Overview

In this assignment, you'll read sparse matrices from files, implement a class, and implement a simple spectral embedding algorithm.

All your code should be put in a script `hw1.py` which will be submitted on Canvas.

Please submit the following files to Canvas:

1. `hw1.py` - a script with all your code, which will generate the figure
2. `sbm.png`

Please use the function, class and file names specified. It will be easier to read your code if everyone uses the same conventions. Additionally, **please include comments in your code** to explain what you're doing (doesn't have to be detailed, but should be clear).

1 Review of Graph Terminology

A graph $G(V, E)$ is a set of vertices V and edges $E \subset V \times V$. We'll denote the number of vertices in a graph by n, and assign each vertex a number in $0, 1, \dots, n-1$.

The adjacency matrix of a graph is a $n \times n$ matrix A, where

$$A_{i,j} = \begin{cases} 1 & (i,j) \in E \\ 0 & (i,j) \notin E \end{cases}$$

2 Read A Sparse Matrix From a File

`sbm.csv` contains the adjacency matrix of a graph in the following format: each row of the file contains the contents for a single non-zero of the adjacency matrix:

`row (int), column (int), value (float)`

Write a function that will take a file name as input and return a `scipy.sparse.coo_matrix`

call this function `read_coo`

If you prefer to work with another sparse matrix type, you can always convert once you have created a COO matrix.

3 Create A Sparse + Rank-1 Matrix Class

In the next part, it will be convenient to have a class to represent matrices of the form

$$S + \alpha u v^T$$

where S is a sparse matrix, u, v are vectors, and α is a scalar. This allows us to use the structure of the matrix to avoid forming a dense array.

use the class name `sparse_rank1`

Write a class definition that

- initializes an object given a sparse matrix S, numpy arrays u and v and a float α (store each of these inputs). Give the object another field `shape`, which is set to be the same as `S.shape`
- implements a `dot` method, which performs matrix-vector multiplication

4 Power Method

In lecture 2, you did an exercise in which you implemented power method for a matrix. Recall, this function finds the eigenpair (λ, v) with largest λ such that $Av = \lambda v$ for a matrix A. Note that the implementation only required that A have a `dot` method that performed matrix-vector multiplication, so you can use it with your new matrix class.

Import this function to your script, and modify it if needed to work with your `sparse_rank1` matrix class.

call this function `power_method`

5 Spectral Embedding

`sbm.csv` contains the adjacency matrix of a graph generated using the stochastic block model. Load this into a sparse matrix `A`.

A spectral embedding assigns vertices of a graph coordinates in Euclidean space that can be used to visualize the graph. One way to generate a spectral embedding is to calculate the top k vectors of the adjacency matrix, and embed in $\mathbb{R}^k$. We'll do this for $k = 2$.

Find the top two eigenvectors of adjacency matrix using the power method using the following deflation algorithm:

1. Calculate the top eigenpair (λ_1, v_1) of A
2. Calculate the top eigenpair (λ_2, v_2) of $(A - \lambda_1 v_1 v_1^T)$

You can either wrap this in a function, or just execute it directly in the script.

Use PyPlot to generate a scatter plot of v_1 vs v_2. **Save this plot as `sbm.png` and submit it on Canvas.**

Note that this is a very simplified version of a specific spectral clustering algorithm. If you want to know more about this spectral clustering setup, I recommend the paper:

"Robust and efficient multi-way spectral clustering" by A. Damle, V. Minden, and L. Ying. (2017)
`https://arxiv.org/abs/1609.08251`

Hints

You don't need to use the hints to complete the assignment - it's ok if you want to use functions other than the ones mentioned.

- Reading a file: check out `numpy.readtxt`
- To specify a data type in a numpy array, pass in a type e.g. `np.array(v, int)`
- $(S + \alpha uv^T)x = Sx + \alpha u(v^T x)$
- Save a figure: check out `savefig` in pyplot
- You should see 2 clusters when you generate the figure