CS2241 Assignment 1

Nico Fidalgo

March 26, 2025

1 Problem 1: PageRank and HITS Algorithm Analysis

1.1 Problem Statement

We are given a directed graph represented by the following adjacency matrix.

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{1}$$

We need to calculate the PageRank scores and Hub and Authority scores. The interpretation of the adjacency matrix is:

- Node a has outgoing edges to nodes c, d, and f
- Node b has outgoing edges to nodes c and f
- Node c has outgoing edges to nodes d and f
- Node d has outgoing edges to nodes b and c
- Node e has an outgoing edge to node a
- Node f has an outgoing edge to node e

1.2 PageRank Algorithm

PageRank was designed to model the behavior of a random surfer on the web. The algorithm assigns a score to each node in a directed graph based on its structural importance. The intuition is that a node is important if many important nodes point to it and the importance of a node is divided among its outgoing links.

$$\mathbf{p} = (1-d) \cdot \frac{\mathbf{1}}{n} + d \cdot M \cdot \mathbf{p} \tag{2}$$

where:

- d is the damping factor (typically 0.85)
- n is the number of nodes
- $\mathbf{1}$ is a vector of all 1's
- M is the transition matrix (columns sum to 1)

For each node i with outgoing links, we set:

$$M_{j,i} = \frac{1}{\text{out-degree}(i)} \tag{3}$$

if there is a link from node i to node j, and 0 otherwise. The algorithm for computing PageRank scores is:

Algorithm 1 PageRank Score Calculation

```
Initialize p^(0) = (1/n) · 1
for t = 0, 1, 2, ... until convergence do
    p^(t+1) = (1 − d) · (1/n) + d · M · p^(t)
    if ||p^(t+1) − p^(t)|| < tolerance then
        break
    end if
end for
Normalize p to sum to 1
```

1.3 PageRank Implementation

I wrote a JavaScript program to implement the PageRank algorithm:

```javascript
import * as math from 'mathjs';

// Adjacency matrix from the problem
const A = [
    [0, 0, 1, 1, 0, 1],  // a -> *
    [0, 0, 1, 0, 0, 1],  // b -> *
    [0, 0, 0, 1, 0, 1],  // c -> *
    [0, 1, 1, 0, 0, 0],  // d -> *
    [1, 0, 0, 0, 0, 0],  // e -> *
    [0, 0, 0, 0, 1, 0]   // f -> *
];

// Number of nodes
const n = A.length;

// Out-degrees
```

```javascript
const out_degrees = A.map(row => row.reduce((sum, val) => sum + val, 0));
console.log("Out-degrees:", out_degrees);

// Create transition matrix (column-stochastic)
const M = Array(n).fill().map(() => Array(n).fill(0));
for (let i = 0; i < n; i++) {
    for (let j = 0; j < n; j++) {
        if (A[i][j] > 0) {
            M[j][i] = 1.0 / out_degrees[i];
        }
    }
}

// PageRank parameters
const d = 0.85;  // Damping factor
const max_iter = 100;
const tol = 1e-6;

// Initialize PageRank
let pr = Array(n).fill(1/n);

// Algorithm iteration
for (let iter = 0; iter < max_iter; iter++) {
    // Calculate M * pr
    const M_pr = Array(n).fill(0);
    for (let i = 0; i < n; i++) {
        for (let j = 0; j < n; j++) {
            M_pr[i] += M[i][j] * pr[j];
        }
    }

    // Calculate (1-d)/n + d * (M * pr)
    const pr_new = M_pr.map(val => (1-d)/n + d * val);

    // Check convergence
    const diff = math.norm(pr_new.map((val, idx) => val - pr[idx]));
    if (diff < tol) {
        pr = pr_new;
        console.log(`\nPageRank converged after ${iter+1} iterations.`);
        break;
    }

    pr = pr_new;
}

// Normalize to sum to 1
const pr_sum = pr.reduce((sum, val) => sum + val, 0);
pr = pr.map(val => val / pr_sum);
```

1.4 PageRank Results

After running the algorithm, I obtained the following PageRank scores:

Node	PageRank Score
a	0.186551
b	0.091079
c	0.182643
d	0.155479
e	0.190060
f	0.194188

Nodes f, e, and a have the highest PageRank scores, indicating they are structurally important in this network. Node b has the lowest score, suggesting it's less central in the graph's link structure.

1.5 HITS Algorithm

The Hyperlink-Induced Topic Search (HITS) algorithm identifies two types of important nodes in a directed graph: hubs which are nodes that point to many good authorities, and authorities which are nodes that are pointed to by many good hubs. The hub ($\mathbf{h}$) and authority ($\mathbf{a}$) vectors satisfy:

$$\mathbf{a} = A^T\mathbf{h} \tag{4}$$

$$\mathbf{h} = A\mathbf{a} \tag{5}$$

where A is the adjacency matrix of the graph. The algorithm for computing HITS scores is:

Algorithm 2 HITS Score Calculation

1: Initialize $\mathbf{h}^{(0)} = \mathbf{1}$ and $\mathbf{a}^{(0)} = \mathbf{1}$
2: **for** $t = 0, 1, 2, \ldots$ until convergence **do**
3: $\quad \mathbf{a}^{(t+1)} = A^T\mathbf{h}^{(t)}$
4: $\quad$ Normalize $\mathbf{a}^{(t+1)}$
5: $\quad \mathbf{h}^{(t+1)} = A\mathbf{a}^{(t+1)}$
6: $\quad$ Normalize $\mathbf{h}^{(t+1)}$
7: $\quad$ **if** $\|\mathbf{a}^{(t+1)} - \mathbf{a}^{(t)}\| <$ tolerance and $\|\mathbf{h}^{(t+1)} - \mathbf{h}^{(t)}\| <$ tolerance **then**
8: $\quad\quad$ **break**
9: $\quad$ **end if**
10: **end for**

1.6 HITS Implementation

Again, I wrote a JavaScript program to implement the HITS algorithm:

```
import * as math from 'mathjs';

// Adjacency matrix from the problem
const A = [
    [0, 0, 1, 1, 0, 1],  // a -> *
    [0, 0, 1, 0, 0, 1],  // b -> *
```

```
    [0, 0, 0, 1, 0, 1],  // c -> *
    [0, 1, 1, 0, 0, 0],  // d -> *
    [1, 0, 0, 0, 0, 0],  // e -> *
    [0, 0, 0, 0, 1, 0]   // f -> *
];

// Number of nodes
const n = A.length;

// HITS parameters
const max_iter = 100;
const tol = 1e-6;

// Initialize hub and authority scores
let hub = Array(n).fill(1);
let auth = Array(n).fill(1);

// Compute transpose of A
const AT = Array(n).fill().map(() => Array(n).fill(0));
for (let i = 0; i < n; i++) {
    for (let j = 0; j < n; j++) {
        AT[i][j] = A[j][i];
    }
}

// HITS iteration
for (let iter = 0; iter < max_iter; iter++) {
    // Update authority scores: a = A^T * h
    const auth_new = Array(n).fill(0);
    for (let i = 0; i < n; i++) {
        for (let j = 0; j < n; j++) {
            auth_new[i] += AT[i][j] * hub[j];
        }
    }

    // Normalize authority scores
    const auth_norm = math.norm(auth_new);
    const auth_normalized = auth_new.map(val => val / auth_norm);

    // Update hub scores: h = A * a
    const hub_new = Array(n).fill(0);
    for (let i = 0; i < n; i++) {
        for (let j = 0; j < n; j++) {
            hub_new[i] += A[i][j] * auth_normalized[j];
        }
    }

    // Normalize hub scores
    const hub_norm = math.norm(hub_new);
    const hub_normalized = hub_new.map(val => val / hub_norm);

    // Check convergence
    const auth_diff = math.norm(auth_normalized.map((val, idx) => val - auth[idx]));
    const hub_diff = math.norm(hub_normalized.map((val, idx) => val - hub[idx]));

```

```
    if (auth_diff < tol && hub_diff < tol) {
        auth = auth_normalized;
        hub = hub_normalized;
        console.log(`HITS converged after ${iter+1} iterations.`);
        break;
    }

    auth = auth_normalized;
    hub = hub_normalized;
}
```

1.7 HITS Results

After running the HITS algorithm, I obtained the following scores:

Node	Hub Score	Authority Score
a	0.684439	0.000000
b	0.501536	0.113935
c	0.446890	0.590796
d	0.283360	0.454889
e	0.000000	0.000000
f	0.000000	0.656548

Nodes a, b, and c have high hub scores, indicating they are good at pointing to authority nodes. This makes sense as a and b point to multiple nodes including high authority nodes c and f. Nodes e and f have zero hub scores because they don't point to any nodes with high authority scores.

Nodes f and c have the highest authority scores, followed by node d. This means they are pointed to by good hub nodes. Again, this makes sense as node f and c are pointed to by nodes a and b which have high hub scores. Nodes a and e have zero authority scores because they aren't pointed to by any good hub nodes.

1.8 Assumptions

For PageRank scores:

- Used a damping factor of 0.85 (standard value)
- Defined convergence as when L2 norm difference $< 10^{-6}$

For HITS:

- Defined convergence as when L2 norm difference $< 10^{-6}$

2 Problem 2: Random Walk Methods for Hub and Authority Scores

2.1 Problem Statement

We need to prove that the Hub score for a page is proportional to the number of outlinks and that the Authority score is proportional to the number of inlinks when using the following random walk approach:

For Authority scores:

- From page p_1, follow back a random inlink to page p_2
- From p_2, follow forward a random outlink to page p_3
- The step takes us from p_1 to p_3

For Hub scores:

- From page p_1, follow forward a random outlink to page p_2
- From p_2, follow backward a random inlink to page p_3
- The step takes us from p_1 to p_3

We assume the Markov chains are finite, irreducible, and aperiodic, ensuring a unique stationary distribution.

2.2 Definitions

- $A[i, j] = 1$ if there's a link from page i to page j, 0 otherwise
- $\text{in}(j)$ = number of inlinks to page $j = \sum_i A[i, j]$
- $\text{out}(i)$ = number of outlinks from page $i = \sum_j A[i, j]$

2.3 Transition Probabilities

For Authority Random Walk:
When starting at page i:

- The probability of following a random inlink back to page k is $\frac{A[k,i]}{\text{in}(i)}$
- The probability of following a random outlink from k to j is $\frac{A[k,j]}{\text{out}(k)}$

Therefore, the transition probability from i to j is:

$$P_a(i, j) = \sum_k \frac{A[k, i]}{\text{in}(i)} \times \frac{A[k, j]}{\text{out}(k)} \tag{6}$$

For Hub Random Walk:
When starting at page i:

- The probability of following a random outlink to page k is $\frac{A[i,k]}{\text{out}(i)}$
- The probability of following a random inlink back from k to j is $\frac{A[j,k]}{\text{in}(k)}$

Therefore, the transition probability from i to j is:

$$P_h(i,j) = \sum_k \frac{A[i,k]}{\text{out}(i)} \times \frac{A[j,k]}{\text{in}(k)} \tag{7}$$

2.4 Authority Score Proof

Let's hypothesize that the stationary distribution $\pi_a(i)$ is proportional to $\text{in}(i)$, i.e., $\pi_a(i) = c \times \text{in}(i)$ for some constant c.

For this to be a stationary distribution, it must satisfy:

$$\pi_a(j) = \sum_i \pi_a(i) P_a(i,j) \tag{8}$$

Substituting our hypothesis:

$$\pi_a(j) = \sum_i c \times \text{in}(i) \times \sum_k \frac{A[k,i]}{\text{in}(i)} \times \frac{A[k,j]}{\text{out}(k)} \tag{9}$$

$$= c \times \sum_i \sum_k \frac{A[k,i] \times A[k,j]}{\text{out}(k)} \tag{10}$$

$$= c \times \sum_k \frac{A[k,j]}{\text{out}(k)} \times \sum_i A[k,i] \tag{11}$$

Since $\sum_i A[k,i] = \text{out}(k)$ (the number of outlinks from page k):

$$\pi_a(j) = c \times \sum_k \frac{A[k,j]}{\text{out}(k)} \times \text{out}(k) \tag{12}$$

$$= c \times \sum_k A[k,j] \tag{13}$$

$$= c \times \text{in}(j) \tag{14}$$

This confirms the hypothesis that the Authority score $\pi_a(j)$ is proportional to $\text{in}(j)$, the number of inlinks to page j.

2.5 Hub Score Proof

Similarly, hypothesize that the stationary distribution $\pi_h(i)$ is proportional to $\text{out}(i)$, i.e., $\pi_h(i) = d \times \text{out}(i)$ for some constant d.

For this to be a stationary distribution, it must satisfy:

$$\pi_h(j) = \sum_i \pi_h(i) P_h(i,j) \tag{15}$$

Substituting our hypothesis:

$$\pi_h(j) = \sum_i d \times \text{out}(i) \times \sum_k \frac{A[i,k]}{\text{out}(i)} \times \frac{A[j,k]}{\text{in}(k)} \tag{16}$$

$$= d \times \sum_i \sum_k \frac{A[i,k] \times A[j,k]}{\text{in}(k)} \tag{17}$$

$$= d \times \sum_k \frac{A[j,k]}{\text{in}(k)} \times \sum_i A[i,k] \tag{18}$$

Since $\sum_i A[i,k] = \text{in}(k)$ (the number of inlinks to page k):

$$\pi_h(j) = d \times \sum_k \frac{A[j,k]}{\text{in}(k)} \times \text{in}(k) \tag{19}$$

$$= d \times \sum_k A[j,k] \tag{20}$$

$$= d \times \text{out}(j) \tag{21}$$

This confirms the hypothesis that the Hub score $\pi_h(j)$ is proportional to $\text{out}(j)$, the number of outlinks from page j.