restrict attention to scalar reduction, PC-edge to Bernoulli, and PC-edge to Gauss. This is a powerful and widely applicable primitive.

Technical remark

What is the bit complexity of Gaussian PDS? how to define polynomial time?

Recall:
For two sequences of statistical models $\mathcal{M}_n = (\mathcal{X}_n, (P_{n,\theta})_{\theta \in \Theta_n})$ and $\mathcal{N}_n = (\mathcal{Y}_n, (Q_{n,\theta})_{\theta \in \Theta_n})$ with $\Theta_n \uparrow \Theta$, we say they are asymptotically equivalent if and only if

$$\Delta(\mathcal{M}_n, \mathcal{N}_n) \to 0, \qquad \text{as } n \to \infty.$$

- $\mathcal{M}_n$: Gaussian PDS with $n \times n$ matrix.
- $\mathcal{N}_n$: Gaussian PDS with $n \times n$ matrix where each entry is discretized to $\Theta(\log n)$ bits.
  Show asymptotic equivalence. We will ignore this.

$Bern(1/2) \to Bern(\ell)$ and $Bern(1) \to Bern(2\ell)$

Key difficulty: We do not know which distribution our input is from.

Algorithm:

- If $B = 1$, output $Z \sim Bern(2\ell)$.
- If $B = 0$, output $Z = 0$.

Analysis:

- Clearly if $B \sim Bern(1)$, $Z \sim Bern(2\ell)$.
- If $B \sim Bern(1/2)$, $Prob(Z = 1) = \frac{1}{2} \times 2\ell = \ell$. So $Z \sim Bern(\ell)$.

This example was very clean. Unfortunately, most things in life are not so clean.

Attempt 1: $Bern(1/2) \to N(0,1)$ and $Bern(1) \to N(\mu,1)$

Same idea as before -
Algorithm:

- If $B = 1$, output $Z \sim N(\mu, 1)$.
- If $B = 0$, output ???.

Working backwards, want to output a distribution whose pdf if $2N(0,1) - N(\mu,1)$.

- Problem: pdf thus produced has negative values.
- Exact mapping impossible.
- Idea: Reduce approximately in TV distance.

Attempt 2: $Bern(1/2) \to N(0,1)$ and $Bern(1) \to N(\mu,1)$

Rejection sampling (assume μ is appropriately small) -
Algorithm:

- If $B = 1$, output $Z \sim N(\mu, 1)$.
- If $B = 0$, sample $Z \sim N(0,1)$. With probability $\max\{0, 1 - \frac{N(\mu,1)(Z)}{2N(0,1)(Z)}\}$, output Z. Otherwise repeat, for a maximum of T times.

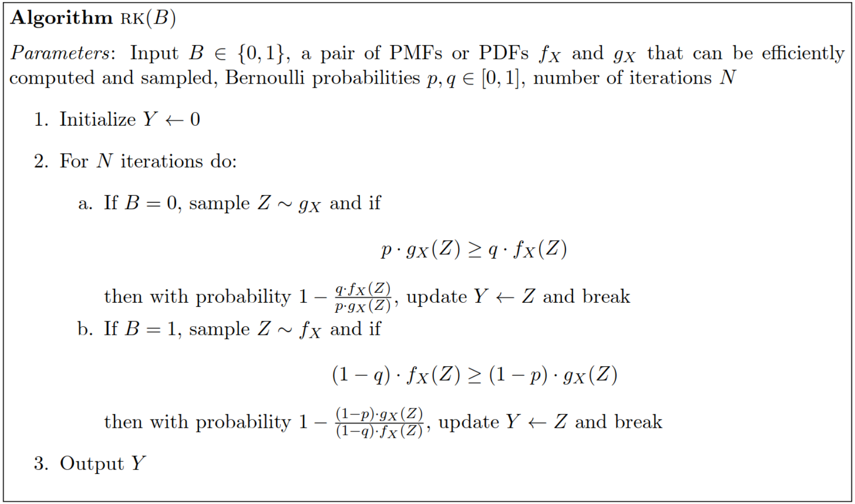
Procedure is well defined, unlike on previous slide.
Assume: $T = +\infty$ and $2N(0,1)(Z) - N(\mu,1)(Z) > 0$ for all Z.
Then clearly for $B \sim Bern(1/2)$, output $\sim N(0,1)$.

- Maximum of T tries is for computational efficiency. Need finite runtime. Setting $T = \Theta(\log(\frac{1}{\epsilon}))$ is good enough to guarantee termination except with $O(\epsilon)$ probability.
- If μ is appropriately small, $Prob(2N(0,1)(Z) - N(\mu,1)(Z) > 0)$ is close to 1.

These two errors control the TV approximation error, and we can show it is small.

True in much more general way: BBH19

$Bern(p) \to f_X$ and $Bern(q) \to g_X$. Very useful primitive.



```
Algorithm RK($B$)

Parameters: Input $B \in \{0,1\}$, a pair of PMFs or PDFs $f_X$ and $g_X$ that can be efficiently
computed and sampled, Bernoulli probabilities $p, q \in [0,1]$, number of iterations $N$

  1. Initialize $Y \leftarrow 0$

  2. For $N$ iterations do:

       a. If $B = 0$, sample $Z \sim g_X$ and if

                         $p \cdot g_X(Z) \geq q \cdot f_X(Z)$

          then with probability $1 - \frac{q \cdot f_X(Z)}{p \cdot g_X(Z)}$, update $Y \leftarrow Z$ and break
       b. If $B = 1$, sample $Z \sim f_X$ and if

                         $(1-q) \cdot f_X(Z) \geq (1-p) \cdot g_X(Z)$

          then with probability $1 - \frac{(1-p) \cdot g_X(Z)}{(1-q) \cdot f_X(Z)}$, update $Y \leftarrow Z$ and break

  3. Output $Y$
```

Figure 7: Rejection kernel in Lemma 5.1

Game Theory: How hard is it to approximate the best Nash equilibrium? (Hazan, Krauthgamer)

Definition (2-player non-cooperative game)

- Row player and column players
- Two non-negative payoff matrices $R, C \in \mathbb{R}^{N \times N}$ (entries bounded in $[0, M]$).
- Row player plays strategy $i \in [N]$ and column player plays $j \in [N]$. Row player wins $R[i,j]$, column player wins $C[i,j]$, total payoff is $R[i,j] + C[i,j]$.

Given R, C how should the players play to maximize their own payoff?

Definition (Nash equilibrium)

There always exists a **mixed** strategy (r,c), called a Nash equilibrium-

- For any other strategy x, $x^T Rc \leq r^T Rc$.
- For any other strategy y, $r^T Ry \leq r^T Rc$.

Finding Nash eqb is hard. What about appx Nash equilibria?

Finding a Nash equilibrium is known to be PPAD-complete. Can we at hope to efficiently approximate it?

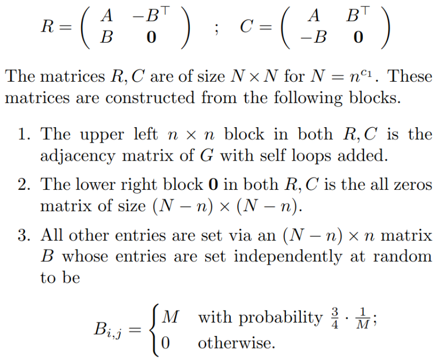
Definition (ϵ-appx Nash equilibrium)

There always exists a **mixed** strategy (r,c), called a Nash equilibrium-

- For any other strategy x, $x^T Rc \leq r^T Rc + \epsilon$.
- For any other strategy y, $r^T Ry \leq r^T Rc + \epsilon$.

- Known $N^{O(\frac{\log N}{\epsilon^2})}$ time algorithm, but no known poly time algorithm.
- Worst case problem, but no known reduction from worst-case hard problems.
- HK show hardness of a related problem. Assuming PC **recovery** hardness, no poly time algorithm for ϵ-**optimal** ϵ-appx Nash equilibrium.
- Quasipoly algorithm also exists for this problem, which reminds us of PC in hard regime.

The reduction (Image from Hazan Krauthgamer)



$$R = \begin{pmatrix} A & -B^\top \\ B & \mathbf{0} \end{pmatrix} \quad ; \quad C = \begin{pmatrix} A & B^\top \\ -B & \mathbf{0} \end{pmatrix}$$

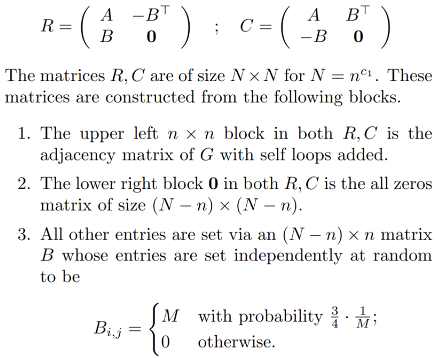
The matrices R, C are of size $N \times N$ for $N = n^{c_1}$. These matrices are constructed from the following blocks.

1. The upper left $n \times n$ block in both R, C is the adjacency matrix of G with self loops added.
2. The lower right block $\mathbf{0}$ in both R, C is the all zeros matrix of size $(N-n) \times (N-n)$.
3. All other entries are set via an $(N-n) \times n$ matrix B whose entries are set independently at random to be

$$B_{i,j} = \begin{cases} M & \text{with probability } \frac{3}{4} \cdot \frac{1}{M}; \\ 0 & \text{otherwise.} \end{cases}$$

- Outside first n strategies, it is a zero sum game. Only these strategies contribute to total payoff.
- $A \sim G(n, 1/2, k)$ with $2 \log n \ll k \ll \sqrt{n}$.

Continued: The reduction (Image from Hazan Krauthgamer)



$$R = \begin{pmatrix} A & -B^\top \\ B & \mathbf{0} \end{pmatrix} \quad ; \quad C = \begin{pmatrix} A & B^\top \\ -B & \mathbf{0} \end{pmatrix}$$

The matrices R, C are of size $N \times N$ for $N = n^{c_1}$. These matrices are constructed from the following blocks.

1. The upper left $n \times n$ block in both R, C is the adjacency matrix of G with self loops added.
2. The lower right block $\mathbf{0}$ in both R, C is the all zeros matrix of size $(N-n) \times (N-n)$.
3. All other entries are set via an $(N-n) \times n$ matrix B whose entries are set independently at random to be

$$B_{i,j} = \begin{cases} M & \text{with probability } \frac{3}{4} \cdot \frac{1}{M}; \\ 0 & \text{otherwise.} \end{cases}$$

- There is an equilibrium with total payoff 2, which is maximum possible.
- Given any appx equilibrium with large value, can find an appx eqb with large value supported on $[n]$.
- This appx eqb contains info about planted clique.

Summary

1. Statistical and computational notions of tractability differ.
2. Statistical-computational gaps are a relatively widespread phenomenon. They can arise in different ways.
3. Need new tools to study computational tractability of statistical problems.
4. Even with these new tools, the lower bound ideas from this course are vital.

Key ideas:

- Analysing restricted classes of algorithms (like low degree polynomials).
- Reductions

Questions?

References

- BBH18: Reducibility and Computational Lower Bounds for Problems withPlanted Sparse Structure - Brennan, Bresler, Huleihel
- BB20: Reducibility and Statistical-Computational Gaps from Secret Leakage - Brennan, Bresler
- HK: How hard is it to approximate the best Nash equilibrium? - Hazan, Krauthgamer
- Lug: Lectures on Combinatorial Statistics - Lugosi
- WX18: Statistical Problems with Planted Structures:Information-Theoretical and Computational Limits - Wu, Xu

Next lecture: Le Cam's two-point method