CS 331, Fall 2024

Lecture 26 (12/9)

Today: – Kevin's research
– Fine-grained complexity
– Popular conjectures

Kevin's research

Algorithmic primitives for "big" data science

$\approx 1/2$: continuous algos foundations (opt, samp, NLA)

$\approx 1/2$: trustworthy ML (robustness, privacy, fairness)

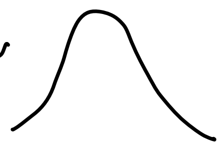
Key themes:

- Many problems hard in worst case. Harness structure.
- Where does the data come from? Minimal assumptions.
- Dataset sizes enormous, only growing. Near-linear time?

Trustworthy ML

Textbook setting for statistics / learning:

$$\begin{array}{l} X_1 = (\qquad) \\ X_2 = (\qquad) \\ X_3 = (\qquad) \\ \quad \vdots \\ X_n = (\qquad) \end{array} \sim$$



known density, e.g. Gaussian

$\Rightarrow$ A(data)

analyst runs learning algo to train model

$\Rightarrow$ new data $\to$ model $\to$ prediction

Real life is harder.

- What if we're wrong about the data? <u>Robustness</u>
- The data is coming from humans. <u>Privacy / fairness</u>
- Why do we believe the model's conclusion? <u>Interpretability</u>
- ... all needs to happen efficiently ...

Continuous algos

What kind of tools are useful for modern algo design?

OPT: Minimize structured objectives.

e.g. minimax / stochastic optimization

semidefinite programming ("matrix LP")

structured nonconvex problems (sparsity, GLM, ...)

SAMP: Sample from structured densities.

e.g. logconcave sampling (basic tractable family)

structured multimodal problems

NLA: numerical linear algebra primitives

e.g. preconditioning (solving linear systems, regression)

sparsification (replace data w/ representatives)

2014: Google internship. Not good at it...

2015: Complexity research. Not good at it...

2016: Genomics research. Really fun! I liked algos best...

2017: Genomics / NLP / stats research. (Ph.D. rotations)

2018: Approximate maxflow.

2019: Nash equilibria, optimal transport, SDP.

2020: Sampling. SOTA for some logconcave families.

2021: Robust stats. PCA, regression, clustering in near-linear time.

2022-24: Privacy, interpretability, etc.

I am trying to learn more about modern ML...

Algorithms are cool and come in many flavors. There are so many connections. Just keep learning and enjoy ☺

Fine-grained complexity

Thanks to: Virginia Williams (notes)
Amir Abboud } Youtube
Nick Fischer }

So far: Complexity theory for "small data"

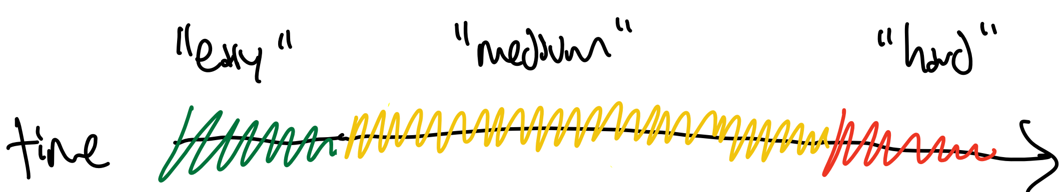
$n = 100, 1000, 10000 \ldots$

polytime: Sort/search, arithmetic, linear algebra, strings, reachability, APSP

not polytime: (we think) SAT, VertexCover, Knapsack, TSP, MaxCut, Sudoku

It's 2024. We need complexity theory for "big data"

$n = 10^9, 10^{12}, 10^{15} \ldots$



near-linear time | polytime \ near-linear | not polytime

NP: a tool to prove problems hard.

Today: how to prove problems medium.

Known easy	Suspected medium
FFT	3-SUM
Shortest path	All-pairs shortest paths
Maxflow	Dynamic maxflow
Longest increasing subseq.	Longest common subseq.
Closest pair in $\mathbb{R}^2$	Closest pair in $\mathbb{R}^d$
Longest palindromic substring	Edit distance

e.g. APSP $\quad n^3 \quad \ldots \quad \ldots \quad n^{3-o(1)}$

Floyd-Warshall '62 $\qquad$ Williams '14

Why are we so good at problems on LHS

... but bad at problems on RHS?

Goal in FGC: web of reductions, $\underbrace{\text{common source of hardness}}_{\text{must attack first!}}$

Popular conjectures

Let $\varepsilon > 0$ be small constant.

3-SUM: Given list L of #'s, $\exists\, a, b, c \in L$ s.t. $a + b + c = 0$?

... cannot be solved in time $O(|L|^{2-\varepsilon})$

APSP: Given graph $G = (V, E, w)$ compute $|V| \times |V|$ matrix encoding all-pairs shortest paths

... cannot be solved in time $O(|V|^{3-\varepsilon})$

SETH: $\exists$ constant k s.t. k-SAT on Φ w/ m clauses, n variables

... cannot be solved in time $O(2^{n(1-\varepsilon)} \mathrm{poly}(m))$

Rough intuition: we care about the exponent now.

$\boxed{\text{3-SUM}}$

Computational geometers Gajentaan – Overmars '95

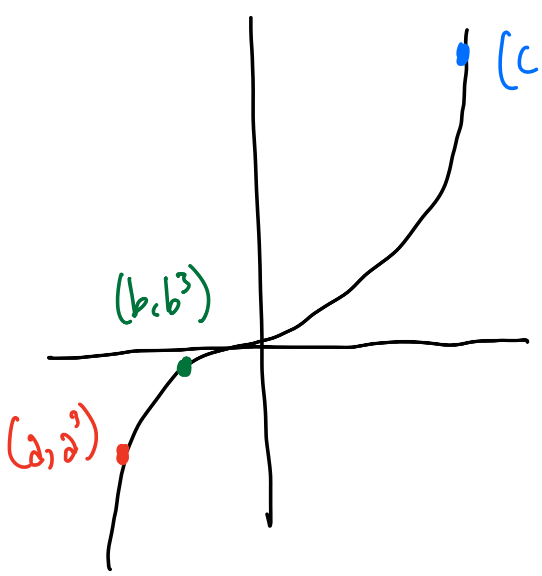
kickoff field of FGC by reducing to 3-SUM.

Example: Collinearity requires $\approx n^2$ time.

Input: n points in $\mathbb{R}^2$. $\exists$ three on a line?

(This proof is crazy.) Reduce 3-SUM to collinearity.

Check collinearity$(\{(x, x^3) \mid x \in L\})$



$$\frac{b^3 - a^3}{b - a} = \frac{c^3 - b^3}{c - b}$$

$$a^2 + ab + b^2 = c^2 + cb + b^2$$

$$a^2 - c^2 = b(c - a)$$

$$-a - c = b \quad \text{(3-SUM)}$$

More: visibility / reachability, motion planning, polygon containment $\Big\}$ several require ingenuity to solve in $\approx n^2$.

APSP

This is really about "combinatorial" matrix multiplication.

Many amazing "algebraic" innovations:

$(n \times n)\ (n \times n)$ takes time...

multiply

n^3 (duh)

$n^{2.807}$ Strassen

$\vdots$

$n^{2.3714}$ ADWXXZ

Use crazy cancellations on huge tensors...

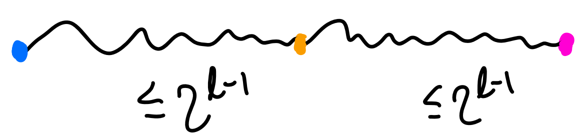
What if we can only use more "baseline" techniques?

Recall <u>divide-and-conquer</u> + <u>DP</u> algo for APSP

$DP[s][t][\ell]$: shortest $s-t$ path w/ $\leq 2^{\ell}$ edges

$$= \min_{u \in V} DP[s][u][\ell-1] + DP[u][t][\ell-1]$$

"guess the midpoint"



This is the same problem as "dot product"

except sum $\longrightarrow$ min

product $\longrightarrow$ sum

$$DP_{\ell-1} \otimes DP_{\ell-1} = DP_\ell \quad \forall \ell \in [O(\log(n))]$$

"min-plus" convolution

To solve APSP, need to solve combinatorial matmul $O(\log(n))$ times. Thus, APSP conjecture:

Combinatorial matmul needs $\approx n^3$ time ☹

Good news: <u>structured</u> matmul / inversion / ... easier!

$\boxed{\text{SETH}}$

Recall 3-SAT solvable in: $O(2^n m)$ time

Improvement: Try 7/8 for one clause, recurse.

$$T(n) \leq 7T(n-3) + O(m) \Rightarrow T(n) = O(1.913^n m)$$

So, 3-SAT does not need 2^n time.

More general: k-SAT in $2^{n(1-O(\frac{1}{k}))}$ m time.

But we need tight base (will later choose $n = \log$)
base becomes exponent.

Hence SETH: Choose large enough k.

Almost all modern reductions use SETH thru:

$$\text{SETH} \leq \text{OV}$$ Williams, 2005

(orthogonal vectors)

OV implies FGC of so many problems:

- Diameter
- Dynamic reachability
- Single-source maxflow
- Edit distance
- Fréchet distance
- LCS
- Local alignment
- Stable matching
- Closest pair

2-OV problem:

Let $d = \omega(\log(n))$ "sparse subset"

$A, B \subset \{0,1\}^d$, size n

$\exists\, a \in A,\ b \in B$ s.t. $\underbrace{a^T b = 0}_{\text{orthogonal vectors}}$?

Conjecture: no better than $\approx n^2 d$ possible.

k-OV: there are k sets

$A_1\ A_2\ A_3\ \dots \subset \{0,1\}^d$

$\exists\, a_1 \in A_1,\ a_2 \in A_2,\ a_3 \in A_3, \dots$

s.t. $\sum_{i \in [d]} a_1[i]\, a_2[i]\, a_3[i] \dots = 0$?

Conj: needs $\approx n^k d$ time.

Obs 1: 2-OV is very fundamental. FGC!

Obs 2: 2-OV $\gtrsim$ 3-OV $\gtrsim \dots \gtrsim$ k-OV.

Obs 3: "OV $\gtrsim$ SETH".

Suppose there's k-OV in $O(N^{k(1-\varepsilon)})$.

Take k-SAT formula Φ, n variables
m clauses

Create $A_1, \dots, A_k \subset \{0,1\}^m$:

$x \in \{0,1\}^n \to (\underbrace{x_1}_{n/k} \mid \underbrace{x_2}_{n/k} \mid \dots \mid \underbrace{x_k}_{n/k})$

blocks

A_i = index by $N = 2^{n/k}$ assignments to ith block

Faster k-OV $\Rightarrow$ SAT in time $N^{k(1-\varepsilon)} = 2^{n(1-\varepsilon)}$

(violates SETH!)