CSCI 145 Problem Set 1

September 1, 2025

Submission Instructions

Please upload *your* work by **11:59pm Monday September 22, 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 Friday *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 $q \in \{2, 4, 8, 16, 32, 64, 128, 256\}$.

Problem 2: Opinion Dynamics

Recently, many people have observed that politics feels more polarized. Some of this phenomenon is likely due to echo chambers and social media, but we can also gain insight through a more mathematical lens: A simplified model of how people form opinions.

Consider a graph, where the nodes represent people and the edges represent the friendships between them. Let $\mathbf{z}^{(0)} \in \mathbb{R}^n$ be a vector representing people's initial opinions on a topic, e.g., positive is for and negative is against. The DeGroot update procedure is arguably the most simple way to model opinion dynamics: At every iteration t, we build the new opinions $\mathbf{z}^{(t)}$ so that each person's opinion becomes the average of their friends' prior opinions $\mathbf{z}^{(t-1)}$.

Part A: Linear Algebra Representation

Define a matrix $\mathbf{A}$ so that

$$\mathbf{z}^{(t)} = \mathbf{A}\mathbf{z}^{(t-1)}.$$

Part B: Power Method

Using the `networkx` package in Python, load a small graph of your choice (e.g., the karate club graph). Build the matrix $\mathbf{A}$ for this graph. Then update a randomly initialized opinion vector 100 times.

Part C: Mean Centering and Normalization

As you may expect, averaging the opinions will make them converge to a scaling of the all ones vector (under some mild assumptions about the structure of the graph).

Instead, after each update step, mean center the opinions, and divide by their standard deviation. Mathematically, this effectively projects off the top eigenvector. Socially, this is like zooming in to focus only on the *differences* in opinion.

Now, plot the opinions after 100 updates. (It will help if you fix the position of the nodes in the visualization from one random run to the next.) What do you notice about the final opinions when you visualize them for different randomly initialized opinions?

Part D: Eigenvector View

Use `numpy` to compute the eigenvectors of $\mathbf{A}$. What is the first eigenvector? Overlay the second eigenvector on your graph visualization, how does it compare to the final opinion vector when you mean center and normalize?

Part E: Philosophy

Let's assume DeGroot dynamics are realistic, what could you conclude about the way people form opinions?