Rumor Source Detection with Graph Convolutional Networks

Longjie Zheng
517030910360

Yaxin Liu
517030910307

June 20, 2020

1 Background

Detecting rumors in social networks is one of the key issues for defeating rumors automatically. There are some methods which detect multiple rumor sources in an assumed setting in which the underlie propagation model is known, However, in most real settings, we are not accessible to the propagation model. **LPSI**(label based propagation source identification) is a leading algorithm which tries to avoid the underlie propagation model by using label propagation. Nonetheless, it still suffer from drawbacks that the node label is simply an integer which may restrict the prediction precision.

We have tried to solve source detection problem using graph convolutional networks and the result is promising, we could achieve higher f1 score in experiment conducted on several network datasets compared with our baseline(LPSI).

2 Algorithm

We introduce LPSI and GCN-based method in this section. In general GCN-based method utilizes result obtained by traditional LPSI and performs better.

2.1 LPSI

To our knowledge, only a few studies have been carried out for source identification without the requirement of knowing the underlying information propagation model. LPSI is the first attempt on this problem, which propagates integer labels in network and predicts the rumor sources based on the convergent node labels.

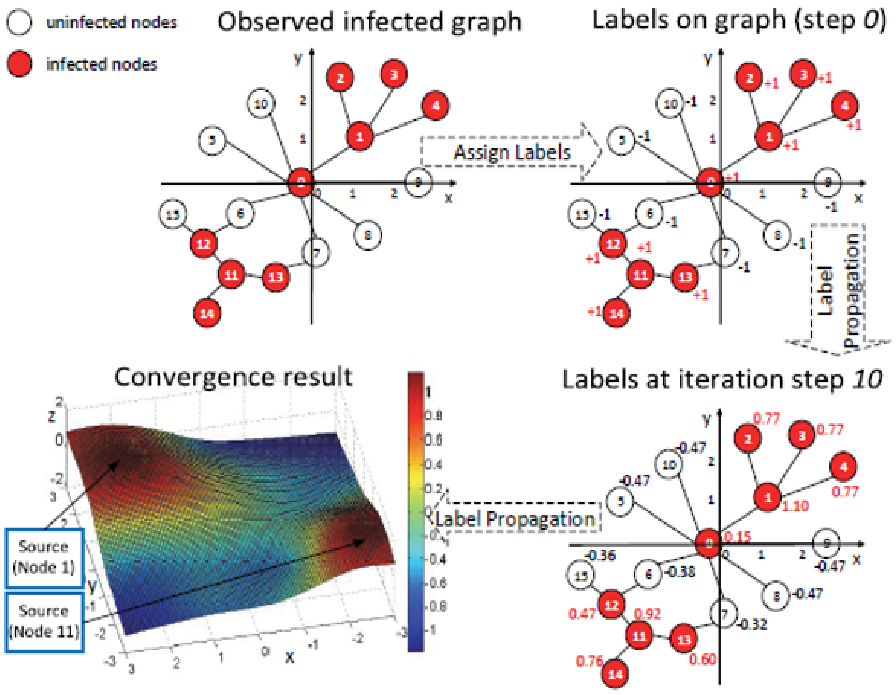


Figure 1: an illustration of LPSI

```
Algorithm 1 Label Propagation based Source Identification
Input: The infected network G = (V,E), parameter $\alpha$, the initial infection state vector Y.
Form the weight matrix W defined by $W_{ij} = 1$ if there exists an edge connecting nodes i and j
Construct the matrix $S = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, where D is a diagonal matrix with its (i,i) element equal to the sum of the i-th row of W.
$\Gamma^{t=0} \leftarrow Y$
while $\Gamma^t$ does not converge do
    for each node i do
        $\Gamma_i^{t+1} = \alpha \sum_{j:j \in N(i)} S_{ij} \Gamma_j^t + (1-\alpha) Y_i$
    end for
    t = t + 1
end while
ANS = {}
for each node i do
    if $\Gamma_i >$ all i's neigbours' $\Gamma$ value then
        $ANS = ANS \cup \{i\}$
    end if
end for
return ANS
```

As an example, **Fig. 1** depicts a partially infected network, in which a sub-network has been infected by a stochastic process starting from two sources (nodes 1 and 11). The red nodes are infected nodes and the white ones are uninfected. At first, we assign positive labels (+1) to infected ones, and negative labels (-1) to uninfected ones. After that, their label values are propagated and updated iteratively, and the propagation result at iteration step 10 is shown at the bottom right of Fig. 1. Finally, we get the convergence result, where nodes 1 and 11 are two local maximum points. As a result, we consider these two nodes as infection sources. the iteration formulation and convergent state are defined as follows

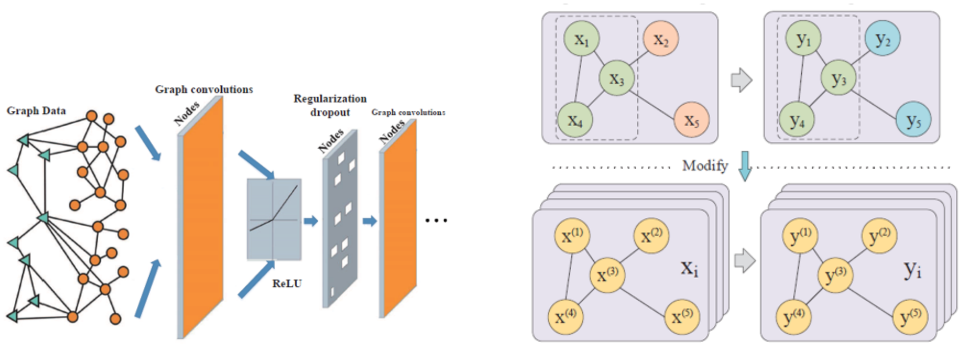
$$\Gamma_i^{t+1} = \alpha \sum_{j:j \in N(i)} S_{ij}\Gamma_j^t + (1-\alpha)Y_i \tag{1}$$

$$\Gamma^* = (1-\alpha)(I-\alpha S)^{-1}Y \tag{2}$$

$\alpha \in (0, 1)$ is the parameter used to control the influence that node i gets from its neighbors. N(i) represents the set of neighbors of node i. Γ_i^t is the infection state of node i at iteration t . S_{ij} is the (i, j)-th element of the regularized Laplace matrix S of G. Y_i is the given infection state of node i.

2.2 Graph Convolutional Network

Graph Convolutional Network is an extension of CNN on graph data, which generates local permutation-invariant aggregation on the neighborhood of a node in a graph such that the features of a graph can be efficiently captured.



(a) GCN (b) Modified GCN

Figure 2: Structure of traditional GCN and modified GCN layer

As GCN is originally designed for semi-supervised learning, although it is able to effectively capture and properly express the features of a graph, it cannot be directly applied for the supervised learning based source detection problem. We modify GCN and apply it in our graph feature capturing task. The details can be found in Fig 2.

In the modified GCN, each node receives a component of xi. When all the training data is received by GCN and it reaches convergence, the model can be used to predict rumor sources for test samples.

2.2.1 Input Generation

The existing work such as LPSI treats the original infection state Y as labels of nodes. However, in our point of view, the elements of Y are simply integers, which is not informative enough to express the complicated connection structure between nodes. Therefore, we propose an input generation algorithm to expand the integer label into a multi-dimensional vector for each node. And details could be found at algorithm 2.

Algorithm 2 Input Generation

```
Input: The infected network G = (V,E,Y), parameter $\alpha$, $Y = \{Y_1, \ldots, Y_{|v|}$.
Form the weight matrix W defined by $W_{ij} = 1$ if there exists an edge connecting nodes i and j
Construct the matrix $S = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, where D is a diagonal matrix with its (i,i) element equal to the sum of the i-th row of W.
$V_3 = Y, V_4 = Y$
for $i < len(Y)$ do
    if $Y_i = -1$ then
        $V_{3i} = 0$
    else
        $V_{4i} = 0$
    end if
end for
$d_1 = Y$
$d_2 = (1 - \alpha)(I - \alpha S)^{-1} Y$
$d_3 = (1 - \alpha)(I - \alpha S)^{-1} V_3$
$d_4 = (1 - \alpha)(I - \alpha S)^{-1} V_4$
return concatenate($d_1, d_2, d_3, d_4$)
```

To be more specific, d1 is the infection state value which is equal to Y where the infected nodes are set to 1s and the uninfected ones are set to -1s. d2 is the output of LPSI with Y as input, where the value of a node is updated based on its neighbors' values. d3 and d4 are the outputs of LPSI with modified inputs. In fact, d3 and d4 are generated respectively by changing all -1 to 0 in Y and changing all 1 to 0 in Y. In LPSI, the propagation of infected nodes is counteracted by that of uninfected ones. So by changing all -1 to 0, d3 is contains more information about the infected nodes. And by changing all 1 to 0, d4 keeps more information about the uninfected nodes.

2.2.2 Architecture

The architecture of our model GCN is shown in Fig. 3. First, the Laplacian matrix is generated from G. Next, a set of training samples are generated by Algorithm 1. Then, each training sample x flows via multiple GCN layers and a single dense layer. The active functions are ReLU for GCN layer and Sigmoid for dense layer, respectively. Finally, the prediction output of GCN is y.

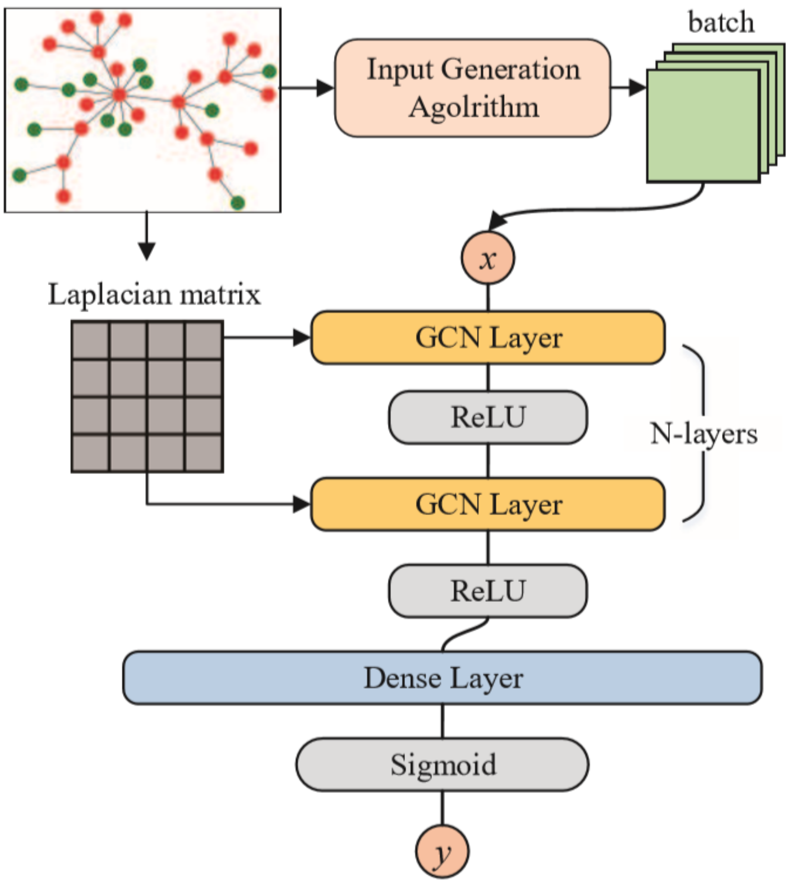


Figure 3: Architecture of GCN

3 Experiments

3.1 Datasets

In our experiments, we perform LPSI and GCN methods on three datasets:

1. **Karate** is a social network of friendships between 34 members of a karate club at an US university in the 1970s. It's a classic dataset that is widely used in many work for fast verification.

2. **Dolphin** is an undirected social network of frequent associations between 62 dolphins in a community living off Doubtful Sound, New Zealand.

3. **Jazz** is a network of Jazz bands performing from 1912 to 1940.

3.2 Generation of samples

We generate partly infected networks based on specific network structures produced by datasets using two Mathematical models of epidemic diseases SI and SIR, which are also widely used in simulating the spreading of rumors.

SI model divides the crowd into two categories: $Susceptible$ and $Infectious$ and spreads rumors with a given infectious rate β. While SIR model also take into consideration that there's also a possibility of the Infectious' recovering and thus adds a new category $Recovered$.

The generation of training samples can be described in the following steps:

1. Load network structure based on a given dataset.

2. Randomly choose 2 or 3 or 5 nodes as rumor sources.

3. Generate partly infected network G with SI or SIR model.

4. Generate input for GCN with Input Generation Algorithm.

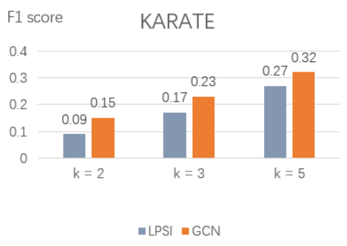
The generation of testing samples is similar to that of training samples, except that the number of rumor sources in step 2 is decided manually.

In our experiments, we sample 10000 infection sates independently for training GCN and 2000 for testing LPSI or GCN.

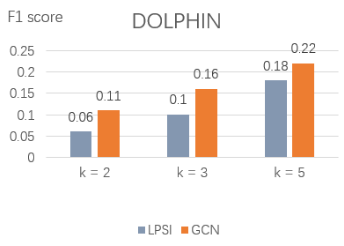
3.3 Result

Figure 4 shows that:

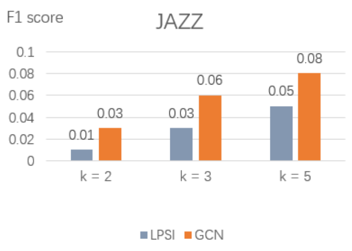
1. GCN based methods outperforms LPSI by 20% to 80% on all of the three datasets, and especially when K (ground truth rumor sources) is smaller.
2. When K is larger, both of the two methods performs better.
3. Though achieving great improvements when there are more nodes in the dataset (198 in Jazz), F1-score of predictions made by GCN becomes lower.



(a) Karate



(b) Dolphin



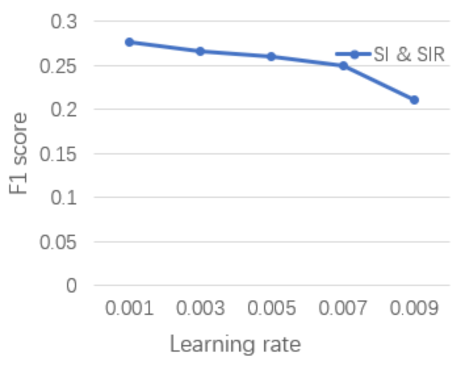
(c) Jazz

Figure 4: Performance of GCN on different datasets

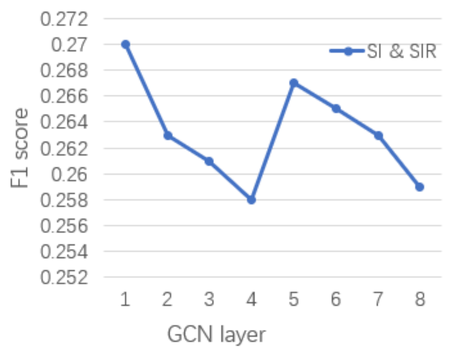
3.4 Hyper-parameter Exploration

Lastly, the impact of some important parameters in GCNSI are examined and the validation results are shown in Fig. 5. Four parameters are chosen to be validated, i.e., number of learning rate, GCN layers, hidden unit size, dropout rate. Since the validation of all datasets is massive, we only demonstrate the validation on Karate dataset as a case study. For each parameter, both of SI and SIR models are chosen as the underlying propagation models. Actually, after the cross-validation, we finally obtain the best settings of these four parameters. Therefore, when we adjust a parameter, the other three are fixed with the best settings, and the impact of each parameter is introduced separately in the rest of this section.

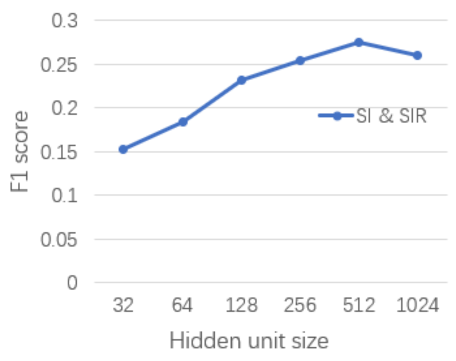
As we could conclude from Fig 5, a relatively smaller learning rate yields a better result, since a smaller learning rate makes a more stable optimizing process. And in general larger hidden size and smaller dropout probability lead to better performance. While things become more tricky when we try to adjust number of GCN layers, maybe for a small social network like karate, more GCN layers make it more susceptible to overfitting problems.



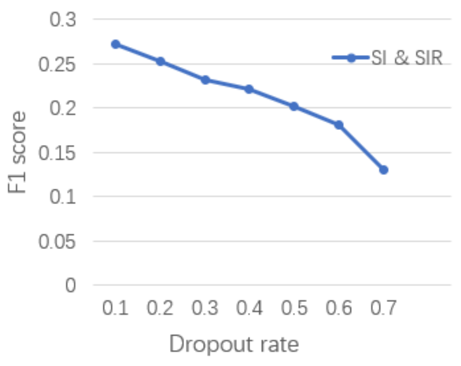
(a) learning rate



(b) number of layers



(c) hidden unit size



(d) dropout

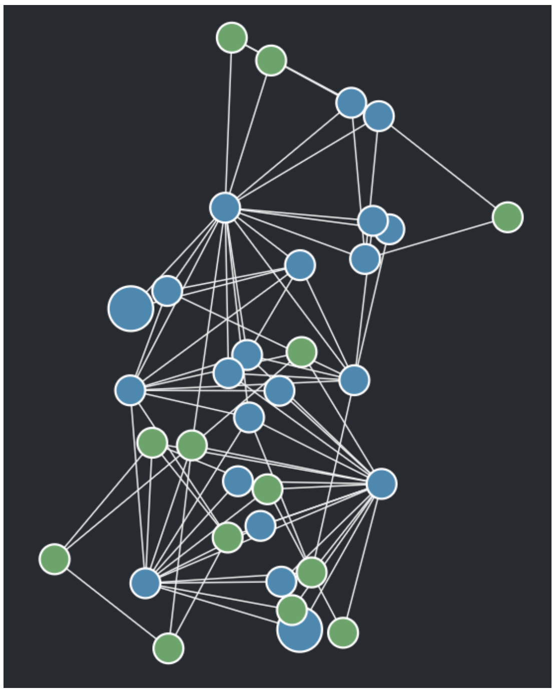
Figure 5: impact of hyper-parameters

4 Visualization

We also design a website to visualize and help understand the prediction procedure of GCN based on HTML and Flask.

The network demonstrated on the website is generated based on Karate (34 nodes) where at least 60% of the nodes are infected. There are different kind of nodes in Figure 5:

1. **Blue** nodes represent the **infected** ones.
2. **Green** nodes represent the **uninfected** ones.
3. **Larger** nodes represent the **ground truth** rumor sources.
4. **Red** nodes represent the rumor sources **predicted by GCN**.


(a) Before prediction

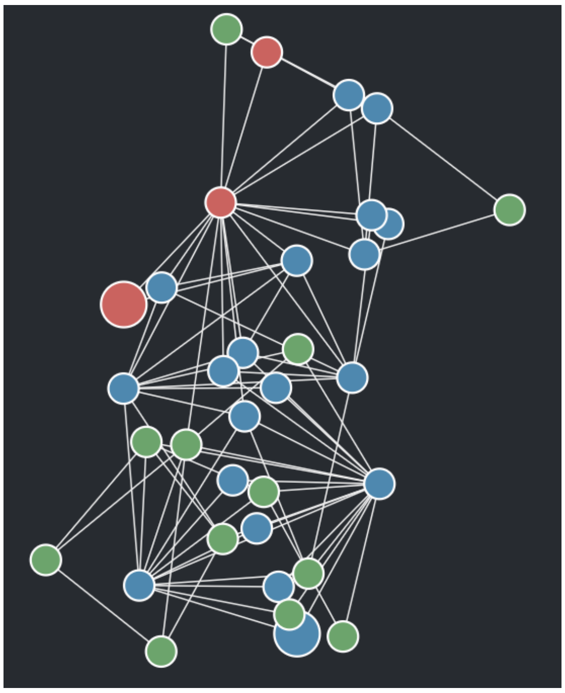

(b) After prediction

Figure 6: Visualization of the network

5 Conclusion

As source detection is an important task in defeating rumors in social network, in this project, we study the multiple rumor source detection (MRSD) problem. To solve MRSD with high precision, we propose a deep learning based model GCN to locate multiple rumor sources without prior knowledge of underlying propagation model. Furthermore, an input generation algorithm is developed to extend the integer label into vector for node representation. Then, we conduct experiment by comparing our model with the stateof-the-art baselines on three real-world networks, and the results demonstrate the effectiveness of GCN where it outperforms the state-of-the-art method. In the future, we plan to apply GCN into other rumor defeating problems such as content-based rumor identification etc. Due to the complex features of rumor content (including text features and users' behaviors), we believe it is of great potential for applying GCN.

6 Division of Work

6.1 Report

1. **Background** and **Algorithm**: Zheng Longjie 517030910360.
2. **Experiments** and **Visualization**: Liu Yaxin 517030910307.

6.2 Project

1. Together: Realization of LPSI part and construction of the websites.
2. Zheng Longjie 517030910360: Realization and modification of the GCN's structure in PyTorch.
3. Liu Yaxin 517030910307: Generation of the training and testing samples and the training process of GCN on different datasets.