CSE 547: Machine Learning for Big Data **Spring 2024**

Problem Set 3

Please read the homework submission policies.

Assignment Submission All students should submit their assignments electronically via GradeScope. No handwritten work will be accepted. Math formulas **must** be typeset using LaTeX or other word processing software that supports mathematical symbols (E.g. Google Docs, Microsoft Word). Simply sign up on Gradescope and use the course code V84RPB. Please use your UW NetID if possible.

For the non-coding component of the homework, you should upload a PDF rather than submitting as images. We will use Gradescope for the submission of code as well. Please make sure to tag each part correctly on Gradescope so it is easier for us to grade. There will be a small point deduction for each mistagged page and for each question that includes code. Put all the code for a single question into a single file and upload it. Only files in text format (e.g. .txt, .py, .java) will be accepted. **There will be no credit for coding questions without submitted code on Gradescope, or for submitting it after the deadline**, so please remember to submit your code.

Coding You may use any programming languages and standard libraries, such as NumPy and PySpark, but you may not use specialized packages and, in particular, machine learning libraries (e.g. sklearn, TensorFlow), unless stated otherwise. Ask on the discussion board whether specific libraries are allowed if you are unsure.

Late Day Policy All students will be given two no-questions-asked late periods, but only one late period can be used per homework and cannot be used for project deliverables. A late-period lasts 48 hours from the original deadline (so if an assignment is due on Thursday at 11:59 pm, the late period goes to the Saturday at 11:59pm Pacific Time).

Academic Integrity We take academic integrity extremely seriously. We strongly encourage students to form study groups. Students may discuss and work on homework problems in groups. However, each student must write down the solutions and the code independently. In addition, each student should write down the set of people whom they interacted with.

Discussion Group (People with whom you discussed ideas used in your answers):

On-line or hardcopy documents used as part of your answers:

I acknowledge and accept the Academic Integrity clause.

*(Signed)*____

1 Dead Ends in PageRank Computations (25 points)

We learned about PageRank in lecture which is an algorithm for ranking webpages. In this problem, we are going to focus on dead ends and examine how they affect the PageRank computations.

Suppose we denote the matrix of the Internet as the n-by-n matrix M, where n is the number of webpages. Suppose there are k links out of the node (webpage) j, and

$$M_{ij} = \begin{cases} 1/k & \text{if there is a link from } j \text{ to } i \\ 0 & \text{otherwise} \end{cases}$$

For a webpage j that is a *dead end* (i.e., one having zero links out), the column j is all zeroes.

Let $\mathbf{r} = [r_1, r_2, \ldots, r_n]^\top$ be *an estimate* of the PageRank vector. In one iteration of the PageRank algorithm, we compute the next estimate $\mathbf{r}'$ of the PageRank as: $\mathbf{r}' = M\mathbf{r}$.

Given any PageRank estimate vector $\mathbf{r}$, define $w(\mathbf{r}) = \sum_{i=1}^{n} r_i$.

(a) **[6pts]** Suppose the Web has no dead ends. Prove that $w(\mathbf{r}') = w(\mathbf{r})$.

(b) **[9pts]** Suppose there are still no dead ends, but we use a teleportation probability (i.e., the probability of jumping to some random page) of $1 - \beta$, where $0 < \beta < 1$. The expression for the next estimate of r_i becomes $r'_i = \beta \sum_{j=1}^{n} M_{ij} r_j + (1 - \beta)/n$. Under what conditions will $w(\mathbf{r}') = w(\mathbf{r})$? Prove your conclusion.

Hint: The conditions required for the equality are related to $w(\mathbf{r})$.

(c) **[10pts]** Now, let us assume a teleportation probability of $1 - \beta$ in addition to the fact that there are one or more dead ends. Call a node "dead" if it is a dead end and "live" if not. Assume $w(\mathbf{r}) = 1$. At each iteration, when not teleporting, each live node j distributes r_j PageRank uniformly across each of the nodes it links to, and each dead node j distributes r_j/n PageRank to all the nodes.

Write the equation for r'_i in terms of β, M, $\mathbf{r}$, n, and D (where D is the set of dead nodes). Then, prove that $w(\mathbf{r}') = 1$.

What to submit

(i) Proof [1(a)]

(ii) Condition for $w(\mathbf{r}') = w(\mathbf{r})$ and Proof [1(b)]

(iii) Equation for r'_i and Proof [1(c)]

2 Implementing PageRank and HITS (30 points)

In this problem, you will learn how to implement the PageRank and HITS algorithms in Spark. You will be experimenting with a small randomly generated graph (assume that the graph has no dead-ends) provided in `graph-full.txt`.

There are 100 nodes ($n = 100$) in the small graph and 1000 nodes ($n = 1000$) in the full graph, and $m = 8161$ edges, 1000 of which form a directed cycle (through all the nodes) which ensures that the graph is connected. It is easy to see that the existence of such a cycle ensures that there are no dead ends in the graph. There may be multiple directed edges between a pair of nodes, and your solution should treat them as the same edge. The first column in `graph-full.txt` refers to the source node, and the second column refers to the destination node.

Implementation hint: You may choose to store the PageRank vector r either in memory or as an RDD. Only the matrix of links is too large to store in memory.

(a) PageRank Implementation [15 points]

Assume the directed graph $G = (V, E)$ has n nodes (numbered $1, 2, \ldots, n$) and m edges, all nodes have positive out-degree, and $M = [M_{ji}]_{n\times n}$ is a an $n \times n$ matrix as defined in class such that for any $i, j \in [1, n]$:

$$M_{ji} = \begin{cases} \frac{1}{\deg(i)} & \text{if } (i \to j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Here, $\deg(i)$ is the number of outgoing edges of node i in G. If there are multiple edges in the same direction between two nodes, treat them as a single edge. By the definition of PageRank, assuming $1-\beta$ to be the teleport probability, and denoting the PageRank vector by the column vector r, we have the following equation:

$$r = \frac{1-\beta}{n}\mathbf{1} + \beta M r, \tag{1}$$

where $\mathbf{1}$ is the $n \times 1$ vector with all entries equal to 1.

Based on this equation, the iterative procedure to compute PageRank works as follows:

1. Initialize: $r^{(0)} = \frac{1}{n}\mathbf{1}$
2. For i from 1 to k, iterate: $r^{(i)} = \frac{1-\beta}{n}\mathbf{1} + \beta M r^{(i-1)}$

Run the aforementioned iterative process in Spark for 40 iterations (assuming $\beta = 0.8$) and (the estimate) of the PageRank vector r. In particular, you don't have to implement the blocking algorithm from lecture. The matrix M can be large and should be processed as an RDD in your solution. Compute the following:

- List the top 5 node ids with the highest PageRank scores.
- List the bottom 5 node ids with the lowest PageRank scores.

For a test case, we have provided a smaller dataset (`graph-small.txt`). In that dataset, the top node has id 53 with value 0.036.

(b) HITS Implementation [15 points]

Assume that the directed graph $G = (V, E)$ has n nodes (numbered $1, 2, \ldots, n$) and m edges, all nodes have non-negative out-degree, and $L = [L_{ij}]_{n\times n}$ is a an $n \times n$ matrix referred to as the *link matrix* such that for any $i, j \in [1, n]$:

$$L_{ij} = \begin{cases} 1 & \text{if } (i \to j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Given the link matrix L and scaling factors λ and μ, the hubbiness vector h and the authority vector a can be expressed using the equations:

$$h = \lambda L a, a = \mu L^T h \tag{2}$$

where $\mathbf{1}$ is the $n \times 1$ vector with all entries equal to 1.

Assume that $\lambda = 1, \mu = 1$. Then the iterative method to compute h and a is as follows:

1. Initialize h with a column vector (of size $n \times 1$) of all 1's.
2. Compute $a = L^T h$ and scale so that the largest value in the vector a has value 1.
3. Compute $h = La$ and scale so that the largest value in the vector h has value 1.
4. Go to step 2.

Repeat the iterative process for 40 iterations, and obtain the hubbiness and authority scores of all the nodes (pages). The link matrix L can be large and should be processed as an RDD. Compute the following:

- List the 5 node ids with the highest hubbiness score.
- List the 5 node ids with the lowest hubbiness score.
- List the 5 node ids with the highest authority score.
- List the 5 node ids with the lowest authority score.

For a test case, you should confirm that `graph-small.txt` has highest hubbiness node id 59 with value 1 and highest authority node id 66 with value 1.

What to submit

(i) List 5 node ids with the highest and least PageRank scores [2(a)]

(ii) List 5 node ids with the highest and least hubbiness and authority scores [2(b)]

3 Spectral Clustering (45 points)

We saw in lecture several methods for partitioning different types of graphs. In this problem, we explore (yet another) such partitioning method called "spectral clustering". The name derives from the fact that it uses the *spectrum of certain matrices* (that is, the eigenvalues and eigenvectors) derived from the graph and the key idea is to minimize cut. Mathematically, cut minimization is closely related to modularity maximization. We could also use the spectral clustering algorithm to solve the modularity maximization problem [1].

Our goals in this problem are to derive a simple algorithm for spectral clustering, apply it on a dataset, and interpret the clustering results.

Let us first fix the notation we'll use in this problem.

- Let $G = (V, E)$ be a simple (that is, no self- or multi-edges), undirected, connected graph with $n = |V|$ and $m = |E|$.
- We use the notation $\{i, j\} \in E$ to denote that the nodes i and j are connected via an edge (note that since this is an undirected graph, we do not talk about the direction of the connection).
- Let A be the adjacency matrix of G: that is, $A_{ij} = \begin{cases} 1 & \text{if } \{i,j\} \in E \\ 0 & \text{otherwise} \end{cases}$.
- We use d_i to denote the degree of the i-th node; by definition of the adjacency matrix, $d_i = \sum_{j=1}^{n} A_{ij}$. We define the diagonal matrix D formed by placing the degrees of the nodes along its diagonal. That is, $D_{ii} = d_i$ for all $i = 1, 2, \ldots, n$.
- We define the graph Laplacian as the $n \times n$ matrix $L = D - A$.
- We define a vector $e_i \in \mathbb{R}^n$ as zero on all coordinates *except* the i-th, at which it is 1. In this case, since $|V| = n$, the vector e_i is n-dimensional.
- Define the vector $e \in \mathbb{R}^n$ as the vector of all 1s. Again, e is an n-dimensional vector in this case.

For a set of nodes $S \subseteq V$, we associate two values that measure, in some sense, its quality as a cluster: the "cut" and the "volume". We define these two values below.

The "cut" of a set S is defined as the number of edges that have one end point in the set S and the other in its complement, $\overline{S} = V \backslash S$:

$$\text{cut}(S) = \sum_{i \in S, j \in \overline{S}} A_{ij}.$$

Observe that by definition, $\text{cut}(S) = \text{cut}(\overline{S})$.

The "volume" of a set is defined as the sum of degrees of nodes in S:

$$\text{vol}(S) = \sum_{i \in S} d_i, \tag{3}$$

where d_i is the degree of node i.

In addition to the above measures associated with set S, we define the *normalized cut* of a graph (associated with a partitioning S) as

$$\text{NCUT}(S) = \frac{\text{cut}(S)}{\text{vol}(S)} + \frac{\text{cut}(\overline{S})}{\text{vol}(\overline{S})} \tag{4}$$

For a set S to have a small normalized cut value, it must have very few edges connecting the nodes inside S to the rest of the graph (making the numerators small), *as well as* roughly equal volumes of S and $\overline{S}$, so that neither denominator is too small.

We are now ready to start proving things. **Please be careful when reading expressions involving S and $\overline{S}$, since at a quick glance they may look the same.**

(a) Establishing Some Basics [21 points]

We first make some observations that will help us formulate the problem of minimizing normalized cut nicely in the next sub-problem.

Given a set of nodes S, we define a vector $x_S \in \mathbb{R}^n$, such that the i-th coordinate $x_S^{(i)}$ of x_S is defined as follows:

$$x_S^{(i)} = \begin{cases} \sqrt{\frac{\text{vol}(\overline{S})}{\text{vol}(S)}} & \text{if } i \in S \\ -\sqrt{\frac{\text{vol}(S)}{\text{vol}(\overline{S})}} & \textit{otherwise} \end{cases} \tag{5}$$

To clarify (because the font may not be clear), in Equation 5, in the case $i \in \overline{S}$, the term in the denominator under the square root is vol($\overline{S}$).

In the following, we are using the notation established at the start of this problem and some set of node $S \subseteq V$. Prove the following statements:

1. $x_S^\top L x_S = c \cdot \text{NCUT}(S)$ for some constant c that depends on the problem parameters. Note that you should specify this constant.

   Hint: For any vector $x \in \mathbb{R}^n$, it holds that $x^\top Lx = \sum_{\{i,j\} \in E}(x_i - x_j)^2$. You can start the proof using this hint.

2. $x_S^\top De = 0$.

3. $x_S^\top Dx_S = 2m$.

(b) Normalized Cut Minimization [12 points]

Based on the facts we just proved about x chosen as per Equation 5, we can formulate the normalized cut minimization problem as follows:

$$\begin{array}{ll} \underset{S\subset V}{\text{minimize}} & \frac{x_S^\top Lx_S}{x_S^\top Dx_S} \\ \text{subject to} & x_S^\top De = 0, \\ & x_S^\top Dx_S = 2m. \end{array}$$

To be clear, we are minimizing over all (non-trivial) partitions $(S, \overline{S})$, where the vectors x_S are defined as described in Equation 5. Note that the two constraints appearing in the optimization are trivially maintained due to the form x_S as we have shown in the previous sub-problem. However, constraining x to take the form of Equation 5 makes this optimization problem NP-Hard. We will instead relax the optimization problem, making it tractable, and then round the relaxed solution back to a feasible point of the *original* problem. The relaxation we choose eliminates the constraint on the form of x. This gives us the following *relaxed* optimization problem:

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & \frac{x^\top Lx}{x^\top Dx}, \\ \text{subject to} & x^\top De = 0, \\ & x^\top Dx = 2m. \end{array} \tag{7}$$

We could show that a minimizer of the optimization problem 7 is

$$x^* = D^{-1/2}v,$$

where v is an eigenvector corresponding to the second smallest eigenvalue of the *normalized graph Laplacian* $\mathcal{L} = D^{-1/2}LD^{-1/2}$.

Let's prove this conclusion step by step.

1. Please prove that the smallest eigenvalue of L is 0 and the corresponding eigenvector is e.

2. Please prove the eigenvector corresponding to the smallest eigenvalue of $\mathcal{L}$ (normalized graph Laplacian) is $D^{1/2}e$.

   Hint 1: You can start by proving that $\mathcal{L}$ is PSD and build your solution on that. However, specify your arguments clearly.

   Hint 2: Use the conclusion from part 1.

3. Use the linear transformation $z = D^{1/2}x$ and $r = D^{1/2}e$. This is a valid operation because the graph is connected, and therefore all the degrees are positive. The optimization problem 7, in the transformed space, becomes:

$$\begin{array}{ll} \underset{z\in\mathbb{R}^n}{\text{minimize}} & \frac{z^\top \mathcal{L} z}{z^\top z} \\ \text{subject to} & z^\top r = 0 \\ & z^\top z = 2m. \end{array}$$

   Please prove the solution of this optimization problem is $z^* = v$ and $x^* = D^{-1/2}v$

   Finally, to round the solution back to a feasible point in the original problem, we can take the nodes corresponding to the positive entries of the eigenvector to be in the set S and those corresponding to the negative entries to be in $\overline{S}$.

   Hint 1: For a symmetric matrix P, we can write any vector z as $z = \sum_{i=1}^n w_i v_i$, where the v_i are orthonormal eigenvectors of P.

   Hint 2: Use the conclusion from part 2.

(c) Programming [12 points]

Now that we have obtained the new transformed version of our normalized cut minimization problem and its solution, we will implement the spectral clustering algorithm for $k = 2$ clusters.

We have provided the dataset `songs.csv` in the folder `spectral_clustering/data`, which includes 10 features (e.g., `release`, `loudness`, `danceability`) for 1000 songs. The features are already normalized. In this question, you are going to implement spectral clustering and run it on the songs dataset to cluster these songs into two clusters. Then, you are going to investigate these inferred clusters and identify features that distinguish the two clusters.

There are various methods for defining an adjacency matrix. For this question, we are going to generate a simple adjacency matrix where two songs i, j are connected (i.e., $A_{ij} = 1$) only if their Euclidean distance is smaller than 1. The adjacency matrix A for this dataset is defined as follows:

$$A_{ij} = \begin{cases} 1 & \text{if } \|x_i - x_j\| < 1 \\ 0 & \textit{otherwise} \end{cases} \tag{6}$$

Note that here we are adopting a simple version of a general method called ϵ-neighbourhood graph which connects nodes (i.e., points) within ϵ distance of each other. Specifically, we are defining a 1-neighbourhood graph ($\epsilon = 1$) for our data matrix.

The algorithm for $k = 2$ clusters is as follows:

Algorithm 1 Spectral Clustering for $k = 2$

```
procedure Spectral Clustering
    Construct the adjacency matrix $A \in \mathbb{R}^{n \times n}$
    Compute the degree matrix $D$
    Compute the Laplacian $L = D - A$ and the normalized Laplacian $\mathcal{L} = D^{-1/2} L D^{-1/2}$
    Compute the eigenvector $v$ corresponding to the second smallest eigenvalue of $\mathcal{L}$ and compute $x = D^{-1/2} v \in \mathbb{R}^n$
    Let $x_i$ be the $i^{th}$ element of $x$ for $i = 1, \ldots, n$
    for each $x_i$ do
        Cluster $x_i$ using the indicator function $I$, as defined in Equation 7, into clusters $C_1, C_2$
    end for
end procedure
```

Most spectral clustering algorithms would use k-means algorithm in the last step. Here, for simplicity, we use an indicator function instead. The indicator function is defined as follows:

$$I(i) = \begin{cases} i \in C_1 & \text{if } x_i \geq 0 \\ i \in C_2 & otherwise \end{cases} \tag{7}$$

1. Please report the final cluster labels for the first 5 songs. For eigendecomposition, you may use library functions for obtaining the eigenvalues and eigenvectors of a matrix (e.g., `numpy.linalg.eig` in Python).

   Hint: For test case, the clustering results for songs 6, 7, 8 are C_1, C_1, C_1.

2. Now we start to explore the systematic differences between the two clusters. One simple way is to compare the mean difference of each of the 10 original features between the two clusters. In other words, for the two clusters you learned, calculate the mean feature values across all samples in that cluster. Once you have this mean representation for each cluster, calculate the difference between the mean features of the two clusters.

   Please report the top 3 features with the highest absolute value of mean difference. By looking at the features, briefly explain (1-2 sentences) what these clusters might represent (any reasonable answer is correct).

3. Simply looking at the mean features for each cluster is useful yet quite limited. An alternative way of investigating the learned clusters is to run t-tests for each feature to see whether the differences between the two clusters are statistically significant.

   For the 3 features you found in part 2, please run t-tests and report the p-values for each of the 3 features. If you apply a threshold of p-value=0.05, which of these 3 features are still significant? Briefly explain (1-2 sentences) why you think this is the case.

   For running t-tests, please use the library function `scipy.stats.ttest_ind(equal_var = True)` in Python.

What to submit

i. Proof of the 3 equalities in part 3(a).

ii. Proof of the 3 conclusions in part 3(b).

iii. The clustering result for the first 5 users [3(c)].

iv. The top 3 features with the highest absolute value of mean difference and interpretation of the results [3(c)].

v. The p-values for the 3 features, the list of significant features, and interpretation of the results [3(c)].

vi. Upload your implementation of the spectral clustering algorithm to Gradescope.

References

[1] Newman, M. E. Spectral methods for community detection and graph partitioning. *Physical Review E 88*, 4 (2013), 042822.