HW5

Mengying Lin SID: 3038737132.

1. Directed and Undirected Graphs

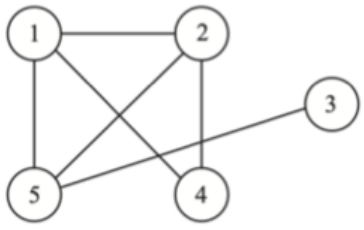


Figure 1: Simple Undirected Graph

Figure 1 shows a simple undirected graph whose adjacency matrices we want to make sure you can write down. Generally, an unnormalized adjacency matrix between the nodes of a directed or undirected graph is given by:

$$A_{i,j} = \begin{cases} 1 : \text{if there is an edge between node i and node j,} \\ 0 : \text{otherwise.} \end{cases} \tag{1}$$

This will be a symmetric matrix for undirected graphs. For a directed graph, we have:

$$A_{i,j} = \begin{cases} 1 : \text{if there is an edge from node i to node j,} \\ 0 : \text{otherwise.} \end{cases} \tag{2}$$

This need not to be symmetric for a directed graph, and is in fact typically not a symmetric matrix when we are thinking about directed graphs (otherwise, we'd probably be thinking of them as undirected graphs).

Similarly, the degree matrix of an undirected graph is a diagonal matrix that contains information about the degree of each vertex. In other words, it contains the number of edges attached to each vertex and it is given by:

$$D_{i,j} = \begin{cases} deg(v_i) : \text{if i == j,} \\ 0 : \text{otherwise.} \end{cases} \tag{3}$$

where the degree $\deg(v_i)$ of a vertex counts the number of times an edge terminates at that vertex.

Sometimes, imbalanced weights may undesirably affect the matrix spectrum (eigenvalues and eigenvectors). This occurs when a vertex with a large degree results in a large diagonal entry in the Laplacian matrix dominating the matrix properties. To solve that issue, a normalization scheme is applied which aims to make the influence of such vertices more equal to that of other vertices, by dividing the entries of the Adjacency matrix by the vertex degrees.

In that sense, a normalized adjacency matrix is given by:

$$A^{Normalized} = AD^{-1} \tag{4}$$

and a symmetrically normalized adjacency matrix is given by

$$A^{SymNorm} = D^{-1/2}AD^{-1/2} \tag{5}$$

Given a simple graph G with n vertices $v_1, ..., v_n$, its unnormalized Laplacian matrix $L_{n \times n}$ is defined element-wise as:

$$L_{i,j} = \begin{cases} deg(v_i) : \text{if i == j}, \\ -1 : \text{if i != j and } v_i \text{ is adjacent to } v_j, \\ 0 : \text{otherwise.} \end{cases} \tag{6}$$

or equivalently by the matrix:

$$L = D - A \tag{7}$$

where D is the degree matrix and A is the adjacency matrix of the graph.

We could also compute the symmetrically normalized Laplacian which is inherited from the adjacency matrix normalization scheme as shown below:

$$L^{SymNorm} = I - A^{SymNorm} \tag{8}$$

(a) **Show that $L^{SymNorm}$ could also be written as**:

$$L^{SymNorm} = D^{-1/2} L D^{-1/2} \tag{9}$$

where D is the dregree matrix, and L is the unnormalized Laplacian.

a) pf: $L = D - A = D^{\frac{1}{2}}(I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) D^{\frac{1}{2}}$.

$= D^{\frac{1}{2}}(I - A^{SymNorm}) D^{\frac{1}{2}}$

$= D^{\frac{1}{2}} L^{SymNorm} D^{\frac{1}{2}}$.

$L^{SymNorm} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$. Q.E.D.

(b) **Write the unnormalized adjacency A, the degree matrix, D, and the symmetrically normalized adjacency matrix, $A^{SymNorm}$, of the graph in Figure. 1.**

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix} \qquad D = \begin{pmatrix} 3 & & & & \\ & 3 & & & 0 \\ & & 1 & & \\ & & & 2 & \\ 0 & & & & 3 \end{pmatrix}$$

$$A^{\text{SymNorm}} = \begin{pmatrix} 0 & \frac{1}{3} & 0 & \frac{1}{\sqrt{6}} & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 & \frac{1}{\sqrt{6}} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{\sqrt{3}} & 0 & 0 \end{pmatrix}$$

:) **Write the symmetrically normalized Laplacian matrix of the graph in Figure. 1.**

$$L^{\text{SymNorm}} = I - A^{\text{SymNorm}}$$

$$= \begin{pmatrix} 1 & -\frac{1}{3} & 0 & -\frac{1}{\sqrt{6}} & -\frac{1}{3} \\ -\frac{1}{3} & 1 & 0 & -\frac{1}{\sqrt{6}} & -\frac{1}{3} \\ 0 & 0 & 1 & 0 & -\frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & 0 & 1 & 0 \\ -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{\sqrt{3}} & 0 & 1 \end{pmatrix}$$

(d) **Compute A^2, A^3**

We now want to estimate the traffic flow of inner downtown Berkeley and we know the road network shown below. The goal of the estimation is to estimate the traffic flow on eache road segment. The flow estimates should satisfy the conservation of vehicles exactly at each intersection as indicated by the arrows.

d) $A^2 =$

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 3 & 2 & 1 & 1 & 1 \\ 2 & 3 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 2 & 2 \\ 1 & 1 & 0 & 2 & 3 \end{pmatrix}$$

$$A^3 = \begin{pmatrix} 3 & 2 & 1 & 1 & 1 \\ 2 & 3 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 2 & 2 \\ 1 & 1 & 0 & 2 & 3 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 4 & 5 & 1 & 5 & 6 \\ 5 & 4 & 1 & 5 & 6 \\ 1 & 1 & 0 & 2 & 3 \\ 5 & 5 & 2 & 2 & 2 \\ 6 & 6 & 3 & 2 & 2 \end{pmatrix}$$

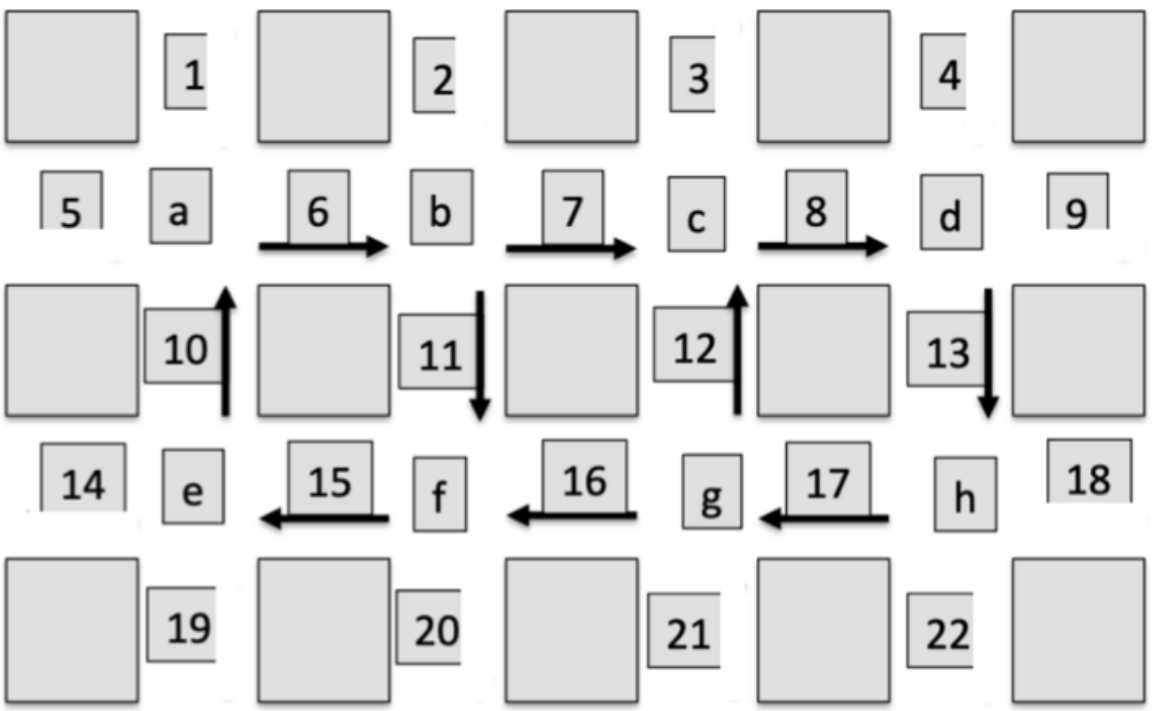


Figure 2: Simple Directed Graph

The intersections are labeled a to h. The road segments are labeled 1 to 22. The arrows indicate the direction of traffic.

Hint: think about the best way to represent the road network in terms of matrices, vectors, etc.

(e) **Write the unnormalized adjacency matrix of the graph in Figure 2.**

let intersections be the vertices.

The unnormalized adjacency matrix is as followed:

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

(f) **Write the In-degree D_{in} and Out-degree D_{out} matrix of the graph in Figure. 2.**

$$D_{in} = \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 2 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & 2 & \\ & & & & & & 1 \\ & & & & & & & 1 \end{pmatrix} \qquad D_{out} = \begin{pmatrix} 1 & & & & & & \\ & 2 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & 2 & \\ & & & & & & 1 \\ & & & & & & & 1 \end{pmatrix}$$

(g) **Write both of the symmetrically normalized In-degree and Out-degree Laplacian matrix of the graph in Figure. 2.**

$$A_{in}^{SymNorm} = \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \frac{1}{\sqrt{2}} & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & \frac{1}{\sqrt{2}} & \\ & & & & & & 1 \\ & & & & & & & 1 \end{pmatrix} \begin{pmatrix} 0&1&0&0&0&0&0 \\ 0&0&1&0&0&1&0 \\ 0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0 \\ 0&0&1&0&0&1&0 \\ 0&0&0&0&0&0&1 \end{pmatrix} \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \frac{1}{\sqrt{2}} & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & \frac{1}{\sqrt{2}} & \\ & & & & & & 1 \\ & & & & & & & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0&1&0&0&0&0&0 \\ 0&0&\frac{1}{\sqrt{2}}&0&0&\frac{1}{\sqrt{2}}&0 \\ 0&0&0&\frac{1}{\sqrt{2}}&0&0&0 \\ 0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0 \\ 0&0&0&\frac{1}{\sqrt{2}}&0&0&0 \\ 0&0&\frac{1}{\sqrt{2}}&0&0&\frac{1}{\sqrt{2}}&0 \\ 0&0&0&0&0&0&1 \end{pmatrix}$$

$$L_{in}^{SymNorm} = I - A_{in}^{SymNorm} = \begin{pmatrix} 1&-1&0&0&0&0&0 \\ 0&1&-\frac{1}{\sqrt{2}}&0&0&-\frac{1}{\sqrt{2}}&0 \\ 0&0&1&-\frac{1}{\sqrt{2}}&0&0&0 \\ 0&0&0&1&0&0&-1 \\ -1&0&0&0&1&0&0 \\ 0&0&0&-\frac{1}{\sqrt{2}}&1&0&0 \\ 0&0&-\frac{1}{\sqrt{2}}&0&0&-\frac{1}{\sqrt{2}}&1&0 \\ 0&0&0&0&0&0&-1&1 \end{pmatrix}$$

$$A_{out}^{SymNorm} = \begin{pmatrix} 1 & & & & & & \\ & \frac{1}{\sqrt{2}} & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & \frac{1}{\sqrt{2}} & \\ & & & & & & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & & & & & & \\ & \frac{1}{\sqrt{2}} & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & \frac{1}{\sqrt{2}} & \\ & & & & & & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$I_{out}^{SymNorm} = I - A_{out}^{SymNorm}$$

$$= \begin{pmatrix} 1-\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1-\frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1-1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{\sqrt{2}} & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & 1 \end{pmatrix}$$

2. Graph Dynamics

Some graph neural network methods operate on the full adjacency matrix. Others, such as those discussed here https://distill.pub/2021/gnn-intro/, at each layer apply the same local operation to each node based on inputs from its neighbors.

This problem is designed to:

- show connections between these methods.
- show that for a positive integer k, the matrix A^k has an interesting interpretation. That is, the entry in row i and column j gives the number of walks of length k (i.e., a collection of k edges) leading from vertex i to vertex j.

To do this, let's consider a very simple deep linear network that is built on an underlying graph with n vertices. In the 0-th layer, each node has a single input with weight 1 that is fed a one-hot encoding of its own identity — so node i in the graph has a direct input which is an $n-$dimensional vector that has a 1 in position i and 0s in all other positions. You can view these as n channels if you want.

The weights connecting node i in layer k to node j in layer $k+1$ are simply 1 if vertices i and j are connected in the underlying graph and are 0 if those vertices are not connected in the underlying graph. At each layer, the operation at each node is simply to sum up the weighted sum of its inputs and to output the resulting n-dim vector to the next layer. You can think of these as being depth-wise operations if you'd like.

(a) Let A be the $n \times n$ size adjacency matrix for the underlying graph where the entry $A_{i,j} = 1$ if vertices i and j are connected in the graph and 0 otherwise. **Write the output of the j-th node at layer k in this network in terms of the matrix A.**
(Hint: This output is an n-dimensional vector since there are n output channels at each layer.)

a) Input matrix is I_n.

The output of kth layer is $A^k I_n = A^k$.

The jth node in kth layer is the jth row of A^k.

(b) Here is some helpful notation: Let $V(i)$ be the set of vertices that are connected to vertex i in the graph. Let $L_k(i, j)$ be the number of distinct paths that go from vertex i to vertex j in the graph where the number of edges traversed in the path is exactly k. Recall that a path from i to j in a graph is a sequence of vertices that starts with i, ends with j, and for which every successive vertex in the sequence is connected by an edge in the graph. The length of the path is 1 less than the number of vertices in the corresponding sequence. **Show that the i-th output of node j at layer k in the network above is the count of how many paths there are from i to j of length k**, where by convention there is exactly 1 path of length 0 that starts at each node and ends up at itself.
(Hint: Can applying induction on k help?)

b) Prove by induction.

Define the given statement as P(k).

① when k=0. the output is I_n, which denotes the paths

starting at each node and ending up at the same one.

P(0) holds true.

② Assume P(t) is true for a given $t \geq 0$.

$L_0(i,j) = (A^t I_n)_{i,j} = A^t_{i,j}$

$L_{t+1}(i,j) = (A \cdot A^t)_{i,j} = \sum_{k=1}^{n} (A_{i,k} \cdot A^t_{k,j}) = \sum_{k=1}^{n} (A_{i,k} \cdot L_t(k,j))$

$A_{i,k} = 1$ if i and k are connected.

Therefore $A_{i,k} \cdot L_t(k,j)$ indicates the number of path that go through i and k in order, and then traverse to j in t steps. Hence $L_{t+1}(i,j)$ indicates the number of path that go from i to j in t+1 steps.

P(t+1) holds true.

③ For an arbitrary k, the statement P(k) is true.

Q.E.D.

(c) The structure of the neural network in this problem is compatible with a straightforward linear graph neural network since the operations done (just summing) are locally permutation-invariant at the level of each node and can be viewed as essentially doing the exact same thing at each vertex in the graph based on inputs coming from its neighbors. This is called "aggregation" in the language of graph neural nets. In the case of the computations in previous parts, **what is the update function that takes the aggregated inputs from neighbors and results in the output for this node?**

$f(x) = Ax$

(d) The simple GNN described in the previous parts counts paths in the graph. If we were to replace sum aggregation with max aggregation, **what is the interpretation of the outputs of node j at layer k?**

If there exists a path from i to j with steps k, then the output is 1. Otherwise, it's 0.

4. Graph Neural Networks

For an undirected graph with no labels on edges, the function that we compute at each layer of a Graph Neural Network must respect certain properties so that the same function (with weight-sharing) can be used at different nodes in the graph. Let's focus on a single particular "layer" ℓ. For a given node i in the graph, let $\mathbf{s}_i^{\ell-1}$ be the self-message (i.e. the state computed at the previous layer for this node) for this node from the preceeding layer, while the preceeding layer messages from the n_i neighbors of node i are denoted by $\mathbf{m}_{i,j}^{\ell-1}$ where j ranges from 1 to n_i. We will use w with subscripts and superscripts to denote learnable scalar weights. If there's no superscript, the weights are shared across layers. Assume that all dimensions work out.

(a) **Tell which of these are valid functions for this node's computation of the next self-message $\mathbf{s}_i^{\ell}$. For any choices that are not valid, briefly point out why.**

Note: we are *not* asking you to judge whether these are useful or will have well behaved gradients. Validity means that they respect the invariances and equivariances that we need to be able to deploy as a GNN on an undirected graph.

(i) $\mathbf{s}_i^{\ell} = w_1 \mathbf{s}_i^{\ell-1} + w_2 \frac{1}{n_i} \sum_{j=1}^{n_i} \mathbf{m}_{i,j}^{\ell-1}$

(ii) $\mathbf{s}_i^{\ell} = \max(w_1^{\ell} \mathbf{s}_i^{\ell-1}, w_2 \mathbf{m}_{i,1}^{\ell-1}, w_3 \mathbf{m}_{i,2}^{\ell-1}, \ldots, w_{n_i-1} \mathbf{m}_{i,n_i}^{\ell-1})$ where the max acts component-wise on the vectors.

(iii) $\mathbf{s}_i^{\ell} = \max(w_1^{\ell} \mathbf{s}_i^{\ell-1}, w_2 \mathbf{m}_{i,1}^{\ell-1}, w_2 \mathbf{m}_{i,2}^{\ell-1}, \ldots, w_2 \mathbf{m}_{i,n_i}^{\ell-1})$ where the max acts component-wise on the vectors.

a) (i) (iii) are valid and (ii) is not because this function is not permutation-invariant on neighbors.

(b) We are given the following simple graph on which we want to train a GNN. The goal is binary node classification (i.e. classifying the nodes as belonging to type 1 or 0) and we want to hold back nodes 1 and 4 to evaluate performance at the end while using the rest for training. We decide that the surrogate loss to be used for training is the average binary cross-entropy loss.

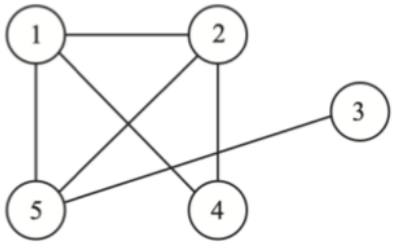


Figure 3: Simple Undirected Graph

nodes	1	2	3	4	5
y_i	0	1	1	1	0
$\hat{y}_i$	a	b	c	d	e

Table 1: y_i is the ground truth label, while $\hat{y}_i$ is the predicted probability of node i belonging to class 1 after training.

Table 1 gives you relevant information about the situation.

Compute the training loss at the end of training.

Remember that with n training points, the formula for average binary cross-entropy loss is

$$\frac{1}{n}\sum_x \left(y(x)\log\frac{1}{\hat{y}(x)} + (1-y(x))\log\left(\frac{1}{1-\hat{y}(x)}\right)\right)$$

where the x in the sum ranges over the training points and $\hat{y}(x)$ is the network's predicted probability that the label for point x is 1.

loss = $-\frac{1}{5}(\log(1-a) + \log b + \log c + \log d + \log(1-e))$

(c) Suppose we decide to use the following update rule for the internal state of the nodes at layer ℓ.

$$\mathbf{s}_i^\ell = \mathbf{s}_i^{\ell-1} + W_1 \frac{\sum_{j=1}^{n_i} \tanh\left(W_2 \mathbf{m}_{i,j}^{\ell-1}\right)}{n_i} \tag{12}$$

where the tanh nonlinearity acts element-wise.

For a given node i in the graph, let $\mathbf{s}_i^{\ell-1}$ be the self-message for this node from the preceeding layer, while the preceeding layer messages from the n_i neighbors of node i are denoted by $\mathbf{m}_{i,j}^{\ell-1}$ where j ranges from 1 to n_i. We will use W with subscripts and superscripts to denote learnable weights in matrix form. If there's no superscript, the weights are shared across layers.

(i) **Which of the following design patterns does this update rule have?**

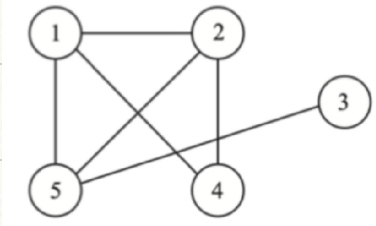
- ☑ Residual connection
- ☐ Batch normalization

(ii) **If the dimension of the state $\mathbf{s}$ is d-dimensional and W_2 has k rows, what are the dimensions of the matrix W_1?**

(iii) If we choose to use the state $\mathbf{s}_i^{\ell-1}$ itself as the message $\mathbf{m}^{\ell-1}$ going to all of node i's neighbors, **please write out the update rules corresponding to (12) giving $\mathbf{s}_i^\ell$ for the graph in Figure 3 for nodes $i = 2$ and $i = 3$ in terms of information from earlier layers.** Expand out all sums.

ii) $d \times k$.

iii).



$s_2^l = s_2^{l-1} + W_1 \frac{\tanh W_2 s_1^{l-1} + \tanh W_2 s_4^{l-1} + \tanh W_2 s_5^{l-1}}{3}$

$s_3^l = s_3^{l-1} + W_1(\tanh W_2 s_4^{l-1})$

7. Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student!
We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

(a) **What sources (if any) did you use as you worked through the homework?**

(b) **If you worked with someone on this homework, who did you work with?**
List names and student ID's. (In case of homework party, you can also just describe the group.)

(c) **Roughly how many total hours did you work on this homework? Write it down here where you'll need to remember it for the self-grade form.**

(b) Name : Xiang Fei

SID : 3038733024.

(c) 15h