Machine Learning with Graphs
Homework 1

Tobias Kallehauge

Aalborg University, November 28, 2021

1 Link Analysis

Before answering the questions - I will give some theory.
The personalised PageRank equation is given by:

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{\mathbb{1}(j \in I)}{|I|} \tag{1}$$

where $\mathbb{1}$ is the indicator function and I is the interest set. On matrix form:

$$\mathbf{r} = \beta \mathbf{M}\mathbf{r} + (1-\beta)\mathbf{p}, \tag{2}$$

where $\mathbf{p}_j = \mathbb{1}(j \in I)/|I|$. Rewriting (2), I get:

$$\underbrace{(\mathbf{I} - \beta \mathbf{M})}_{\mathbf{X}}\mathbf{r} = \underbrace{(1-\beta)\mathbf{p}}_{\mathbf{y}} \Rightarrow \mathbf{X}\mathbf{r} = \mathbf{y} \tag{3}$$

Notice that the matrix $\mathbf{X}$ is independent of the interest set I. This formulation lends the following theorem:

Theorem 1. *Let $I_1, \dots, I_n$ be interest sets with known associated page rank vectors $\mathbf{r_1}, \dots, \mathbf{r}_n$ such that*

$$\mathbf{r}_i = \beta \mathbf{M}\mathbf{r}_i + (1-\beta)\mathbf{p}_i, \quad i = 1, \dots, n$$

are known. Then the PageRank vector $\mathbf{r}$ for any interest set I is given by a linear combination of $\mathbf{r_1}, \dots, \mathbf{r}_n$ if $\mathbf{p} \in Span(\mathbf{p}_1, \dots, \mathbf{p}_i)$. Furthermore, given the coefficients $a_1, \dots, a_n$ to the solution

$$\mathbf{p} = \sum_{i=1}^{n} a_i \mathbf{p}_i, \tag{4}$$

the PageRank vector is given by:

$$\mathbf{r} = \sum_{i=1}^{n} a_i \mathbf{r}_i,$$

Proof. Given that $\mathbf{p} \in \text{Span}(\mathbf{p}_1, \dots, \mathbf{p}_i)$, there exists coefficients $a_1, \dots, a_n$ such that (4) is fulfilled. Using the notation in (3) and multiplying both sides of (4) with $(1-\beta)$, I get

$$\begin{aligned}
\mathbf{y} &= \sum_{i=1}^{n} a_i \mathbf{y}_i, \Leftrightarrow \\
\mathbf{X}\mathbf{r} &= \sum_{i=1}^{n} a_i \mathbf{X}\mathbf{r}_i, \Leftrightarrow \\
\mathbf{r} &= \sum_{i=1}^{n} a_i \mathbf{r}_i,
\end{aligned}$$

where second equation uses (3) and the third multiplies both sides with $\mathbf{X}^{-1}$ assuming that it exists. □

The proof relies on $\mathbf{X}^{-1}$ existing. I suspect that it can be shown without this assumption. The $\mathbf{p}$ vectors for the interest sets given in the homework are:

$$\mathbf{p}_A = \frac{1}{3}\begin{bmatrix}1\\1\\1\\0\\0\end{bmatrix}, \quad \mathbf{p}_B = \frac{1}{3}\begin{bmatrix}0\\0\\1\\1\\1\end{bmatrix}, \quad \mathbf{p}_C = \frac{1}{3}\begin{bmatrix}1\\0\\0\\1\\1\end{bmatrix} \quad \mathbf{p}_D = \begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix} \tag{5}$$

1.1 Personalized PageRank I

I have

$$\mathbf{p}_E = \begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix} = 3\mathbf{p}_A - 3\mathbf{p}_B + 3\mathbf{p}_C + 2\mathbf{p}_D \tag{6}$$

Using Theorem 1 i get then get $\mathbf{r}_E = 3\mathbf{r}_A - 3\mathbf{r}_B + 3\mathbf{r}_C + 2\mathbf{r}_D$

1.2

Since $\mathbf{p}_E = \begin{bmatrix}0 & 0 & 0 & 0 & 1\end{bmatrix}^T$ is not in the span of the vectors for the known interrest sets, it is not possible to find $\mathbf{r}_E$ using the known PageRank vectors and theorem 1.

1.3

Since

$$\mathbf{p}_G = \begin{bmatrix}0.1\\0.2\\0.3\\0.2\\0.2\end{bmatrix} = 0.6\mathbf{p}_A - 0.3\mathbf{p}_B + 0.3\mathbf{p}_C - 0.2\mathbf{p}_D \tag{7}$$

we get $\mathbf{r}_E = 0.6\mathbf{r}_A - 0.3\mathbf{r}_B + 0.3\mathbf{r}_C - 0.2\mathbf{r}_D$

1.4 Personalized PageRank II

Answered by theorem 1.

1.5 A different equation for PageRank

We need to show that:

$$\mathbf{r} = \mathbf{A}\mathbf{r} \Leftrightarrow \mathbf{r} = \beta\mathbf{M}\mathbf{r} + \frac{1-\beta}{N}\mathbf{1} \tag{8}$$

for $\mathbf{A} = \beta\mathbf{M} + \frac{1-\beta}{N}\mathbf{1}\mathbf{1}^T$ when $\sum_{i=1}^{N}\mathbf{r}_i = 1$.

Proof. The statement (8) is true if

$$\mathbf{A}\mathbf{r} = \beta\mathbf{M}\mathbf{r} + \frac{1-\beta}{N}\mathbf{1} \tag{9}$$

for $\sum_{i=1}^{N}\mathbf{r}_i = 1$. This is indeed the case since

$$\mathbf{1}^T\mathbf{r} = \begin{bmatrix}1 & \dots & 1\end{bmatrix}\begin{bmatrix}r_1\\\vdots\\r_N\end{bmatrix} = \sum_{i=1}^{N} r_i = 1, \tag{10}$$

thus

$$\begin{aligned}\mathbf{Ar} &= \left(\beta\mathbf{M} + \frac{1-\beta}{N}\mathbf{1}\mathbf{1}^T\right)\mathbf{r} \\ &= \beta\mathbf{M}\mathbf{r} + \frac{1-\beta}{N}\mathbf{1}\underbrace{\mathbf{1}^T\mathbf{r}}_{=1} \\ &= \beta\mathbf{M}\mathbf{r} + \frac{1-\beta}{N}\mathbf{1} \end{aligned} \tag{11}$$

□

2 Relational Classification I

I implemented the algorithm using the `networkx` module in Python. After two steps I get

2.1

$$P(Y_3 = +) \approx 0.6328 \tag{12}$$

2.2

$$P(Y_4 = +) \approx 0.4707 \tag{13}$$

2.3

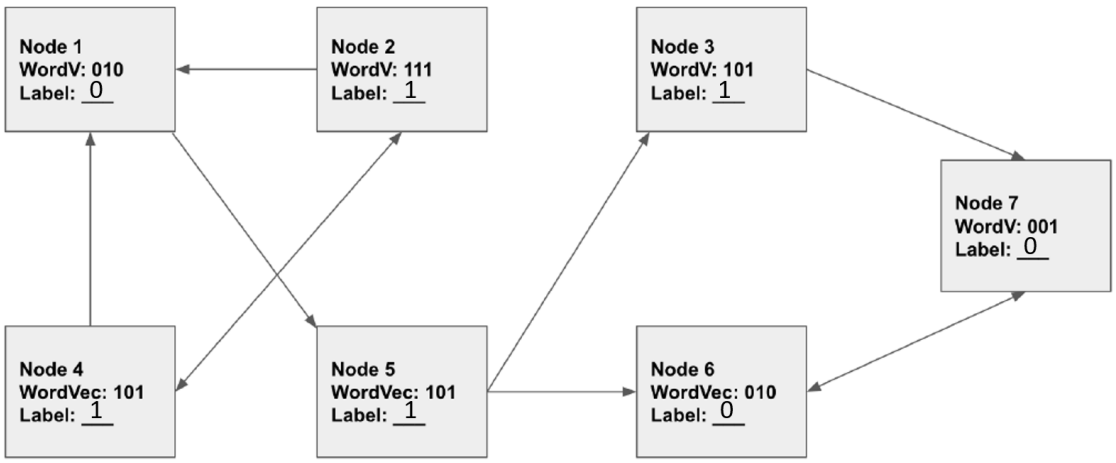
$$P(Y_8 = +) \approx 0.3564 \tag{14}$$

2.4

The following nodes belong to the "−" class after 2 iterations: $[2, 4, 5, 8, 9]$.

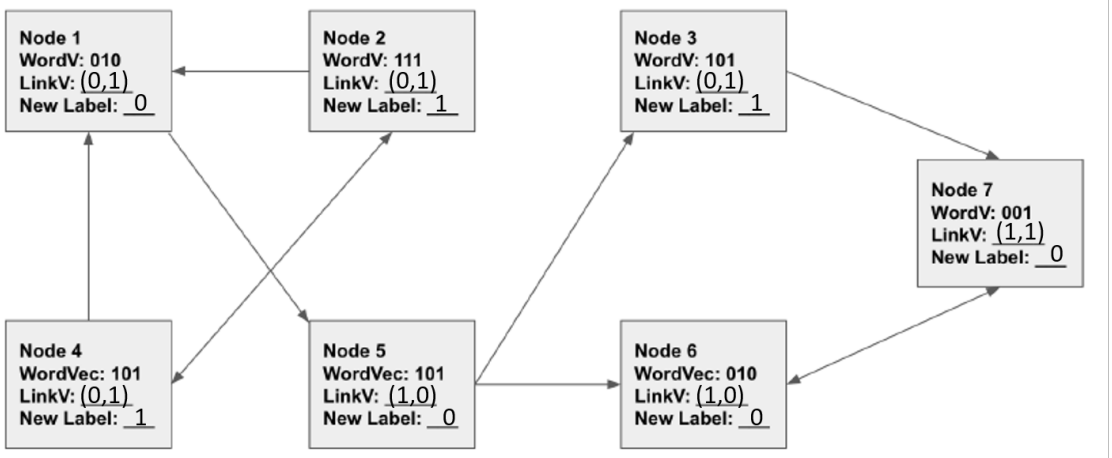
3 Relational Classification II

3.1 Bootstrap Phase

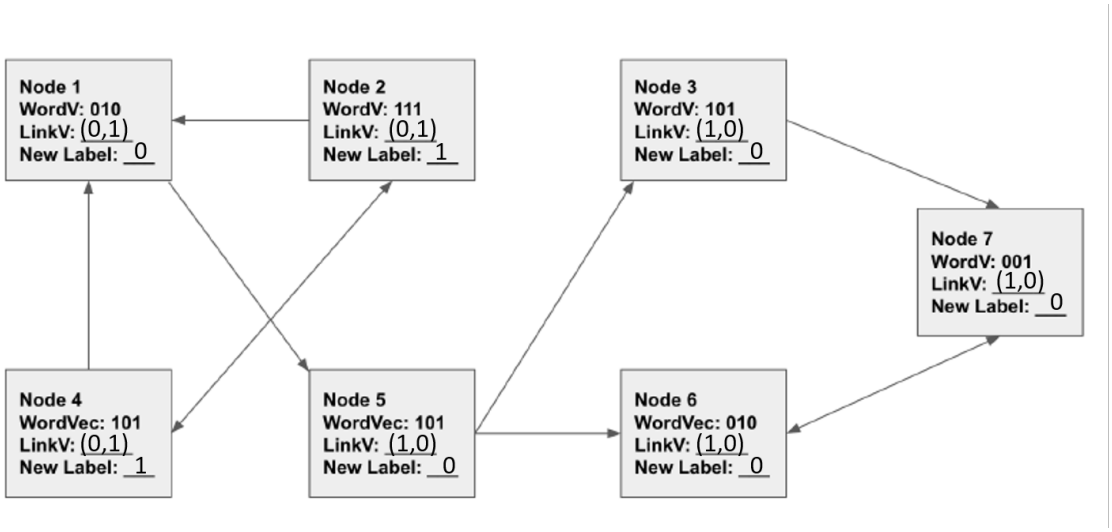


3.2 Iteration 1

I apply g function for the nodes in order $i = 1, 2, \ldots, 7$. I use the new label as input for the link vector if possible, otherwise I use the label from the bootstrap phase. I get the following.



3.3 Iteration 2



3.4 Convergence

Yes, the graph has converged since:

- Node 1,6 have words 010 and will always be 0 independent of their neighbours.
- Node 5,3,7 only have incoming 0 so they will stay at 0
- Node 2,4 do not have incoming 0 or the codeword 010 so they will stay at 1

Thus the graph will not change at the next iteration and it has converged.

4 GNN Expressiveness

4.1 Effect of Depth on Expressiveness

Counting from layer $k = 0$ where the embeddings are all 1, the two nodes are different at layer $k = 3$ (18 and 17 respectively).

4.2 Random walk matrix

i $\mathbf{M} = \begin{bmatrix} 0 & 1/2 & 0 & 1/3 \\ 1/2 & 0 & 0 & 1/3 \\ 0 & 0 & 0 & 1/3 \\ 1/2 & 1/2 & 1 & 0 \end{bmatrix}$

ii $r = \begin{bmatrix} 0.47 \\ 0.47 \\ 0.24 \\ 0.71 \end{bmatrix}$

4.3 Relation to Random Walk (i)

The adjacency matrix $\mathbf{A}$ describe random walks of length 1 although it is unnormalised. Multiplying $\mathbf{A}$ with the inverse degree matrix from the left will normalise along its rows which gives the transition matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$. Multiplying from the right would give the column stochastic matrix $\mathbf{M} = \mathbf{A}\mathbf{D}^{-1}$, which is normalised along the columns.

4.4 Relation to Random Walk (ii)

The transition matrix with the introduced skip connection is:

$$\mathbf{P} = \frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{A}\mathbf{D}^{-1} \tag{15}$$

where $\mathbf{I}$ is the identity matrix of size corresponding to the number of nodes.

4.5 Over-Smoothing Effect

The transition of states $\mathbf{h}^{(l)} \to \mathbf{h}^{(l+1)}$ is characterised by the transition $\mathbf{h}^{(l+1)} = \mathbf{P}\mathbf{h}^{(l)}$. This is the setup in the Markov chain convergence theorem which states that this process will converge if the equivalent Markov Chain is:

- Irreducible (All states commute) **True** if all nodes are connected to the graph with edges.
- Positive recurrent (The expected number of steps to visit the same node is finite): **True** by construction of the process if the graph is finite.
- Aperiodic: There are no "periods" in the graph that always visit a node at some (non-unit) integer values of steps: usually **True** when the graph is undirected but not necessarily the case.

Combining the three properties the graph is refereed to as *ergodic* which will be the case for most graphs of sufficient complexity, e.g., the graph in question 4.2 where there exists the paths $1 \to 2 \to 1$ of length 2 and $1 \to 2 \to 4 \to 1$ of length 3 which makes the transition process aperiodic for all states.

This only proves that $\mathbf{h}^{(l)}$ converges, not that it converges to the same value for all the nodes although this intuitively make sense under the equation $h_i^{(l+1)} = \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} = h_j^{(l)}$ (I did not prove this).

4.6 Learning BFS with GNN

The message, aggregation and update step can be done with the following rule:

$$\mathbf{h}_{l+1} = \sigma\left((\mathbf{A} + \mathbf{I})\mathbf{h}_l\right),$$

where $\mathbf{A}$ is the adjacency matrix and

$$\sigma(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases}$$

is the Heaviside step function.

5 Node Embedding and its relation to matrix factorization

5.1 Simple matrix factorization

Given to columns $\mathbf{z}_v, \mathbf{z}_u$ from $\mathbf{Z}$, the decoder is the similarity of the nodes v, u as a function of the embeddings which is simply $\mathbf{z}_v^T\mathbf{z}_u \approx \mathbf{A}_{v,u}$

5.2 Alternate matrix factorization

To promote $\mathbf{z}_v^T \mathbf{W} \mathbf{z}_u \approx \mathbf{A}_{v,u}$ the objective function should be:

$$\left\| \mathbf{A} - \mathbf{Z}^T \mathbf{W} \mathbf{Z} \right\|_{\ell_2} \tag{16}$$

5.3 Relation to eigendecomposition

When the graph is undirected, the adjacency matrix $\mathbf{A}$ is symmetric and its eigenvalue decompositon is given by:

$$\mathbf{A} = \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^T \tag{17}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with the eigenvalues of $\mathbf{A}$ and $\mathbf{Q}$ are the corresponding eigenvectors. Setting $\mathbf{W} = \mathbf{\Lambda}$ in the matrix factorization, will learn the eigenvectors. $\mathbf{A}$ have to be non-defect (full rank) for the factorization to be exact which also gives the dimension of the embedding is equal to the number of nodes.

5.4 Multi-hop node similarity

To define the new measure of similarity we construct the matrix:

$$\mathbf{N}^{(k)} = \sum_{i=1}^{k} \mathbf{A}^k, \tag{18}$$

where $\mathbf{N}_{u,v}$ is the number of walks between node u and v of any lengths up till length k. Applying the Haviside to this we get $\mathbf{B} = \sigma(\mathbf{N})$ and the matrix factorization problem becomes:

$$\left\| \mathbf{B} - \mathbf{Z}^T \mathbf{W} \mathbf{Z} \right\|_{\ell_2} \tag{19}$$

5.5 Limitation of node2vec (i)

Since the left and right cliques of the graph are identical, the random walks (over the edges) experienced at both sides should also be similar and I would expect that the vector embedding would reflect the structural similarity for both cliques although the optimization algorithm might converge to two different embeddings — one for each clique. Thus the embeddings between the two cliques will properly be different despite having the same structure.

5.6 Limitation of node2vec (ii)

With node2vec, you can reach all the nodes in the right part of the graph with one step. For the struct2vec, you can reach the left part of the graph too assuming $g_1(w, u) > 1$ for nodes u on the right side which indeed seems reasonable since the left and right part of the graph are structurally similar.

5.7 Limitation of node2vec (iii)

By changing k in the random walk, we consider graphlets of different sizes thus sampling large range of possible local structures. Having k constant would only express graphlet structures of the same size.

5.8 Limitation of node2vec (iv)

I would expect a similar embedding for nodes in the two cliques since we jump to any node in either the current clique or other clique with the same probability. So the random walks between nodes within the two cliques are indistinguishable.