# HW 4

Due: Friday 12/05/2025 @ 11:59pm EST

## Disclaimer

I encourage you to work together, I am a firm believer that we are at our best (and learn better) when we communicate with our peers. Perspective is incredibly important when it comes to solving problems, and sometimes it takes talking to other humans (or rubber ducks in the case of programmers) to gain a perspective we normally would not be able to achieve on our own. The only thing I ask is that you report who you work with: this is **not** to punish anyone, but instead will help me figure out what topics I need to spend extra time on/who to help. When you turn in your solution (please use some form of typesetting: do **NOT** turn in handwritten solutions), please note who you worked with.

### Question 1: Using Pagerank to Encode Other Properties

The pagerank transform is defined as follows (for $0 \le \alpha \le 1$):

$$\mathbf{M}' = \alpha\mathbf{M} + (1-\alpha)\frac{1}{|V|}\mathbf{1}$$

Where the uniform matrix $\frac{1}{|V|}\mathbf{1}$ is used to guarantee that $\mathbf{M}'$ is aperiodic and connected.

The Pagerank transform is most famously used by Google as the backend of their search. However, Pagerank in its current form is not very useful to Google: it certainly ranks vertices (webpages) based on the connectivity of the internet, but often users want to find webpages that are important **and** contain the information they want: something the Pagerank transform is currently lacking.

Your job is to augment the Pagerank transform so that webpages will be ranked based on the connectivity of the internet as well as whether a webpage contains a keyword $k$ that we are searching for. Your solution must rank webpages higher if keyword $k$ is present in that webpage than webpages that do not contain the keyword. For simplicity, assume that we can tell if vertex $v$ contains $k$ with the expression $k \in v$.

To be clear, you must do two things:

1. Prove that your solution encodes the desired properties.
2. Prove that for vertices containing the keyword $k$, those vertices have higher rankings according to the stationary distribution than they did when the stationary distribution was calculated solely from the connectivity of the internet.

## Question 2: From Binary Classification to Multi-class Classification

So far in class, we have talked about binary classification and completely ignored multi-class classification (i.e. when there are more than 2 classes). In this problem, you are to design a procedure for solving a multi-class classification problem but you are only allowed to use binary classification models to do so. To be clear, you are allowed to use as many binary *classifiers* as you want, but the only models you are allowed to use are binary classification models.

To be clear, you must do two things:

1. Prove that your solution can function for an arbitrary number of classes.
2. Answer any details like *how* your solution trains/predicts using its binary classifiers.

## Question 3: Performance Metrics

Consider a dataset with the following label distribution:

| Class $c$ | $Pr[c]$ |
|---|---|
| Red | 0.05 |
| Blue | 0.87 |
| Green | 0.08 |

What metric(s) would you **avoid** when evaluating a model on a this dataset and why? You do **not** have to use a two-column proof structure for this question but I expect you to back up your answer with mathematical justification (e.g. formally define *when* metric A is bad vs. good, etc.).

**Extra Credit: Max-miniumum Degree Problem (25 points)**

Consider an undirected, unweighted graph $G = (V, E)$. For any subset of vertices $X \subseteq V$, the graph *induced* by $X$, denoted $G_X = (X, E_X)$ is the graph containing vertex $X$ as well as the subset of edges $E_X \subseteq E$ that have *both* of their endpoints in $X$ (i.e. $G_X$ is what remains of $G$ if we delete vertices not in $X$ as well as any edge with either one or both of its endpoints not in $X$). For brevity, let us denote $deg(v, G)$ as the degree of vertex $v$ in some graph $G$.

Given $G = (V, E)$, find $S \subseteq V$ such that

$$\min_{v \in S} deg(v, G_S)$$

is maximized.

To be clear, you are to design a polynomial time algorithm which solves this problem *optimally.* You are to prove the running time of your algorithm is polynomial time as well as prove that your algorithm is optimal.