Homework-1 solutions

CS224W (Fall 2021)

Karan Bania*
Machine Learning Department
Carnegie Mellon University
kbania@andrew.cmu.edu

1 Link Analysis

Before getting to the answers, let us recall that the Personalized-PageRank (PPR) equation **for user** s is (Slide numbers 36 and 53 from slide 4 here)-

$$r_{s_t} = \beta M r_{s_{t-1}} + (1-\beta)T_s$$

where, r_{s_t} is a vector of dimension $\mathbb{R}^N$ (N is the number of nodes), $M(\in \mathbb{R}^{N\times N})$ is the transition matrix, β is the given teleport parameter, and $T_s(\in \mathbb{R}^N)$ is the teleport vector, i.e., the vector that makes the algorithm "personalized". For example, for the user A, $T_A = \frac{1}{3}[1, 1, 1, 0, \dots]$. It is important to note that, T_s **does not** depend on time, so,

$$r_{s_{t+1}} = \beta M r_{s_t} + (1-\beta)T_s$$
$$r_{s_{t+1}} = \beta M(\beta M r_{s_{t-1}} + (1-\beta)T_s) + (1-\beta)T_s$$

Solving these equations we can get to an expression for r_{s_t} for a user s, which is -

$$r_{s_t} = \beta^t M^t r_0 + (1-\beta)(\sum_{k=0}^{t}(\beta M)^k)T_s$$
$$r_{s_t} = a + b \times T_s$$

*I solved them while I was an undergraduate at BITS Pilani, Goa Campus.

Now, given that the teleport parameter is the same for all users and we have ran the algorithm up to some time t, we can see a **linear dependence** on T_s (we also assume that all users have the same base distribution r_0). Thus, if we denote the PPR vectors for uses A, B, C & D as v_A, v_B, v_C & v_D respectively, then we can perform vector manipulations & find the PPR vectors for users with different teleport sets.

$$\begin{aligned} A &= \{1,2,3\}, \\ B &= \{3,4,5\}, \\ C &= \{1,4,5\}, \\ D &= \{1\}. \end{aligned}$$

Different users will only differ in their T_ss so we need to perform operations to change that.

To solve the next three questions, we need to find v_x for some x whose T_s has been given to us, i.e.,

$$v_x = a \times v_A + b \times v_B + c \times v_C + d \times v_D$$

and then solve for a, b, c & d (This is exactly the **vector space** formed by v_A, v_B, v_C & v_D).

This can be further simplified to

$$v_{x_1} = a/3 + c/3 + d \quad (1)$$
$$v_{x_2} = a/3 \quad (2)$$
$$v_{x_3} = a/3 + b/3 \quad (3)$$
$$v_{x_4} = b/3 + c/3 \quad (4)$$
$$v_{x_5} = b/3 + c/3. \quad (5)$$

This already tells us that if $v_{x_4} \neq v_{x_5}$, then we cannot find the PPR vector.

1.1 Personalized PageRank I

Yes, we **can** do this, solving (1),

$$v_{\text{Eloise}} = v_{\{2\}} = 3 * v_A - 3 * v_B + 3 * v_C - 2 * v_D$$

1.2

We **cannot** compute $v_{\text{Felicity}} = v_{\{5\}}$ from the given information. As we cannot isolate the contribution of of 5, or formally, 5 does not lie in the vector space formed by the basis vectors v_A, v_B, v_C & v_D.

1.3

Yes, we **can** do this, solving (1),

$$v_{\text{Glynnis}} = 0.6 * v_A + 0.3 * v_B + 0.3 * v_C - 0.2 * v_D$$

1.4 Personalized PageRank II

Clearly, it is the **vector space defined by vectors in** V.

1.5 A different equation for PageRank

We have to prove,

$$\mathbf{r} = (\beta \mathbf{M} + \frac{(1-\beta)}{N} \mathbf{1}\mathbf{1}^T)\mathbf{r}$$

is equivalent to,

$$\mathbf{r} = \beta \mathbf{M} r + \frac{(1-\beta)}{N} \mathbf{1}$$

OR,

$$\mathbf{1}^T \mathbf{r} = 1$$

We also know that $\mathbf{r}$ is normalized, i.e., $\sum_{i=1}^{N} \mathbf{r_i} = 1$; clearly,

$$\mathbf{1}^T \mathbf{r} = \sum_{i=1}^{N} \mathbf{r_i} = 1.$$

2 Relational Classification I

2.1

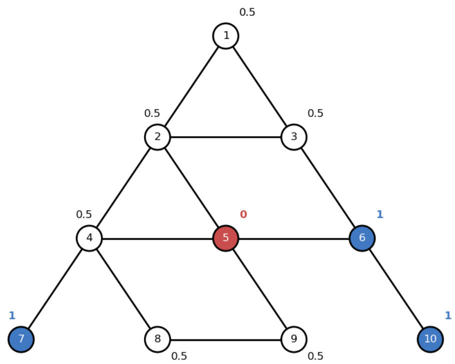
$P(Y_3 = +) \approx 0.633$

2.2

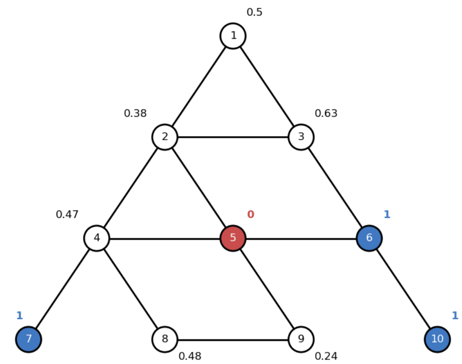
$P(Y_4 = +) \approx 0.471$

2.3

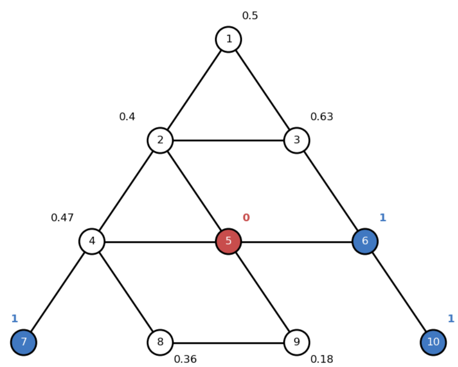
$P(Y_8 = +) \approx 0.356$



(a) $i = 0$



(b) $i = 1$



(c) $i = 2$

Figure 1: Propagation of probabilities.

2.4

Nodes that are '-' after second iteration: 1,2,4,5,8,9

3 Relational Classification II

Just for clarity, these are our functions,

$$g(\texttt{WordV}, \texttt{LinkV}) = \begin{cases} 0 & \text{if } I_0 = 1 \text{ or } \texttt{WordV} = 010 \\ 1 & \text{otherwise} \end{cases}$$

`WordV`	f(`WordV`)
001	0
010	0
101	1
111	1

Table 1: Function definition for f.

3.1 Bootstrap Phase

As we just use f for this phase, the labels are as in 2.

Node	Label
1	0
2	1
3	1
4	1
5	1
6	0
7	0

Table 2: Labels after bootstrap phase.

3.2 Iteration 1

Results in Tables 4 and 3.

Node	`LinkV` (I_0, I_1)
1	(0, 1)
2	(0, 1)
3	(0, 1)
4	(0, 1)
5	(1, 0)
6	(1, 1)
7	(1, 1)

Table 3: `LinkV` for nodes.

Node	Label
1	0
2	1
3	1
4	1
5	0
6	0
7	0

Table 4: Labels using g after iteration 1.

3.3 Iteration 2

Only label for node 5 has changed from 1 to 0, so it can only affect `LinkV` for nodes it is pointing to, i.e., Node 3. Results in tables 6 and 5.

Node	`LinkV` (I_0, I_1)
1	(0, 1)
2	(0, 1)
3	(1, 0)
4	(0, 1)
5	(1, 0)
6	(1, 1)
7	(1, 1)

Table 5: `LinkV` for nodes.

Node	Label
1	0
2	1
3	0
4	1
5	0
6	0
7	0

Table 6: Labels using g after iteration 2.

3.4 Convergence

The only label that changed during the last iteration was for Node 3, so only Node 7's `LinkV` can change now, which also doesn't change, and so the labels **have converged**.

4 GNN Expressiveness

4.1 Effect of Depth on Expressiveness

For this sub-question, our update rule is -

$$h_v^{k+1} = h_v^k + \sum_{i \in \mathcal{N}_v} h_i^k$$

where $\mathcal{N}_v$ is the neighbourhood of node v, and k is the layer number. I will just use a recurrence instead of drawing graphs as I think that is much simpler. I will also assume that the nodes have been labelled from $A - F$ in an anti-clockwise manner and the top most node is named T (for both graphs).

k=0 doesn't work, because that means no propagation and these nodes have the same initial feature vector = [1].

k=1 also doesn't work, because these nodes have the same one-hop neighbourhood, which is the node A.

k=2, will also not work, because they have the same 2-hop neighbourhood (nodes A, B and F), but, I will show this case an example. We need to compute $h_T^{(2)}$ for both graphs, i.e.,

$$h_T^{(2)} = h_T^{(1)} + \sum_{i \in \mathcal{N}_T} h_i^{(1)}$$

. *Left Graph*,

$$\begin{aligned} h_T^{(2)} &= h_T^{(1)} + \sum_{i \in \mathcal{N}_T} h_i^{(1)} \\ &= (h_T^{(0)} + h_A^{(0)}) + h_A^{(1)} \\ &= [2] + (h_A^{(0)} + h_B^{(0)} + h_T^{(0)} + h_F^{(0)}) \\ &= [6] \end{aligned}$$

This will be the same for the *Right Graph*.

k=3, will work, we need to compute $h_T^{(3)}$ for both graphs, i.e.,

$$h_T^{(3)} = h_T^{(2)} + h_A^{(2)}$$

. *Left Graph*,

$$\begin{aligned} h_T^{(3)} &= h_T^{(2)} + h_A^{(2)} \\ &= [6] + (h_A^{(1)} + h_B^{(1)} + h_T^{(1)} + h_F^{(1)}) \\ &= [6] + (h_A^{(0)} + h_B^{(0)} + h_T^{(0)} + h_F^{(0)}) \\ &+ (h_B^{(0)} + h_A^{(0)} + h_C^{(0)}) + (h_T^{(0)} + h_A^{(0)}) \\ &+ (h_A^{(0)} + h_F^{(0)} + h_E^{(0)}) \\ &= [\mathbf{18}] \end{aligned}$$

Right Graph

$$\begin{aligned} h_T^{(3)} &= h_T^{(2)} + h_A^{(2)} \\ &= [6] + (h_A^{(1)} + h_B^{(1)} + h_T^{(1)} + h_F^{(1)}) \\ &= [6] + (h_A^{(0)} + h_B^{(0)} + h_T^{(0)} + h_F^{(0)}) \\ &+ (h_B^{(0)} + h_A^{(0)} + h_C^{(0)}) + (h_T^{(0)} + h_A^{(0)}) \\ &+ (h_A^{(0)} + h_F^{(0)}) \\ &= [\mathbf{17}] \end{aligned}$$

The only difference was Node F having vs. not having a connection to Node E.

4.2 Random Walk Matrix

i M is the transition matrix, we just have to do the following, if there is an edge from $i \to j$ then $M_{ji} = \frac{1}{d_i}$. The matrix then becomes,

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

ii Let $r = [a, b, c, d]^T$, we need to solve $Mr = r$ or, $(M - I)r = 0$. This will give us the following system of linear equations

$$\begin{aligned} -a + b/2 + d/3 &= 0 \\ a/2 - b + d/3 &= 0 \\ -c + d/3 &= 0 \\ a/2 + b/2 + c - d &= 0 \end{aligned}$$

Solving this we can get $r = d * [2/3, 2/3, 1/3, 1]^T$. However, we also know $r * r^T = 1$ (r is normalized), this gives $d = \frac{1}{\sqrt{2}}$. So the final answer is

$$r = \begin{bmatrix} \sqrt{2}/3 \\ \sqrt{2}/3 \\ 1/(\sqrt{2} * 3) \\ 1/\sqrt{2} \end{bmatrix}$$

4.3 Relation to Random Walk (i)

The transition matrix is $\mathbf{D^{-1}A}$. This matrix allows for an uniformly random step, this can be seen by the fact that A will be one for all connections of a node and D^{-1} will weigh all of them equally.

4.4 Relation to Random Walk (ii)

The transition matrix for this case is $\mathbf{D^{-1}}(\frac{\mathbf{A+I}}{\mathbf{2}})$. The intuition behind I is that it adds self loops, and we want to *distribute* the probability accordingly.

4.5 Over-Smoothening Effect

Following the hint, we can model our state-transitions as a Markov Chain; also I think because $h_v^{(l)}$ is not a probability distribution, we will stick to the random jump convention and prove that $\mathbf{r}$ converges. Clearly if $\mathbf{r}$ converges, then so does $\lim_{l\to\infty} h_v^{(l)}$.

For a small and apt introduction to Markov Chains / the convergence theorem, I will redirect the interested reader to [1].

Again we will follow the hint, so, is our markov chain *irreducible*? **yes**. Because the graph is **connected** and there **no** bipartite components, there is always a non-zero probability of the random walk ending up at any vertex v.

And now is it *aperiodic*? **yes**. I do not have a very formal argument for why this holds, but if we start with one probability for some node x and zero probability for all others, then $\forall t > 0$, we can be at node x with non-zero probability, so we can be there for $t = 2$ and $t = 3$, so clearly gcd $\mathcal{T}(x) = 1$, and our chain is **aperiodic**.

Given both of these conditions, we can be assured that the Markov Chain will converge, and thus $\lim_{l\to\infty} h_v^{(l)}$ will also converge.

4.6 Learning BFS with GNN

The intuition is that suppose we are at node j and one of our neighbours node i has a one as it's feature vector at layer l, then we should also have a one as our feature vector at layer $l + 1$. In view of this I define the following `Message`, `Aggregate` and `Update` functions.

$$\begin{aligned}
\texttt{Message}(h_u^{(k)}, h_v^{(k)}, e_{u,v}) &= h_u^{(k)} \\
\texttt{Aggregate}(\{h_u^{(k-1)} | \forall u \in \mathcal{N}_v\}) &= max(\{h_u^{(k-1)} | \forall u \in \mathcal{N}_v\}) \\
\texttt{Update}(h_v^{(k)}, h_{\mathcal{N}_v}^{(k)}) &= max(h_v^{(k)}, h_{\mathcal{N}_v}^{(k)})
\end{aligned}$$

where $h_{\mathcal{N}_v}^{(k)} = \texttt{Aggregate}(\{h_u^{(k-1)} | \forall u \in \mathcal{N}_v\})$. The max in `Update` is necessary because otherwise the source node might lose it's one, i.e., the task will not be learned perfectly.

5 Node Embedding and its relation to matrix factorization

5.1 Simple matrix factorization

The decoder is the *inner product decoder*, i.e., $Z^T Z$. I would also like to point out **why** this is a decoder in the first place, because it confused me for quite some time, we can think of this as **generating** the A matrix and so it is rightly called a decoder.

5.2 Alternate matrix factorization

The objective then becomes,

$$\mathtt{min}_Z ||A - Z^T W Z||_2$$

Although this seems trivial, this adds more expressive power because of W.

5.3 Relation to eigendecomposition

Note that, the eigendecomposition of A will be of the form,

$$A = Q \Lambda Q^{-1}$$

as A is symmetric, $Q^{-1} = Q^T$. So the condition on W would be that it should be diagonal.

5.4 Multi-hop node similarity

The objective then becomes,

$$\mathtt{min}_Z || \sum_{l=0}^{k} A^l - Z^T Z||_2$$

This follows because A^l will have 1s in places where there is a path of length exactly l, and we want **at least** one path of length **at most** k.

5.5 Limitations of node2vec (i)

No, node2vec will fail to capture any form of structural similarity because it is highly unlikely that a node from clique-2 (say the right one) will ever appear on a random walk starting from clique-1 (the left one).

5.6 Limitations of node2vec (ii)

For *node2vec*, we can only go to w's neighbours (clique-2 only). For *strcut2vec*, we can go to any node in the graph.

5.7 Limitations of node2vec (iii)

There are two reasons, first off, if we don't have multiple g_ks, and we have also made the graph a clique, then we have imposed the wrong structure on the model and it won't learn anything useful (we will be jumping around in the wrong graph). Secondly, having more g_ks helps the model with expressivity, i.e., more structural information.

5.8 Limitations of node2vec (iv)

I am not sure how to formally define this but any two nodes in the two cliques, share the same neighbourhood up till 6 hops away (the white node), thus they will be extremely similar in the embedding space.

References

[1] A. Freedman. Convergence theorem for finite markov chains. *Proc. Reu*, 2017.