Computational Complexity

Lecture 13: Meta Complexity

Ronald de Haan
me@ronalddehaan.eu

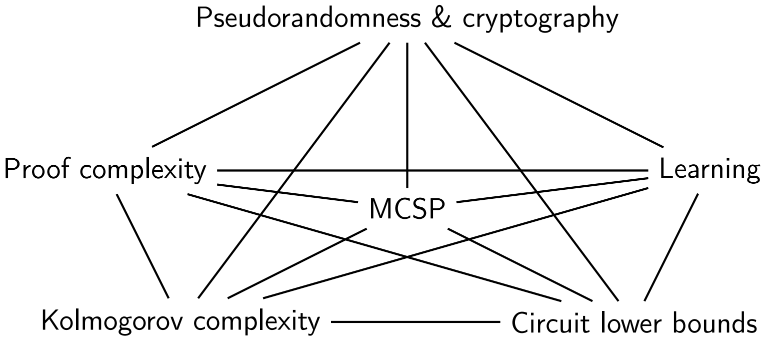
University of Amsterdam

May 23, 2024

What is meta complexity?

- Meta complexity is an informal term referring to the computational complexity study of problems that have a ‘complexity flavor’
- So in a sense, meta complexity studies the complexity of complexity problems (hence the phrase ‘meta’)
- This turns out to be fruitful for studying various notions related to computational complexity, learning, cryptography, etc.

Meta complexity connects



The Minimum Circuit Size Problem

The Minimum Circuit Size Problem (MCSP)

- MCSP:
  - *Input:* a Boolean function F over n variables given by its truth table (containing 2^n entries), and a positive integer $s \in \mathbb{N}$ (given in binary).
  - *Question:* does there exist a Boolean circuit C of size s that expresses the function F?

- MCSP[s], for a function $s : \mathbb{N} \to \mathbb{N}$:
  - *Input:* a Boolean function F over n variables given by its truth table (containing 2^n entries).
  - *Question:* does there exist a Boolean circuit C of size $s(2^n)$ that expresses the function F?

Black-box problem

- Intuitively, MCSP is a black-box problem:
  - We are given the input-output behavior of a function F
  - The task is to see if this function F has small circuits

- Compare this to white-box problems such as SAT, where we are given an explicit way to compute the Boolean function F about which we are answering a question—namely, by means of a formula or circuit

MCSP is in NP

- MCSP is in NP
- Might seem odd at first:
  - Circuits to consider are exponentially large in the size of (the binary encoding of) s
- Main idea:
  - There is always a circuit for F of size $O(2^n)$
  - We are given the truth table of F as input, which is of size 2^n
  - So we can guess a circuit C of size at most $O(2^n)$ in polynomial time
  - And check if C expresses F by iterating over all rows α in the truth table, and checking if $C(\alpha) = F(\alpha)$

Open question: is MCSP in P? Is it NP-complete?

- One main open research question:

Is MCSP **NP-complete?**

- MCSP is not in P assuming OWFs exist.
  - *(Connection via natural properties, which we will see later..)*

Circuit lower bounds

Circuit lower bounds

- One approach to trying to show $\mathsf{P} \neq \mathsf{NP}$ is by giving circuit lower bounds
- Circuit lower bounds for a class $\mathcal{C}$ of circuits: showing that there is a function f that does not have small circuits within $\mathcal{C}$
- For example: Parity (computing whether a string has an even number of 1's) is not in AC^0 (the class of polynomial-size constant-depth circuits), but it is in $\mathsf{AC}^0[2]$
- The idea would be to do this for ever more expressive classes of circuits, leading to $\mathsf{NP} \not\subseteq \mathsf{P/poly}$, which implies $\mathsf{P} \neq \mathsf{NP}$

Circuit lower bounds (in a 'picture')

- After some initial successes, this program stalled in the 1980s
- State of results:

$$\mathsf{AC}^0 \subsetneq \mathsf{AC}^0[p] \subsetneq \mathsf{ACC}^0 \subseteq \mathsf{TC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{AC}^1 \subseteq \mathsf{NC}^2 \subseteq \mathsf{P} \subseteq \mathsf{NP}.$$

- A 2011 paper by Ryan Williams that won the 2024 Gödel Prize showed:

$$\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0.$$

A connection between MCSP and circuit lower bounds

- The following two are equivalent:
  - Showing that DTIME($2^{O(n)}$) does not have Boolean circuits of size $s(n)$
  - Efficiently (in polynomial time) constructing no-instances of MCSP[s']—where $s' = s \circ \log$—of size 2^n,given 2^n in unary.
- Main idea:
  - Suppose there is a problem L in DTIME($2^{O(n)}$) that has no circuits of size $s(n)$.
    Using this, we can compute in time $2^{O(n)} = \text{poly}(2^n)$ the truth table of problem L on inputs of size n.
    This is a no-instance of MCSP[s'] of size 2^n.
  - Suppose you can efficiently construct no-instances of MCSP[s'] of size 2^n.
    Using this, for each input size, we can construct (in exponential time) a truth table of a Boolean function that has no circuits of size $s(n)$.
    This yields a problem in DTIME($2^{O(n)}$) that has no circuits of size $s(n)$.

Natural proofs barrier

- Combinatorial property: $\{P_n\}_{n\in\mathbb{N}}$ where $P_n \subseteq F_n$ and where $F_n = \{0,1\}^{2^n}$ is the set of all Boolean functions on n variables

- A property $\{P_n\}_{n\in\mathbb{N}}$ is a natural property if:
  - Given a 2^n-size truth table for a function f on n variables, checking whether $f \in P_n$ can be done in time $2^{O(n)}$ – Constructiveness, or "easy to detect"
  - $\Pr_{f\sim\{0,1\}^{2^n}} [\, f \in P_n \,] \geq 1/\mathrm{poly}(2^n)$ – Largeness, or "pretty common"

- A natural property $\{P_n\}_{n\in\mathbb{N}}$ is useful against P/poly if for any family $f = \{f_n\}_{n\in\mathbb{N}}$ of Boolean functions such that $f_n \in P_n$ for all n it holds that the circuit size of f is super-polynomial.

- (These properties are a way of showing that functions f are not in P/poly)

Natural proofs barrier (ct'd)

- Result by Razborov and Rudich (1994) that was awarded the 2007 Gödel Prize:
  - If there exists a natural property useful against P/poly, then there exist no (subexponentially-secure) one-way functions.
- In other words, assuming OWFs exist, then one needs non-natural properties to show circuit lower bounds for P/poly
- (Current proofs showing circuit lower bounds use natural properties)

Kolmogorov Complexity

Kolmogorov complexity: origins in randomness

- One of the main roots of Kolmogorov complexity is the study of randomness
- Consider the strings 000000000000 and 011011110010, both of length 12.
  - Is one more 'random' than the other?
- How do we measure this? Perhaps considering a probability distribution over all strings of length 12 and considering the probability of the strings. The uniform distribution doesn't help to define randomness.
- Idea of Kolmogorov complexity: measure the amount to which strings can be compressed.

Kolmogorov complexity

- Pick some universal Turing machine $\mathbb{U}$.
- The Kolmogorov complexity $C(x)$ of a string x is defined as:

$$C(x) = \min\{\ |p| : \mathbb{U}(p) = x\ \}.$$

- In other words, the Kolmogorov complexity $C(x)$ of x is the size of the smallest program p that, when executed by $\mathbb{U}$, yields x as output.

Kolmogorov complexity is uncomputable

- The problem of computing the Kolmogorov complexity $C(x)$ of a string x is uncomputable.
  - Main idea: an incompressibility argument.
  - Suppose, to derive a contradiction, that C is computable.
  - Consider the following algorithm $\mathbb{A}_M$, whose description will be of length $P + \log M$:
    - Iterate over all strings $x \in \{0,1\}^*$, from shortest to longer.
    - For each string x, compute $C(x)$. If $C(x) \geq M$, return x.
    - (In other words, $\mathbb{A}_M$ returns the first string x with $C(x) \geq M$.)
  - Now select M such that $M > P + \log M$.
  - Let x be the string that $\mathbb{A}_M$ returns. So $C(x) \leq P + \log M < M$.
    This contradicts that $C(x) \geq M$.

Time-bounded Kolmogorov complexity

- Resource-bounded variants of Kolmogorov complexity have been considered.
- Let $t : \mathbb{N} \to \mathbb{N}$.
- Then:

$$C^t(x) = \min\{ \ |p| : \mathbb{U}(p) = x \text{ in time } t(|x|) \ \}.$$

- Observation: for each x and each t, it holds that $C(x) \leq C^t(x)$.

Levin's Kt complexity

- Levin's Kt complexity is another variant that is based on time bounds.
- It is defined as follows:

$$Kt(x) = \min\{\ |p| + \log t : \mathbb{U}(p) = x \text{ in time } t\ \}.$$

- Observation: for each x, it holds that $C(x) \leq Kt(x)$.

Computational problems: MINKT, MK^{t}P, and MKtP

- MINKT: given a string x and $s, t \in \mathbb{N}$ in unary, decide whether there is a program p of size $\leq s$ such that $\mathbb{U}(p) = x$ in time t.
  - in NP
- MK^{t}P: given a string x and $s \in \mathbb{N}$ in unary, decide whether there is a program p of size $\leq s$ such that $\mathbb{U}(p) = x$ in time $t(|x|)$.
  - in NP
- MKtP: given a string x and $s \in \mathbb{N}$ in unary, decide whether $Kt(x) \leq s$.
  - in EXP

Hardness vs. randomness

Cryptographic PRGs

Definition

Let $G : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function, and let $\ell : \mathbb{N} \to \mathbb{N}$ be such that $\ell(n) > n$ for each n. Then G is a *secure pseudorandom generator (PRG) of stretch* $\ell(n)$, if $|G(x)| = \ell(|x|)$ for every $x \in \{0,1\}^*$ and for every probabilistic polynomial-time A there exists a negligible function ϵ such that for each n:

$$\left|\Pr[A(G(U_n)) = 1] - \Pr[A(U_{\ell(n)}) = 1]\right| < \epsilon(n).$$

Proposition

If OWFs exist, then for each c there exists a secure PRG with stretch $\ell(n) = n^c$.

- PRGs are useful building blocks for cryptographic schemes.

Complexity-theoretic PRGs

Definition

A distribution R over $\{0,1\}^m$ is (S,ϵ)-*pseudorandom* if for every circuit C of size at most S:

$$|\Pr[C(R)=1] - \Pr[C(U_m)=1]| < \epsilon.$$

Let $S : \mathbb{N} \to \mathbb{N}$ be some function. A 2^n-time computable function $G : \{0,1\}^* \to \{0,1\}^*$ is an $S(\ell)$-*pseudorandom generator* if $|G(z)| = S(|z|)$ for every $z \in \{0,1\}^*$ and for every $\ell \in \mathbb{N}$ the distribution $G(U_\ell)$ is $(S(\ell)^3, 1/10)$-pseudorandom.

- Differences with cryptographic PRGs:
  - Running time may be 2^n (instead of polynomial)
  - We will look at stretch $S(\ell) = 2^{O(\ell)}$, vs. e.g., stretch $\ell(n) = n+1$.
  - The adversaries are circuits rather than probabilistic algorithms.

PRGs and derandomization

- Result by Nisan and Wigderson (1988):
  - If there is some $f \in \mathsf{E} = \mathsf{DTIME}(2^{O(n)})$ that is *average-case hard* for subexponential-size circuits, then there exists a complexity-theoretic PRG with exponential stretch, and as a result $\mathsf{P} = \mathsf{BPP}$.
- Result by Impagliazzo and Wigderson (1997):
  - If there is some $f \in \mathsf{E} = \mathsf{DTIME}(2^{O(n)})$ that is *worst-case hard* for subexponential-size circuits, then there exists a complexity-theoretic PRG with exponential stretch, and as a result $\mathsf{P} = \mathsf{BPP}$.

Learning

PAC Learning

- Probably Approximately Correct (PAC) learning works as follows
- Take an instance space X. A concept $c \subseteq X$ is a subset of instances. A concept class C is a set of concepts.
- An algorithm A PAC-learns C if the following holds, for any (unknown) probability distribution D over the instances, and for any (unknown) correct concept $c_0 \in C$.
  - The algorithm takes as input $0 < \epsilon, \delta < 1$. It runs in time polynomial in $1/\epsilon$ and $1/\delta$.
  - It may (probabilistically) sample instances x according to the distribution D, and it receives the correct answer for x (i.e., whether $x \in c_0$).
  - With probability at least $1 - \delta$ it outputs a concept $h \in C$ such that the average error of h w.r.t. c_0 (according to the distribution D) is at most ϵ.

PAC Learning Boolean Circuits

- An algorithm PAC-learns a class $\mathcal{C}$ of Boolean functions if for any function $f \in \mathcal{C}$ on n variables the following holds:
  - The algorithm takes as input $0 < \epsilon, \delta < 1$.
    The running time does not depend more than polynomially on $1/\epsilon$ and $1/\delta$.
  - It may call f as an oracle—i.e., for strings $x \in \{0,1\}^n$, the oracle returns the value of $f(x)$.
  - With probability at least $1 - \delta$ it outputs a circuit C that agrees with f on all but an ϵ fraction of strings $x \in \{0,1\}^n$.

Learning via natural properties

- Carmosino, Impagliazzo, Kabanets and Kolokolova (2016) showed a connection between natural properties and learning.
- In simplified and imprecise form, the result states that:
  - If there is a natural property that is useful against circuits in class $\mathcal{C}$,
  - then using this property, one can construct a randomized algorithm that PAC-learns $\mathcal{C}$.
  - (The running time of the learning algorithm depends on the strength of the natural property: the stronger the natural property, the faster the learning algorithm.)
  - (The proof of this result uses the connection between circuit lower bounds and PRGs.)

Recent directions and results

Some recent directions and results..

- Worst-case to average-case reductions: ruling out Heuristica
  - See, e.g., a survey paper by Hirahara (2022; link)
- Evidence that MCSP is NP-complete:
  - Partial MCSP is NP-complete (2022; link)
  - Reducing SAT to variants of MCSP (2023; link)
- Connections to Kolmogorov complexity:
  - OWFs exist if and only if time-bounded Kolmogorov complexity is hard on average (2020; link)
  - OWFs exist if and only if an NP-complete problem related to Kolmogorov complexity is hard on average (2020; link)
- Replacing SAT oracles by MCSP oracles in complexity-theoretic results (2018; link)