October 9, 2025

Problem Set 1

Exercise 1: Geometry of High Dimensions. There is a wide class of algorithms that take a set of *data samples* as input, particularly in the field of machine learning. A powerful tool in this setting is to interpret the data geometrically as points lying in $\mathbb{R}^d$. The goal of this exercise is to demonstrate that some geometric notions behave unintuitively in high dimensional space, and hence it is important to thoroughly check our intuitions when working in higher dimensions.

Let $\mathcal{B}_d$ denote the d-dimensional unit ball. It is the set of points with distance at most 1 from the origin:

$$\mathcal{B}_d := \left\{ (x_1, x_2, \ldots, x_d) : \sum_{1 \le i \le d} x_i^2 \le 1 \right\}.$$

The volume of $\mathcal{B}_d$ is denoted by V_d and can be computed directly by integration:

$$V_d := \int_{x \in \mathcal{B}_d} \mathrm{d}x_1 \, \mathrm{d}x_2 \ldots \mathrm{d}x_n \, .$$

Brushing over the details of this calculation, the actual retail price is

$$V_d = \frac{\pi^{d/2}}{\Gamma(d/2+1)} \tag{$*$}$$

where Γ is the gamma function which satisfies $\Gamma(n+1) = n!$ and

$$\Gamma(n+1/2) = \pi^{1/2} \prod_{k=0}^{n-1} (k+1/2).$$

1. Recall that the area of a circle is πr^2 and the volume of a sphere is $\frac{4}{3}\pi r^3$. Do a sanity check for Equation $(*)$ by verifying that $V_2 = \pi$ and $V_3 = \frac{4}{3}\pi$.
2. Evaluate the limit $\lim_{d \to \infty} V_d$ directly.
3. For what dimension d is V_d maximal? Hint: consider the ratio V_d / V_{d-1}.
4. Using your favorite programming language, compute V_d for d in the range $[1, 30]$. Make a plot with V_d on the y-axis and d on the x-axis. Does the plot support the conclusions you made in parts 2-3?
5. (The unit cube). A 3-dimensional cube has vertices, edges, and faces. In d dimensions, these components are all faces: vertices are 0-dimensional faces, edges are 1-dimensional faces, etc.
   (a) How many faces of dimension 0, 1, 2, and 3 does a 3-d cube have?
   (b) Write out the coefficients of the polynomial $(x+2)^3$.
   (c) For a d-dimensional cube, how many faces of dimension k does it have for each $0 \le k \le d$? How does your result compare to part (b)?

Exercise 2: DFAs. A deterministic finite automaton or DFA is a "machine" that on any input, either accepts or rejects that input. It only has a finite state set and a read only head. On input w, it starts on a special state q_0 by reading the first letter and based on the letter and the state it is in, transitions into a new state and goes to read the next letter. This transition rule is given by a function called a transition function. Some of the states are accepting states and the DFA is said to accept its input w if and only if the state it was last at was an accepting state.
Formally, a DFA is a 5-tuple $D = \{Q, \Sigma, \delta, q_0, F\}$: Q is the state set, Σ is the alphabet of the input, $\delta : Q \times \Sigma \to Q$ is the transition function, q_0 is the start state, and $F \subseteq Q$ is the set of accepting states.

1. Design a DFA that only accepts strings with an even number of a's, where $\Sigma = \{a, b\}$.

2. Design a DFA that only accepts strings with $n \equiv 1 \bmod 3$ number of a's and an even number of b's.

3. Design a DFA that only rejects strings that include aab as a substring.

This this set of strings that are accepted by a DFA is the language decided by the DFA (or $L(D)$).

4. Given a DFA D, show that the complement of its language $L(\overline{D}) := \{x : x \notin L(D)\}$ can be decided by a DFA $\overline{D}$.

5. Given a DFA D and R, show that $L(D) \cap L(R)$ and $L(D) \cup L(R)$ both have DFAs that decide them (they may be different DFAs).

Exercise 3: Partial Order. Let A be a set. A **partial order** P on A is defined as a subset of $A \times A$ satisfying:

- Reflexivity: $\forall x\colon P(x, x)$
- Antisymmetry: $\forall x\colon \forall y\colon P(x, y) \wedge P(y, x) \to x = y$
- Transitivity: $\forall x\colon \forall y\colon \forall z\colon (P(x, y) \wedge P(y, z)) \to P(x, z)$

If you've taken CS 250 recently, the first two parts should look familiar:

1. Show that $\leq$, $=$ and divisibility ($D(a, b)$ meaning $\exists x\colon ax = b$) are all partial orders on the naturals, while $\neq$, $<$ and divisibility are not partial orders on the integers.
2. Let $A = \{a, b, c, d, e, f\}$. Let P be a partial order on A. You are given that:
   - $(d, e), (d, b), (a, c), (c, f)$ are all in P.
   - $(a, b), (d, f), (b, e)$ are all not in P.

   In table form:

P	$a)$	$b)$	$c)$	$d)$	$e)$	$f)$
$(a,$		$\times$	✓			
$(b,$					$\times$	
$(c,$						✓
$(d,$		✓			✓	$\times$
$(e,$						
$(f,$						

   Deduce as many forced assignments in the table above as possible. There should be 4 blanks left once you're finished.

Now, let's formalize problems of the same form as the one given above. Let I and O be two subsets of $A \times A$. We define a partial order P on A to be **IO-valid** if $A \subset P$ and $O \cap P = \emptyset$. In the above example, P is assumed to be IO-valid where $I = \{(d, e), (d, b), (a, c), (c, f)\}$ and $O = \{(a, b), (d, f), (b, e)\}$. We were then asked to add as many elements into I and O as possible without changing the possible P. Let's formalize that as follows:

Given a finite set A, I and O as defined above, find I_{max} and O_{max} such that:

- $I_{max}O_{max}$-validity is equivalent to IO-validity, and
- Any pair (I', O') with the same property satisfies $I' \subset I_{max}$ and $O' \subset O_{max}$.

In less formal terms, I_{max} and O_{max} is the final result of adding all the forced assignments to I and O.

3. Is it possible that I_{max} and O_{max} doesn't exist? What would I_{max} and O_{max} be if I and O causes a conflict, so there's no IO-valid partial order?
4. Create any algorithm that solves the problem given $\{A, I, O\}$ in polynomial time.
5. Can you do better than $O(n^3)$ complexity, where $n = |A|$?

Exercise 4: Kolmogorov Complexity. A *universal Turing machine* is a Turing machine U which can simulate any other Turing machine M given some encoding of it, along with an input to simulate it on. In more familiar terms, consider your favorite programming language L. Then a universal program U for L is a program which can simulate any other program, i.e. an *interpreter*.

The *Kolmogorov complexity* of a string $x \in \{0,1\}^*$, denoted $K_U(x)$, is the length of the shortest input which makes U output x on its tape. For all U, the function $K_U(\cdot)$ is not computable, however we can still give interesting and useful bounds on the Kolmogorov complexity of strings.

The first 4 questions are simple and meant to give you a working understanding of how $K_U(\cdot)$ behaves. The next 2 questions are more difficult, but showcase how Kolmogorov complexity can be a very versatile tool.

1. ($\star$) Show that for any two universal Turing machines U, U', there exists a constant c such that for all $x \in \{0,1\}^*$, we have $K_U(x) \leq K_{U'}(x) + c$.

   (**Note:** Hereafter we may simply write $K(\cdot)$, which implicitly relies on some unspecified universal Turing machine.)

2. ($\star$) Show that for all n, there exists $x \in \{0,1\}^n$ such that $K(x) \geq n$.

3. ($\star$) Show that for all $x, y \in \{0,1\}^*$, we have $K(xy) \leq K(x) + K(y) + O(1)$.

4. ($\star\star$) Show that it is not necessarily the case that for all $x, y \in \{0,1\}^*$, we have $K(xy) > K(x)$.

5. ($\star\star$) Show that there are infinitely many prime numbers using Kolmogorov complexity.

6. ($\star\star\star$) Let $p_1, p_2, p_2, \ldots$ be a list of all the primes in ascending order. Show that for all $n \geq 1$, we have $p_n = O(n \log^2 n)$ using Kolmogorov complexity.

   (**Note:** The Prime Number Theorem says $p_n = O(n \log n)$.)