Homework 3

Jose Rebolledo Oyarce

November 15, 2022

Problem 1:

Part A

We can rewrite this propagation rule using the Adjacency Matrix (A), however, we need to normalize this matrix. For that, we can include the self-loop, as:

$$\widetilde{A} = A + I$$

where I is the identity matrix. Then we can normalize symmetrically $\widetilde{A}$ using the degree of this matrix and using a diagonal matrix that contains that values:

$$\hat{A} = D^{-1/2} \cdot \widetilde{A} \cdot D^{-1/2}$$

where $D_{ii} = \sum_j \widetilde{A}_{ij}$

So, the propagation rule at graph level is:

$$H^k = \sigma\left(D^{-1/2} \cdot \widetilde{A} \cdot D^{-1/2} \cdot H^{k-1} W^{k-1}\right)$$

Part B

I implemented a class called GCN that initialize 3 variables: *in_features*, *out_features*, *W*, where the first and the second correspond to the dimension of input and output of our data. And the last term corresponding to the weight values that we are going to change in the optimization process.

Then, in the internal function, it is performed the update of the hidden layer using the dot production between tensor and it is used ReLU as the activation function.

Part C

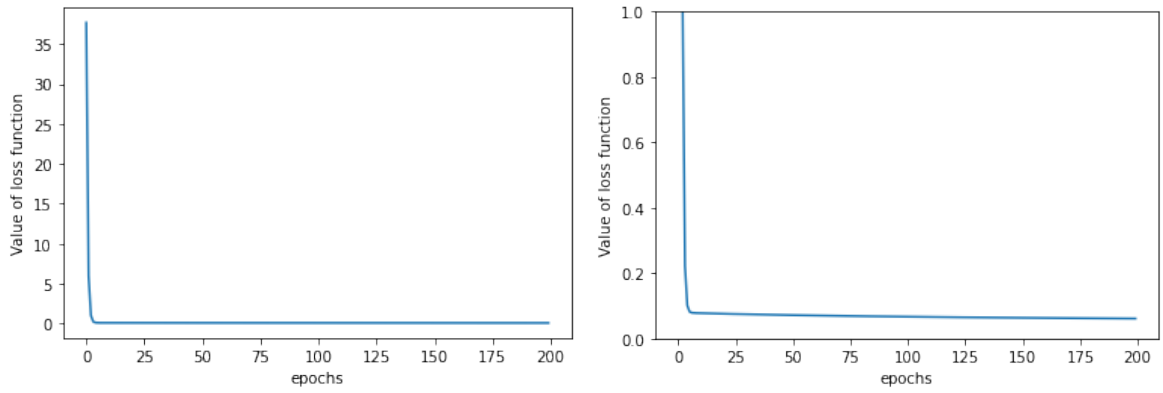
The Pooling layer it is implemented to reduce the dimension of the output of one hideen layer. In the function we sum all values in the row to collapse the value in one dimension.

Part D

In the model level, I initialized the hidden layer (one input hidden layer, and one fully connected layer) and the pooling layer. The inside of this function I normalized the adjacency matrix and then I generated the update layer using the hidden or pooling layer.

Part E

Using 5000 point to training our data and using mean squared error as loss function I obtain the following loss function values vs the epochs



(a) (b)

Figure 1: Evolution of the loss function: (a) complete evolution of the loss function, and (b) close up of the evolution of the loss function

Part F

Then using 1000 to testing our model, we obtain a loss function equal to 0.004

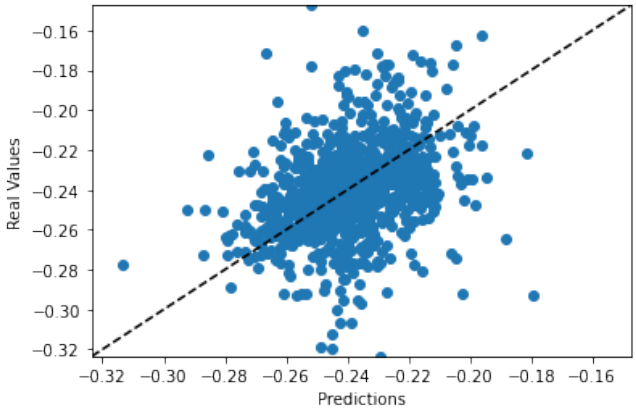


Figure 2: Parity Plot of prediction vs real values

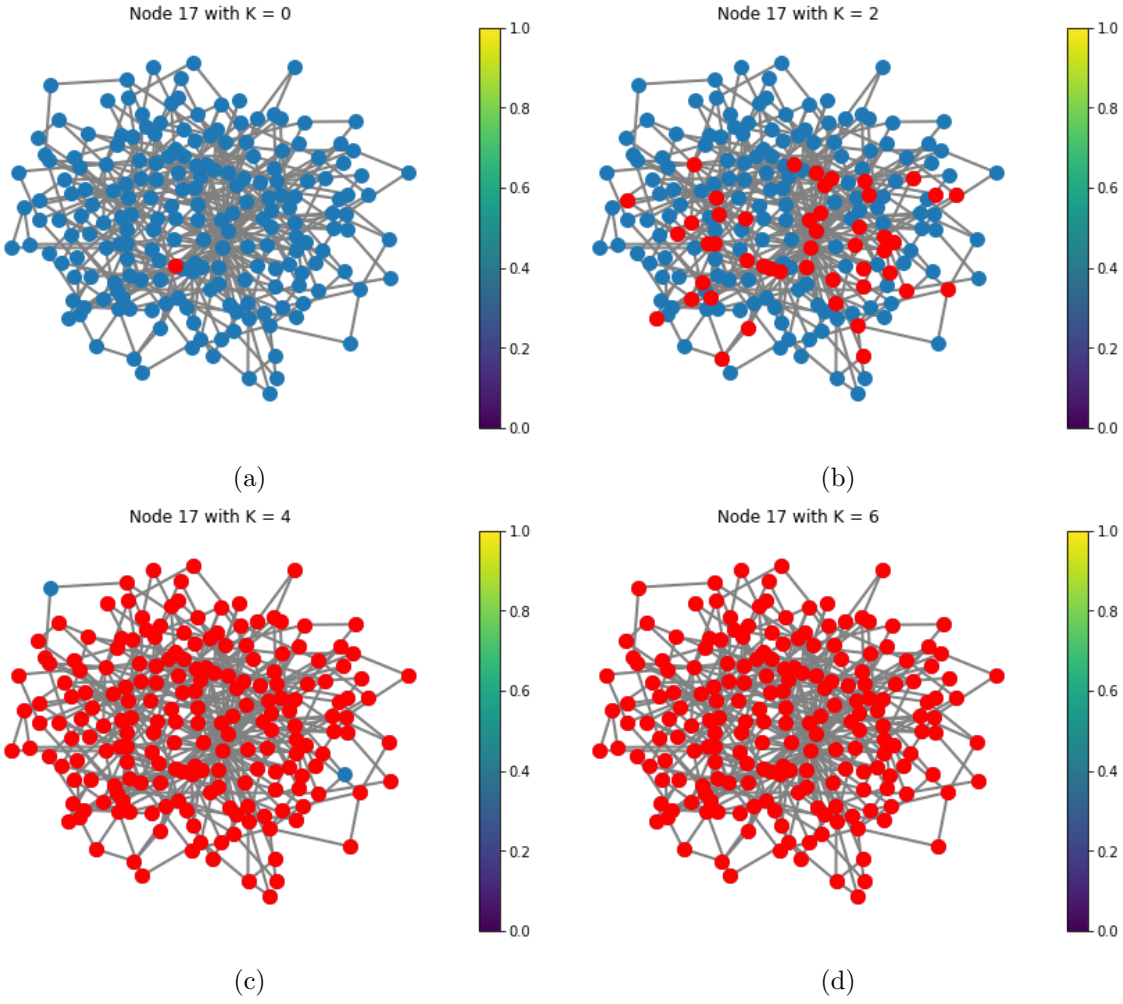
Problem 2:

Part A

I write a Python function to taking the node label ($i = 1, ..., 200$) and we transformed to categorical values these labels.

B

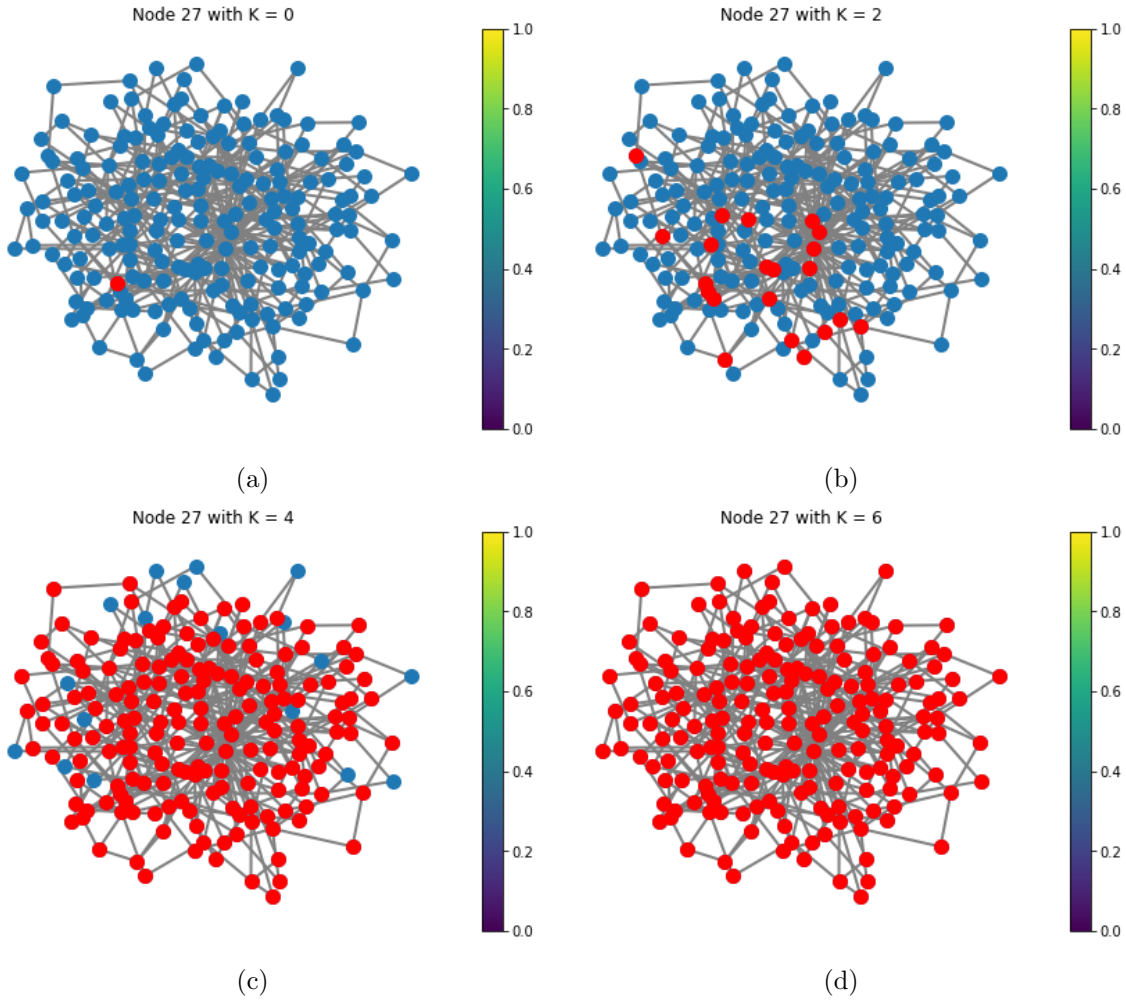
Using Networkx package it is possible to plotting the efective range of nodes $v_{i=17}$ and $v_{i=27}$



(a) (b)

(c) (d)

Figure 3: Effective range of node 17: (a) K = 0, (b) K = 2, (c) K = 4, (d) K = 6



(a) (b)

(c) (d)

Figure 4: Effective range of node 27: (a) K = 0, (b) K = 2, (c) K = 4, (d) K = 6

Part C

Using the same propagation rule from problem 1 and using the function jacobian implemented in Pytorch package it is possible to obtain the Jacobian matrix using model and the initial values (h^0)

Part D

Using the influence score we can plotted using Networks functions as following

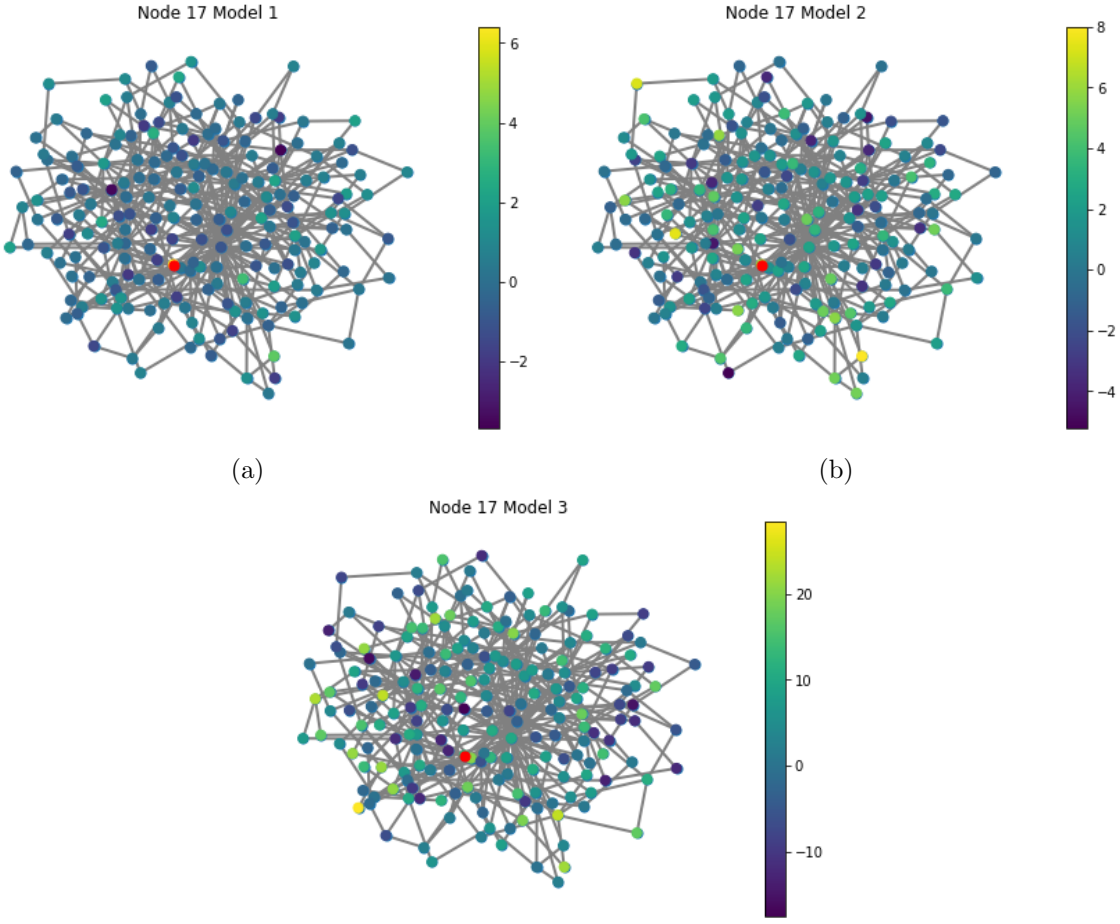


Figure 5: Influence score of node 17: (a) Model 1, (b) Model 2, (c) Model 3

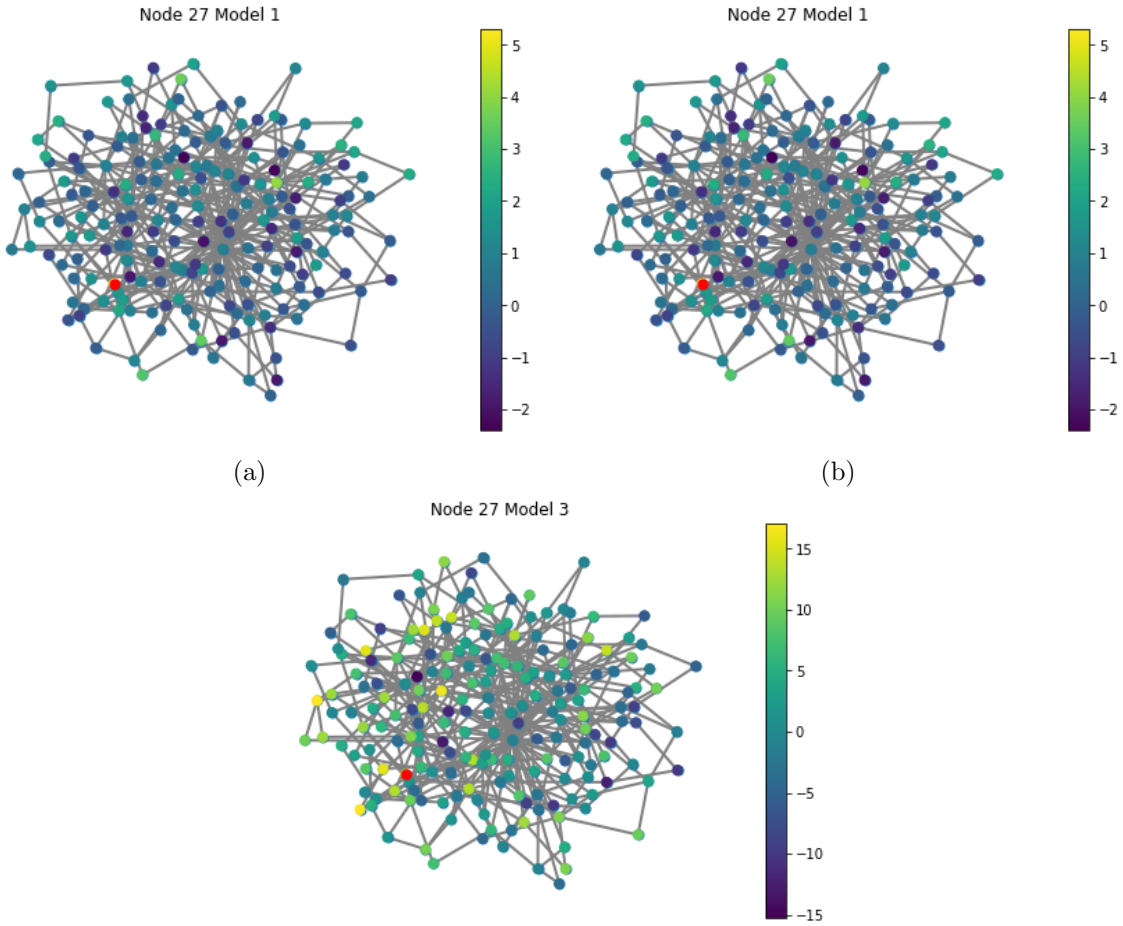


Figure 6: Influence range of node 27: (a) Model 1, (b) Model 2, (c) Model 3

Part E

We can appreciate that as we add more layers we have influence of more nodes on one, which can lead to an overestimation of parameters because the effect of the neighbors is diluted by the influence of the neighbors of the neighbors.