EE378C Information-theoretic Lower Bounds in Data Science Lecture 3 - 04/14/2021

Lecture 6: Statistical vs Computational notions of Tractability

Lecturer: Jay Mardia *Scribe: Ryan Tolsma, David Lin*

In this course, we mainly deal with understanding the fundamental *statistical* limits on inference from data. However, there are other fundamental limits too.

In this lecture we introduce examples where the statistical complexity versus algorithmic tractability of solving a particular problem differ. As a primary example, we examine the Planted Clique problem which exhibits this phenomena and use it to introduce the notion of efficient reductions within complexity theory.

Is it possible that statistical inference might be possible, but not computationally efficient? A place where computational intractability shows up and plays a critical role is the RSA cryptographic protocol, which relies on the practical difficulty of factoring products of two large primes to keep messages secret. From the standpoint of information (and thus statistics), the product of two primes also contain all the information about its two prime factors (in the sense that one can eventually compute it), but it is believed that there is no computationally efficient method to recover that information and thus provides cryptographic security for all its applications.

In this lecture we will see some examples of problems where computational and statistical notions of tractability do not coincide. We will study tools we can use to try and understand / predict computational tractability, analogous to how the tools we learn in this course predict statistical tractability.

1 Basic Notions

In order to reason about computational complexity, we first need to agree on what we mean by computation. A very high level definition that will suffice for our purposes is below.

Computation is a finitely describable set of instructions that takes in an arbitrarily large input and produces an answer. The number of steps (or instructions) that a computation uses is called its **time complexity**.

Different theoretical models of computation such as Turing Machines and Finite Automata attempt to realistically model the functionality of computers in different contexts. The notion of computational efficiency is critical in the analysis of any algorithm.

Definition 1 (Informal)**.**
We call an algorithm ***efficient*** *if its time complexity is at most poly*(n) *for any the input of size* n*. A computational problem is* ***computationally tractable*** *if there exists an efficient algorithm that solves it.*

Equating computational efficiency with polynomial time is reasonable for multiple reasons: it empirically coincides with the practical concept of an efficient algorithm, it is agnostic to precise low-level changes in the computational model (e.g. whether we use Turing Machines vs lines of C) and finally efficiency is preserved under compositions of efficient algorithms (i.e. calling another efficient algorithm as a subroutine).

For statistical problem setups, which often exhibit a contextually clearer notion of signal and noise, we may talk about a similar (shorthand) concept of tractability that is less general but easier to use than the concept of deficiency we have seen earlier in this course.

We call a statistical problem **statistically tractable** if the original signal can be recovered there is a sufficiently high signal to noise ratio (SNR).

We will be interested in studying how the existence of computationally efficient algorithms changes with SNR.

2 Statistical-Computational gaps

The Planted Clique is a hallmark statistical problem which characterizes a divergence between the notions of statistical and computational tractability.

2.1 Problem setup

For a graph G, consider the hypothesis testing problem where we would like to differentiate between hypotheses

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

Here, $G(n, \frac{1}{2})$ is the distribution of an Erdös-Rényi random graph with the edge between any two vertices being present independently with probability $1/2$. $G(n, \frac{1}{2}, K)$ is similar, except that for a subset K of vertices of size k, K is forced to be a clique (and other edges are drawn as per the usual Erdös-Rényi distribution). By interpreting the edges from K as the signal and the other edges as noise, it is evident that the SNR increases as k increases, and the problem should become statistically more tractable.

2.2 Statistical Tractability

Intuitively, if $G(n, 1/2)$ is expected to contain many cliques of size k, then the clique K will be "lost in the crowd" and we will not be able to distinguish whether there is a planted clique. A linearity of expectation argument gives that

$$(\#\text{ of cliques of size } m \text{ in } G(n, 1/2)) = \frac{\binom{n}{m}}{2^{\binom{m}{2}}} \tag{3}$$

and this is $o(1)$ when $m \geq (2 + \epsilon) \log n$. Thus, for $k \geq (2 + \epsilon) \log n$, we expect to be able to recover the planted clique simply by enumerating over all subsets of vertices of size k, and to answer based on whether we found a clique of size k. However, this combinatorial brute-force is too slow to be considered efficient, with a runtime of $n^{O(\log n)}$.

On the other hand, if $k \leq (2-\epsilon) \log n$, we can bound the TV distance using χ^2 divergence, and a computation yields that the TV distance is $o(1)$, and thus we will not be able to distinguish between the two hypotheses.

2.3 Computational Tractability

On the other hand, another way one might distinguish between the two hypotheses is to count the number of edges. Indeed, we have

$$(\#\text{ edges in } G \sim G(n, 1/2)) = \frac{1}{2}\binom{n}{2} \pm \Theta(n) \qquad \text{w.h.p.} \tag{4}$$

$$(\#\text{ edges in } G \sim G(n, 1/2, K)) = \frac{1}{2}\binom{n}{2} + \frac{1}{2}\binom{k}{2} \pm \Theta(n) \qquad \text{w.h.p.} \tag{5}$$

hence if $k \gg \sqrt{n}$, the number of edges differentiates between the two hypotheses. It is worth noting that more advanced techniques (e.g. spectral analysis, convex programming relaxations) are also able to recover the subset K in this regime, but not for smaller k.

2.4 The Conjecturally Hard Regime

In the range $2\log(n) \ll k \ll \sqrt{n}$, there is no known polynomial time solution (or even a solution with time complexity better than $n^{O(\log n)}$). The same $\sqrt{n}$ threshold shows up for a number of different techniques (e.g. spectral analysis). This is a classic example of a conjectural **statistical-computational gap**.

2.5 Other examples

Here we describe two other instances of a conjectured statistical-computational gap.

Sparse PCA. This problem is about distinguishing samples drawn from an isotropic (zero-mean) Gaussian against a similar distribution with a sparse spike in the covariance. Let $X = X_1, \cdots, X_n$ be i.i.d. d-dimensional vectors drawn from one of two distributions:

$$H_0 : X \sim \mathcal{N}(0, I_d)^{\otimes n} \tag{6}$$

$$H_1 : X \sim \mathcal{N}(0, I_d + \theta vv^\top)^{\otimes n} \tag{7}$$

where v is a k-sparse unit vector (i.e. only k coordinates are non-zero).

The threshold for statistical tractability is $\theta \gg \sqrt{\frac{k \log d}{n}}$, while any (known) efficient algorithms requires $\theta \gg \sqrt{\frac{k^2}{n}}$.

Robust Sparse Mean Estimation. This is the Gaussian mean estimation problem with two caveats: the mean is sparse, and some data points may be adversarially corrupted. Let $X_1, \cdots, X_n$ be n i.i.d. d-dimensional vectors drawn from $\mathcal{N}(\mu, I_d)$, where μ is k-sparse. Furthermore, up to ϵn many samples may be adversarially corrupted for some $0 < \epsilon < 1$.

The threshold for statistical tractability is $n \gg \frac{k \log d}{\epsilon^2}$, while known efficient algorithms only exist for $n \gg \frac{k^2 \log d}{\epsilon^2}$. In this case, the efficient algorithm is much more complicated than that of the planted clique problem.

A common theme in both problems is that the statistically optimal settings require us to do some kind of combinatorial search, and this usually arises when we require sparsity or robustness.

3 Tools to understand statistical-computational gaps?

There are two broad ideas that are used to reason about computational tractability of statistical problems.

1. Show the failure of restricted but powerful classes of algorithms.
2. Reductions between statistical problems.

3.1 Hardness of Planted Clique

How do we know that the Planted Clique problem is computationally hard? We don't know for sure (yet!), but there is much evidence pointing in that direction. Computational hardness is known when we restrict our analysis to various classes of algorithms, such as

- Markov chain Monte Carlo sampling
- Lovasz-Schrijver hierarchy
- Sum of Squares hierarchy
- Statistical Query algorithms

All the above classes of algorithms provably fail to give polynomial running times for $k = o(\sqrt{n})$!

The statistical tools we develop in this course are crucial when reasoning about these restricted algorithmic classes.

3.2 Reduction between Statistical Models

There is an analogy between statistical problems in regimes which are statistically tractable but (conjecturally) computationally hard (i.e. unsolvable by efficient algorithms), and NP-hard problems, which are deterministic problems conjectured to be computationally hard. In both cases, there are very few known unconditional computational hardness results, but a rich conjectural landscape exists because we can "reduce" most problems to a single canonical (conjecturally) hard problem.

How might we define reduction for statistical models? We recall one possible definition of ε-deficiency via randomization:

Definition 2. *Model $\mathcal{M} = (\mathcal{X}, (P_\theta)_{\theta\in\Theta})$ is ε-**deficient** relative to $\mathcal{N} = (\mathcal{Y}, (Q_\theta)_{\theta\in\Theta}$ if there exists a stochastic kernel $\mathsf{K} : \mathcal{X} \to \mathcal{Y}$ such that*

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

By an easy computation, the stochastic kernel gives us a way to turn a decision rule $\delta_{\mathcal{N}}$ for model $\mathcal{N}$ into one for model $\mathcal{M}$ by setting $\delta_{\mathcal{M}} = \delta_{\mathcal{N}} \circ \mathsf{K}$. To include computational efficiency into the picture, we can further require that the stochastic kernel K is poly-time computable. Then, we are able to turn poly-time computable decision rules for model $\mathcal{N}$ into that for model $\mathcal{M}$.

In the past decade, many statistical problems have been shown to be computationally hard if the planted clique problem is computationally hard below the $k = o(\sqrt{n})$ clique size threshold. In these applications, it is often crucial that we only need the reduction (the kernel) to map between distributions *approximately* in *total variation distance*. So we see that TV distance, which is the central protagonist in this course, also plays an important role even in our understanding of computational tractability.

3.3 Example: Planted Dense Submatrix Detection

Here, we *partially* describe the key idea in one example of a computationally efficient reduction.

Let A be an $n \times n$ symmetric matrix with 0 along its diagonal entries. Let $K \subset [n]$ be a random subset of size k, with A_{ij} being drawn from two distributions

$$A_{ij} \sim \begin{cases} \mathrm{Bern}(p) & \text{if } i,j \in K \\ \mathrm{Bern}(2p) & \text{otherwise} \end{cases} \qquad \text{independently} \tag{8}$$

The planted clique problem is the case $p = 1/2$. Can we reduce from the planted clique problem to this more general problem? We focus on the key subroutine, which looks only at a scalar (rather than matrix) version of this problem.

For general $0 < p \leq 1/2$, we can perform an efficient reduction if there was some efficient (stochastic) kernel f where

$$B \sim \text{Bern}(1/2) \to f(B) \sim \text{Bern}(p)$$
$$B \sim \text{Bern}(1) \to f(B) \sim \text{Bern}(2p).$$

The key challenge is that we do not know which distribution B is drawn from initially, and our function must be agnostic to this. One possible algorithm that works is

$$f(B) \begin{cases} \sim \text{Bern}(2p) & \text{if } B = 1 \\ = 0 & \text{if } B = 0 \end{cases} \tag{9}$$

However, if we were to apply this method to the Gaussian version of this problem, where

$$A_{ij} \sim \begin{cases} \mathcal{N}(\mu, 1) & \text{if } i, j \in K \\ \mathcal{N}(0, 1) & \text{otherwise} \end{cases} \qquad \text{independently} \tag{10}$$

then the above reduction fails, because we would require a distribution whose pdf is $2\mathcal{N}(0,1) - \mathcal{N}(\mu, 1)$, which has negative values, and hence cannot be the pdf of a distribution. This challenge can be overcome using rejection sampling, which is poly-time computable if we allow some small fixed probability of error. This technique also applies more generally where we wish to transform $(\text{Bern}(p), \text{Bern}(q))$ into any pair of distributions (f_X, g_X).