Lab 7

Due: Friday 11/14/2025 @ 11:59pm EST

The purpose of labs is to give you some hands on experience programming the things we've talked about in lecture (and some we will talk about in lecture). The purpose of this lab is for you to learn how to program the power method

Task 0: Setup

Included with this file is a new file called `requirements.txt`. This file is rather special in Python: it is the convention used to communicate dependencies that your code depends on. For instance, if you open this file, you will see entries for `numpy` and `scikit-learn`: two pythn packages we will need in order to run the code in this lab. You can download and install these python packages through `pip`, which is python's package manager. While there are many ways of invoking `pip`, I like to do the following:

```
python3 -m pip install -r requirements.txt
```

(I let `python3` figure out which `pip` is attached to it rather than the more common usage: `pip install -r requirements.txt`)

Task 1: `function vanilla_pm` (50 points)

In the file `pm.py`, you will find a function called `vanilla_pm`. This method should calculate the vanilla power method for finding the stationary distribution of markov chain $\mathbf{M}$. The way the vanilla power method works is to repeatedly calculate $\vec{x}^{(t+1)} = \vec{x}^{(t)}\mathbf{M}$ until $\vec{x}^{(t+1)}$ and $\vec{x}^{(t)}$ are "close enough" and $\vec{x}^{(0)} := \frac{1}{|V|}\vec{1}$.

Remember that the vanilla power method requires that the transition matrix given to it be **already** connected and aperiodic, so this matrix should **already** be dense (i.e. have type `np.ndarray`). Your function will continue to calculate a distribution over the vertices repeatedly until the distribution converges within a threshold ϵ **or** the total number of iterations exceeds a threshold.

Task 2: `function clever_pm` (50 points)

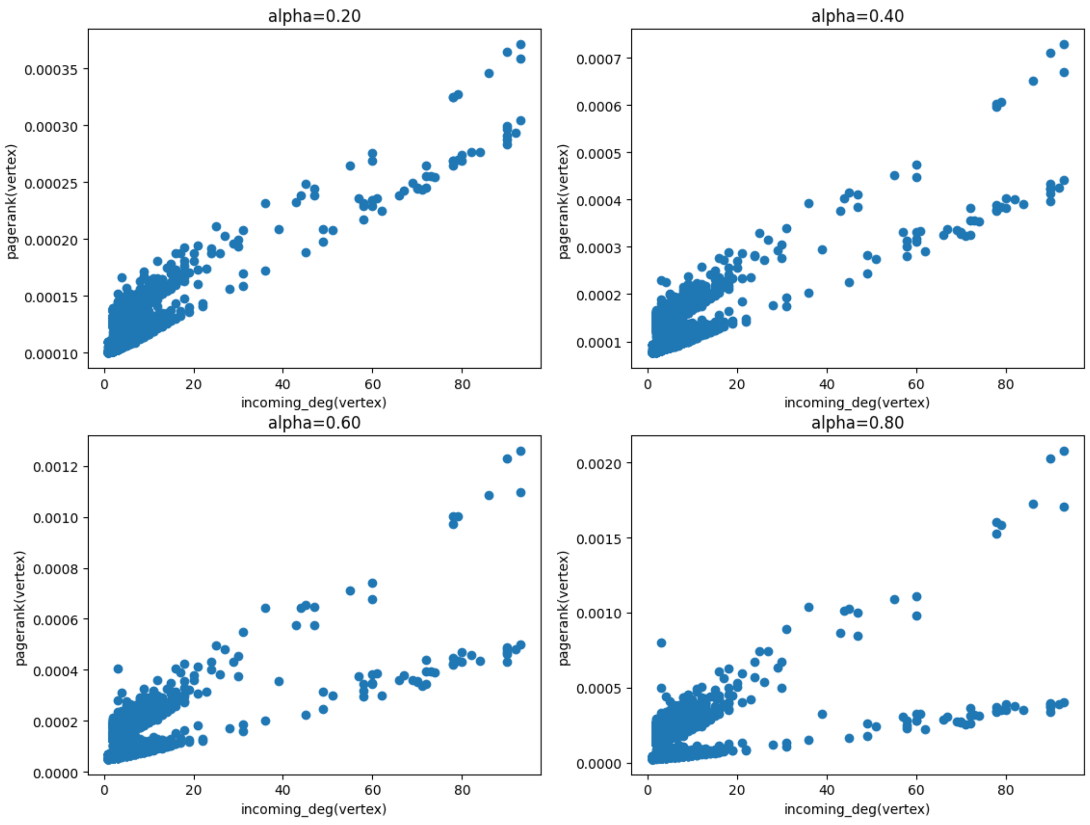
In the file `pm.py`, you will find a function called `clever_pm`. This method should calculate the clever power method which performs the vanilla power method but bakes the pagerank transform into the operation $\vec{x}^{(t)}\mathbf{M}$. The way this happens is that when presented with markov chain $\mathbf{M}'$ (the original markov chain provided as an argument), whenever we want to do $\vec{x}^{(t)}\mathbf{M}$ where $\mathbf{M}$ is the pagerank transform of $\mathbf{M}'$, we instead do the following operation:

$$\begin{aligned}\vec{x}^{(t+1)} &= \alpha\vec{x}^{(t)}\mathbf{M}' \\ \beta &= ||\vec{x}^{(t)}||_1 - ||\vec{x}^{(t+1)}||_1 \\ \vec{x}^{(t+1)} &+= \beta\frac{1}{|V|}\vec{1}\end{aligned}$$

Remember that the clever power method does **not** require that the transition matrix given to it be **already** connected and aperiodic, so this matrix **may** be dense (i.e. of type `np.ndarray`) or it **may** be sparse (i.e. any type in `scipy.sparse` module but I prefer to use the `csr_matrix` format). Your function will continue to calculate the a distribution over the vertices repeatedly until the distribution converges within a threshold ϵ or the total number of iterations exceeds a threshold. Reminder that in the clever power method, we bake the pagerank transformation into the vector-matrix multipication. The paramter α is what you should use as the convex coefficient of this pagerank transformation.

Task 3: Testing (0 points)

I have included a file called `load.py`. This will download a data file which contains an adjacency matrix that we will use for our markov chain. I have also made the file `pm.py` runnable, so if you were to run it, it will attempt to download this data file, unpack the contents, load the adjacency matrix as a `sp.coo_matrix`. After some quick normalizing of the rows, the testing code calculates the stationary distribution via the `vanilla_pm` method, and plots the incoming degree of each vertex against the stationary distribution. If all goes well, you should see a set of plots with a two-band structure like this:



Task 3: Submit Your Lab

Please drag and drop your complete `pm.py` file on gradescope!