CSCI 145 Problem Set 1

August 25, 2025

Submission Instructions

Please upload *your* work by **5pm Friday September 19, 2025.**

- You are encouraged to discuss ideas and work with your classmates. However, you **must acknowledge** your collaborators at the top of each solution on which you collaborated with others and you **must write** your solutions independently.
- Your solutions to theory questions must be written legibly, or typeset in LaTeX or markdown. If you would like to use LaTeX, you can import the source of this document (available from the course webpage) to Overleaf.
- I recommend that you write your solutions to coding questions in a Jupyter notebook using Google Colab.
- You should submit your solutions as a **single PDF** via the assignment on Gradescope.

Grading: The point of the problem set is for *you* to learn. To this end, I hope to disincentivize the use of LLMs by **not** grading your work for correctness. Instead, you will grade your own work by comparing it to my solutions. This self-grade is due the Monday *after* the problem set is due, also on Gradescope.

Problem 1: Fast Matrix-Vector Multiplication

Let d and q be non-negative integers. Consider matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$, and vector $\mathbf{x} \in \mathbb{R}^d$. The goal in this problem is to compute $\mathbf{A}^q\mathbf{x}$ quickly with increasing levels of sophistication.

Part A: Naive Attempt

Describe a method to compute $\mathbf{A}^q\mathbf{x}$ in time $O(q \cdot d^3)$.

Part B: Strategic Matrix Multiplication

For convenience, suppose that q is a power of 2. Describe a method to compute $\mathbf{A}^q\mathbf{x}$ in time $O(\log_2 q \cdot d^3)$.

Part C: Vector Multiplication

Describe a method to compute $\mathbf{A}^q\mathbf{x}$ in time $O(q \cdot d^2)$.

Part D: Eigendecomposition

Suppose that c^q can be computed in constant time, for any real number $c \in \mathbb{R}$. Describe a method to compute $\mathbf{A}^q\mathbf{x}$ in time $O(d^3)$.

Part E: Empirical Checks

Initialize a vector $\mathbf{x}$ and *symmetric* matrix $\mathbf{A}$, where each entry is drawn independently from a Gaussian distribution. Implement your various strategies in python, and plot their runtimes for $d = 10$ and $q \in \{2, 4, 8, 16, 32, 64, 128, 256\}$.