# Final Report For Social Network Mining: Social Network Node Analysis and Community Mining

1st Keyu Mao
*School of Data Science*
*Fudan University*
220 Handan Road, Shanghai 200433
20307130241@fudan.edu.cn

2nd Zehao Zhang
*School of Data Science*
*Fudan University*
220 Handan Road, Shanghai 200433
20307130201@fudan.edu.cn

3rd Shuyan Ge
*School of Data Science*
*Fudan University*
220 Handan Road, Shanghai 200433
20307130091@fudan.edu.cn

***Abstract*—We review the concepts of several classic graph tasks in this project. First of all, we use Louvain, Infomap and label propagation algorithm for the community detection task. Then, we carry out network analysis on graphs, which is mainly based on network attributes and centrality computation. Moreover, we compare methods on link prediction with and without neural networks. Finally, we apply some classic machine learning algorithms and three GNN models for node classification task. As for improvement, we also try GCNII, which outperforms above typical models.**

***Index Terms*—Network analysis, Community detection, Node classification, Link prediction, GNN**

## I. Introduction

### A. Background

Complex networks can represent many systems with network structure characteristics in the real world, such as social relations and paper citation relations. It often has a remarkable community structure. A social network is a social structure composed of many nodes. Nodes usually refer to individuals or organizations, while social networks represent various social relationships. Before the birth of the Internet, social network analysis was an important research branch of sociology and anthropology. Early social networks mainly refer to professional networks established through cooperative relationships, such as scientific research cooperation networks, actor cooperation networks, etc.

Community mining is a hot topic in the field of complex network analysis. The existing community mining algorithms can be classified into three categories: optimization-based algorithms, heuristic algorithms and other algorithms (1).

In the process of using datasets to process social networks, many tasks are closely related to real social networks. For example, by studying Facebook-related social data sets, we can understand the structure of Facebook social circles to a certain extent. In addition, we can also study network evolution to simulate the characteristics of network changes over time in real society. It can also use the existing data to perform link prediction and node classification tasks, so as to be applied to real-world tasks. In brief, through the analysis of social networks attributes and understanding the various properties of social networks, we can have a better comprehension of the relationship in real society.

In this project, we will carry out several social network node analysis and community mining tasks, including community detection, network attribute analysis, link prediction and node classification. This paper is organized as following parts:

- Dataset introduction (ego-Facebook and Cora)
- Community detection based on several algorithms
- Network analysis based on attribute analysis and centrality computation
- Link prediction
- Node classification and improvement

### B. Datasets overview

In the first two parts of the project, we will mainly focus on 2 datasets: ego-Facebook and Cora datasets.

Ego-Facebook dataset consists of circles, or friends lists, from Facebook. Facebook data are collected from survey participants using this Facebook app. The dataset includes node features, circles, and ego networks.

Facebook data has been anonymized by replacing the Facebook-internal ids for each user with a new value. Also, while feature vectors from this dataset have been provided, the interpretation of those features has been obscured.

Meanwhile, the Cora dataset consists of 2708 scientific publications classified into one of seven classes. The citation network consists of 5429 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. Cora dataset is one of the most popular graph datasets and is typically regarded as the benchmark dataset for several topics concerning graph algorithms.

To take an overview of the two datasets mentioned above, we visualize them in Fig. 1 and Fig. 2.

## II. Community Detection

There may not exist a best algorithm for community detection, because datasets in reality often have different definitions for community. Louvain, Infomap and label propagation are three commonly applied algorithms.

### A. Louvain Algorithm

Louvain algorithm performs hierarchical clustering on the graph by maximizing the modularity of the graph. Denote $A \in$

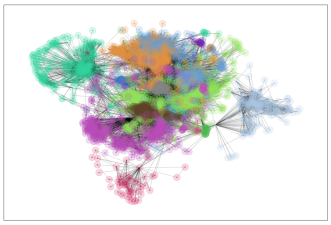

Fig. 1. Ego-Facebook Dataset

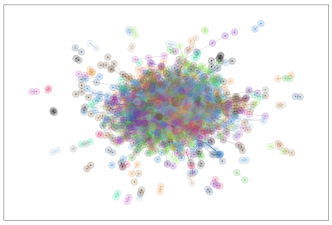

Fig. 2. Cora Dataset

$\mathbb{F}_2^{n\times n}$ to be the adjacency matrix of the graph, $m = \frac{1}{2}\sum_{ij} A_{ij}$ to be the total number of edges and $d_i$ to be the degree of each node, the community $c_i$ of each vertex $i$ is assigned by solving the following maximization problem.

$$c = \text{argmax}_c Q = \text{argmax}_c \frac{1}{2m}\sum_{i,j}\left(A_{ij} - \frac{d_i d_j}{m}\right)\mathbb{I}_{c_i=c_j}. \tag{1}$$

The objective $Q$ of the maximization problem is known as the modularity. The constant $\frac{1}{2m}$ is free to be canceled out in the optimization. Unfortunately the explicit solution is computationally hard as pointed out in (2) and the algorithm takes an iterative pattern. At the start of iteration, each node belongs to an isolated community of size 1. For each node $i$ we compute the increment in the modularity by removing it from the current community $c_i$ and appending to some other community $c_i'$. Note that the gain in $Q$ after moving node $i$ from isolated to a community $c$ can be conveniently computed via formula:

$$\Delta Q(i,c) = \frac{d_{i,c}}{2m} - \frac{d_i D_c}{2m^2}. \tag{2}$$

where $d_{i,c}$ is the number of edges from $i$ with the other end lying in the community $c$, and $D_c$ is the number of edges with both vertices lying in community $c$. The aforementioned increment in $Q$ by transplanting $i$ from community $c_i$ to $c_i'$ can be computed by

$$\Delta Q\left(i, c_i, c_i'\right) = -\Delta Q\left(i, c_i \setminus \{i\}\right) + \Delta Q\left(i, c_i'\right). \tag{3}$$

The process of moving a node from a community to another is repeated until no local increment is available. In this stage we merge each community to a new node to form a new graph and continue performing the Louvain algorithm.

### B. Infomap

The idea of Infomap is to describe the path generated by random walks with the shortest encoding when doing random walks with unlimited step walks on a graph. And its intuition is minimizing the code of the random walk sequence with community detection. A good community partition shall lead to short encoding (3).

*1) Map equation and Two-layer coding:* For implementation, Infomap uses a two-layer coding for the nodes. The first layer is to encode the groups(communities), the second layer is to encode the nodes in one group. Notice that two nodes in different groups may share the same node code.

A good group division can lead to a shorter set of codes in the above structure. According to Shannon's information entropy, there exists a lower bound on code length with respect to a module partition M. The map equation can measure average number of bits per step describing random walk:

$$L(\mathrm{M}) = q_{\curvearrowright} H(Q) + \sum_{i=1}^{m} p_{\circlearrowright}^{i} H\left(P^i\right) \tag{4}$$

One explanation of entropy is the average byte length required to encode each state. Here $H(Q)$ is The average byte length required to encode group names and $H\left(P^i\right)$ is the average byte length required for encoding all nodes in the group i. With $q_{i\curvearrowright}$ indicates the probability to exit module $i$, $q_{\curvearrowright} = \sum_{i=1}^{m} q_{i\curvearrowright}$ is the proportion of all encodings that represent the group name in the encoding. And with $p_\alpha$ for the probability to visit node $\alpha$, $p_{\circlearrowright}^{i} = \sum_{\alpha\in i} p_\alpha + q_{i\curvearrowright}$ is the proportion of the encoding for all nodes that belong to group $i$, including jumping-out nodes. Therefore,

$$H(Q) = -\sum_{i=1}^{m} \frac{q_{i\curvearrowright}}{\sum_{j=1}^{m} q_{j\curvearrowright}} \log\left(\frac{q_{i\curvearrowright}}{\sum_{j=1}^{m} q_{j\curvearrowright}}\right) \tag{5}$$

and

$$\begin{aligned} H\left(P^i\right) = &-\frac{q_{i\curvearrowright}}{q_{i\curvearrowright} + \sum_{\beta\in i} p_\beta} \log\left(\frac{q_{i\curvearrowright}}{q_{i\curvearrowright} + \sum_{\beta\in i} p_\beta}\right) \\ &- \sum_{\alpha\in i} \frac{p_\alpha}{q_{i\curvearrowright} + \sum_{\beta\in i} p_\beta} \log\left(\frac{p_\alpha}{q_{i\curvearrowright} + \sum_{\beta\in i} p_\beta}\right). \end{aligned} \tag{6}$$

*2) Algorithm:* The Infomap algorithm uses the same approach as in PageRank to obtain the access probability of each node. After the probability generation, Infomap detects the communities with the following steps:

1. Assign each node to its own module.
2. In random sequential order, each node is moved to the neighboring module that results in the largest decrease of the map equation.
3. Repeat second step until there is no longer any move generating a decrease of the map equation.

### C. Label propagation

*1) LPA:* The Label Propagation algorithm (LPA) is a fast algorithm for community detection task in a graph. It detects these communities using network structure. A pre-defined objective function or prior information about the communities is not required. However, running this algorithm requires large memory availability. Nodes that have the same community label at convergence can be seen as belonging to the same community (4).

*2) Algorithm:* The algorithm works as follows:

1. Every node is initialized with a unique community label (an identifier).
2. These labels propagate through the network.
3. At every iteration of propagation, each node updates its label to the one that the maximum numbers of its neighbors belongs to.

4. LPA reaches convergence when each node has the majority label of its neighbors.

5. LPA stops if either convergence, or the max-iterations is achieved.

### D. Comparison

We apply the three algorithms on ego-facebook dataset and the visualization results are displayed as follows.

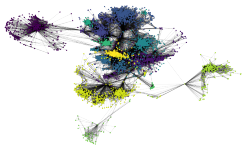

Fig. 3. Louvain

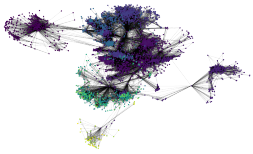

Fig. 4. Infomap

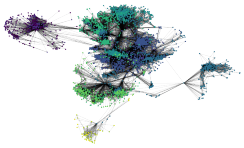

Fig. 5. LPA

TABLE I
COMMUNITY DETECTION ALGORITHMS ON FACEBOOK DATASET

| Method | Clusters | Modularity |
|---|---|---|
| Louvain | 15 | **0.83070** |
| Infomap | 74 | 0.81200 |
| LPA | 44 | 0.73684 |

## III. NETWORK ANALYSIS

### A. Zipf's Law

Zipf's law is an empirical law formulated using mathematical statistics that refers to the fact that for many types of data studied in the physical and social sciences, the rank-frequency distribution is an inverse relation. In the class, we learned that Zipf's law was originally formulated in terms of quantitative linguistics, stating that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table (5).

In most social network, the network data is proved that the degree of each node will follow a certain pattern of power law, which can be concluded as:

$$p_k = \alpha \cdot k^{-b}. \tag{7}$$

Here $k$ denotes the degree value and $p$ is frequency.

In this section, we test the relation between degree and counting times for nodes in ego-Facebook network and Cora dataset. We plot the figure of degree-counts and its log version.

Clearly, Fig. 6, 7, 8 and 9 show the property that ego-Facebook dataset and Cora dataset do follow the Zipf's law. The color in the figures just describe the frequency of nodes with certain degrees, where brighter color denotes higher frequency.

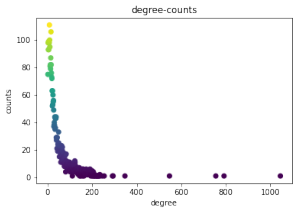


Fig. 6. Ego-Facebook

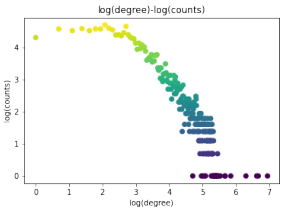


Fig. 7. Ego-Facebook

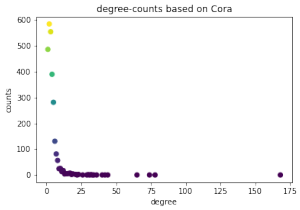


Fig. 8. Cora

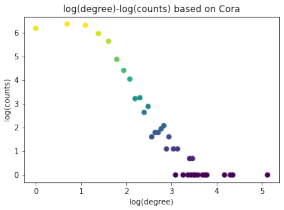


Fig. 9. Cora

### B. Network Attributes

Before thoroughly examining the centrality of social network, we may take a look at the attributes of the network. In this section, although main focus will be laid on the ego-Facebook dataset and Cora dataset, we will also introduce extra models for comparison.

To study the graphs, we can model the generation of them. If we can guarantee the similarity between our generated graphs and the real world networks, then:

- We can analyze cost-efficient simulated graphs instead of real-world networks.
- We can better understand real-world networks by providing concrete mathematical explanations.
- We can perform controlled experiments on synthetic networks when real-world networks are unavailable.

Hence, by generating networks which are similar to ego-Facebook, we can derive a series of network attributes to analyze. We have tested 3 models, random Erdos-Renyi graph, small world, or known as Watts-Strogatz graph, and Barabasi-Albert graph to generate networks with identical number of nodes and similar number of edges. Then we measure a series of attributes like clustering coefficient and density that are displayed below.

TABLE II
ATTRIBUTES OF DIFFERENT MODELS OR DATASET

| Models | Nodes | Edges | Cluster | Distance | Diam | Density |
|---|---|---|---|---|---|---|
| Facebook | 4039 | 88234 | 0.61 | 3.69 | 8 | 0.11 |
| Erdos-R | 4039 | 88186 | 0.01 | 2.61 | 4 | 0.11 |
| Watts-S | 4039 | 84819 | 0.71 | 4.18 | 6 | 0.10 |
| Barabasi-A | 4039 | 84378 | 0.04 | 2.54 | 4 | 0.10 |
| Cora | 2708 | 5278 | 0.24 | 3.90 | 19 | 0.001 |

[1] Cluster refers to clustering coefficient of different models or datasets.
[2] Diam refers to the diameter of the graph.

As we can see in the table II, all the generated models take after ego-Facebook. Erdos-Renyi can merely fit the average distance of real-world model while Watts-Strogatz model performs well in clustering coefficient and diameter. Barabasi-Albert model on the other hand behaves terribly in clustering coefficient but it fits average distance and we can prove that it follows a certain power distribution like real-world model.

As for datasets, Cora dataset has much fewer nodes and edges than ego-Facebook. Besides, Cora dataset has extremely high diameter and average distance. Notice that Cora dataset is prepared for all graph tasks and is not a typical social network.

### C. Network Centrality

In this section, we will try to measure the network centrality based on 2 datasets mentioned in the last section. We try out 6 kinds of centrality measurements: degree centrality, eigenvector centrality, Katz centrality, pagerank centrality, betweenness centrality and closeness centrality. The following table III illustrates the principle of each type of centrality.

TABLE III
CENTRALITY COMPUTING EQUATIONS

| Centrality Measurement | |
|---|---|
| Degree | $C_v = \text{degree}(v)$ |
| Eigenvector | $C_e(v_i) = \lambda^{-1} \sum_{j=1}^{n} A_{ji} C_e(v_j)$ |
| Katz | $C_k(v_i) = \alpha \sum_{j=1}^{n} A_{j,i} C_k(v_j) + \beta$ |
| Pagerank | $C_p(v_i) = \alpha \sum_{j=1}^{n} A_{ji} \frac{C_p(v_j)}{\text{degree}_{\text{out}}} + \beta$ |
| Betweenness | $C_b(i) = \sum_{s \neq t \neq i} \frac{\sigma_{st}(i)}{\sigma_{st}}$ |
| closeness | $C_c(i) = (\frac{1}{n-1} \sum_{i \neq j} l_{ij})^{-1}$ |

It is impractical to display all the results based on all kinds of measurements above. As a result, we decide to set threshold, highlight part of nodes, and visualize the results. We use red, orange, yellow and silver to denote the nodes with different centrality. For instance, red nodes are larger than 75% of the max value of centrality. Likewise, orange: $50\% \sim 75\%$, yellow: $25\% \sim 50\%$, and silver: $0\% \sim 25\%$. The results are shown in the following subsections.

*1) Centrality Reults for Ego-Facebook:* In this subsection, we visualize the centrality results of ego-Facebook dataset. The results are displayed in figure 10 to figure 15.

Obviously, under different kinds of measurements, the centrality of nodes perform with significant difference.

Among all the measurements, closeness centrality turns out to be a unique one since under this measurement, more nodes will tend to possess high centrality. The core idea of closeness centrality is that the closer the node is to the center of the graph, the easier it is to reach other nodes quickly, so the most central node should have the smallest average shortest path to other nodes. In that case, when computing closeness centrality, every node in the network needs to consider all other nodes in the network. Hence, after adding up the distance, the difference among nodes will not be prominent.

Besides, under eigenvector centrality, all nodes with high eigenvector centrality value gather in small regions, which probably means that eigenvector centrality cannot demonstrate the essence of the topological structure given certain networks.

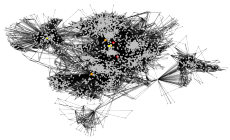

Fig. 10. Degree

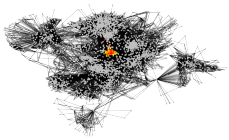

Fig. 11. Eigenvector

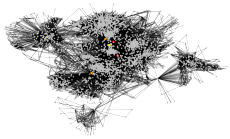

Fig. 12. Katz

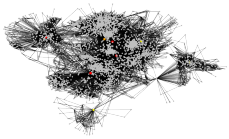

Fig. 13. Pagerank

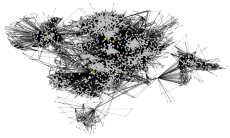

Fig. 14. Betweenness

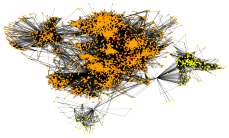

Fig. 15. Closeness

*2) Centrality Reults for Cora dataset:* In this subsection, we visualize the centrality results of Cora dataset. The results are shown in figure 16 to figure 21.

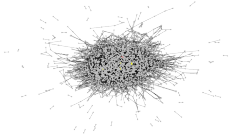

Fig. 16. Degree

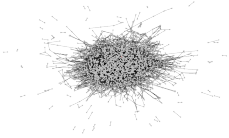

Fig. 17. Eigenvector

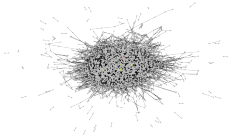

Fig. 18. Katz

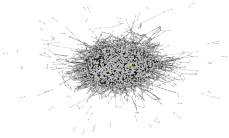

Fig. 19. Pagerank

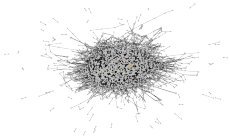

Fig. 20. Betweenness

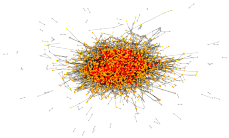

Fig. 21. Closeness

The results of Cora dataset also help to prove the conclusions we have proposed in the previous section.

### D. Network Evolution

Almost all real world networks evolve over time, either by adding or removing nodes or links over time. As a consequence, it is meaningful for us to simulate the process of evolution by applying reasonable network models since we

may get some useful information which can be applied in real world problems.

We decide to use Barabasi-Albert models in this section and derive the process of evolution. To be specific, we start from a graph with 500 nodes and 0 edge, and every time we add 50 edges to the graph. The probability that a node gets selected to add an edge from depends on its current degree. Note that in experiment, we only display nodes that already have edges, and the number of such nodes is just the size of present biggest subgraph.

The following table IV shows the the evolution of network attributes with the process of adding 50 edges. And figures below the table just visualize the evolution process. Apparently, the average distance is around 1.7 and diameter is either 2 or 3. In the 100th iterations when 5000 edges have been added, the average distance tends to achieve the minimum.

TABLE IV
NETWORK ATTRIBUTES

| Degree | Cluster | Distance | Diameter | Size | Iterations |
|---|---|---|---|---|---|
| 1.96 | 0 | 1.96 | 2 | 51 | 1 |
| 16.7 | 0.88 | 1.72 | 2 | 60 | 10 |
| 66.7 | 0.59 | 1.55 | 2 | 150 | 100 |
| 90.0 | 0.27 | 1.82 | 3 | 499 | 449 |

[1] Cluster refers to clustering coefficient of different models or datasets.
[2] Size refers to the the size of present biggest subgraph.

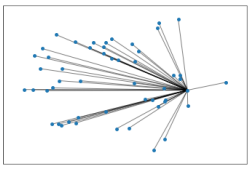

Fig. 22. Iteration = 1

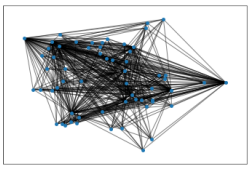

Fig. 23. Iteration = 10

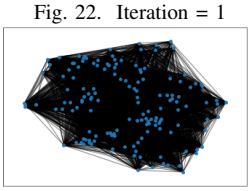

Fig. 24. Iteration = 100

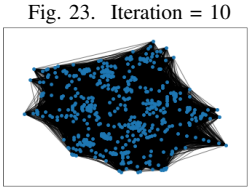

Fig. 25. Iteration = 449

## IV. LINK PREDICTION

Given a graph, the task of link prediction is to predict the probability of existence of an edge between two nodes.

### A. Node Similarity

Node similarity is a direct prediction for the probability of an edge. Let $N(x)$ be the neighborhood of vertex $x$, commonly used metrics for the similarity $s(x, y)$ include Jaccard coefficient, Adamic and Adar coefficient and preferential attachment. The equations are presented in the following.

$$\begin{cases} s_{\text{Jaccard}}(x,y) = \dfrac{|N(x) \cap N(y)|}{|N(x) \cup N(y)|} \\ s_{\text{Adamic\&Adar}}(x,y) = \displaystyle\sum_{z \in N(x) \cap N(y)} \frac{1}{\log |N(z)|} \\ s_{\text{preferential}}(x,y) = |N(x)| \cdot |N(y)| \end{cases} \tag{8}$$

The three metrics above only take the adjacency matrix as input. The preferential attachment is simplest to compute by performing a matrix product while the Adamic and Adar is also simple to compute by accumulating the contribution of each node $z$ to its neighbors. However, the Jaccard coefficient often takes much more time.

### B. GAE

When provided with features of each node, a wider range of methods of link prediction are available. Graph auto-encoders (GAE), or variational graph auto-encoders (VGAE) proposed in (6) embed each node in a latent vector space according to their features. Denote the graph adjacency matrix by $A \in \mathbb{F}_2^{n \times n}$ and the features of each node by $X \in \mathbb{R}^{n \times m}$, the graph auto-encoder is a function $f$ mapping each node to a vector of dimension $d$. In practice, the function $f$ is suggested to take the form $f(A, X) = \tilde{A}\left[\text{ReLU}(\tilde{A}XW_1)\right]W_2$ where $\tilde{A}$ is the symmetrical normalization of $A$ while $W_1$ and $W_2$ are parameter matrices to be optimized.

$$f(A, X) = Z \in \mathbb{R}^{n \times d}. \tag{9}$$

In the dimension-$d$ latent space, the similarity of two nodes $i$ and $j$ is evaluated by the inner product of their latents, $z_i^T z_j$. The probability of two nodes having an edge inbetween is given by $\sigma(z_i^T z_j)$ where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid operation. The probability of connecting each pair of vertices can be thus written in the matrix form, $\sigma(ZZ^T)$.

The goal of GAE is to construct $Z$ such that $\sigma(ZZ^T) \approx A$, which restores the graph and predicts the connectivity of two nodes in consistent with the ground truth. The training loss is then defined by the cross-entropy of the prediction of each edge as Eq. 10. Negative sampling or positive reweighting are required to overcome sample imbalance since the graph is oftentimes sparse.

$$\mathcal{L} = H(\sigma(ZZ^T), A). \tag{10}$$

### C. VGAE

Variational graph auto-encoders (VGAE) (6) embed the graph to multiple Gaussian distributions with dimension $d$ rather a fixed latent matrix compared with GAE. To be specific, with adjacency matrix $A$ and features $X$, VGAE is a function that takes in $A$ and $X$ and outputs the parameters of the Gaussian, mean $\mu \in \mathbb{R}^{n \times d}$ and variance for each entry $\sigma \in \mathbb{R}^{n \times d}$. The latent of each node $i$ is then sampled from the corresponding Gaussian, $z_i \sim N(\mu_i, \sigma_i)$ where $\mu_i, \sigma_i \in \mathbb{R}^d$.

$$f(A, X) = (\mu, \sigma) \in \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times d}. \tag{11}$$

Still the reconstruction of the graph is predicted by $\sigma(ZZ^T)$. However, the training loss need to be corrected by adding a term that penalizes the distrbution from the standard Gaussian.

$$\mathcal{L} = H(\sigma(ZZ^T), A) + \frac{\lambda}{2n}\sum_{i=1}^{n} \|\mathbf{1} + \log\sigma_i^2 - \mu_i^2 - \sigma_i^2\|_1. \quad (12)$$

### D. Comparison

Table V demonstrates the AUC of different methods in link prediction experimented on various ego-networks of the Facebook dataset. On ego-network 698 where there are very few nodes and features, the preferential attachment works best. However, in most cases, GAE outperforms all others and, along with VGAE, reveals increasing power as the graph and feature data grow larger.

TABLE V
AEGO NETWORKS OF FACEBOOK

| Index | 0 | 107 | 348 | 414 | 698 | 1684 |
|---|---|---|---|---|---|---|
| GAE | **0.8091** | **0.8743** | **0.8157** | **0.8312** | 0.7078 | **0.8362** |
| VGAE | 0.7894 | 0.855 | 0.785 | 0.7761 | 0.7284 | 0.8018 |
| Jaccard | 0.618 | 0.6409 | 0.7628 | 0.7596 | 0.6783 | 0.6364 |
| Adamic | 0.6303 | 0.6589 | 0.783 | 0.7706 | 0.7071 | 0.654 |
| pref | 0.743 | 0.7605 | 0.7829 | 0.7724 | **0.8134** | 0.7576 |
| nodes | 347 | 1045 | 227 | 159 | 66 | 792 |
| features | 224 | 576 | 161 | 105 | 48 | 319 |

## V. NODE CLASSIFICATION

Node classification aims to assign labels to nodes with unknown labels in the graph with the help of connectivity relations between nodes and a few labeled nodes.

### A. Label Propagation Algorithm

As illustrated in the community detection part, here LPA is a semi-supervised machine learning method.

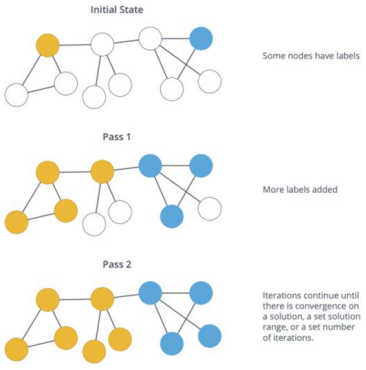


Fig. 26. Brief illustration of label propagation algorithm.

Assume we have data $x_1, ..., x_l, x_{l+1}, ..., x_n \in \mathbb{R}^p$, with $y_1, ..., y_l \in \{1, ..., C\}$. Initialize the label matrix as $Y \in \mathbb{R}^{n\times C}$. $Y_{ij} = 1$ if $y_i = j$ while otherwise zero. By generating prediction matrix $F^\star = ((1-\alpha)I + L_{sym})^{-1}Y$, the algorithm gives prediction that $\hat{Y}_i = \text{argmax}_j F_{i,j}$, where $\alpha = \frac{1}{1+\mu}$, $L_{sym}$ is normalized Laplacian matrix.

We use the 127-th anonymized feature (locale) of the ego-facebook dataset to do 0-1 classification. The visualization result is as follows, and the accuracy is 0.86.

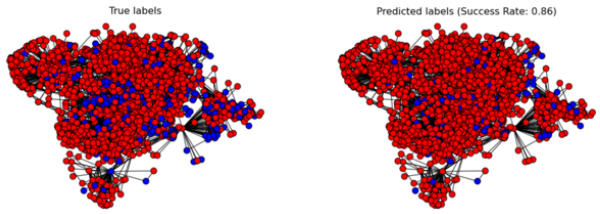


Fig. 27. LPA-visualization

### B. Node2vec

Node2vec is also an embedding algorithm like GAE. For learning node random-walk embeddings such that nearby nodes are close together in the network, our goal is to learn a mapping $f$: $u \mapsto f(u) = z_u \in \mathbb{R}^d$. The log-likelihood objective is

$$\max_f \sum_{u\in V} \log P\left(N_R(u)|z_u\right) \quad (13)$$

$N_R(u)$ is the neighborhood of node $u$ by strategy $R$. Equivalently, finding mapping $f$ minimizes $\mathcal{L}$ presented in Eq. 14.

$$\mathcal{L} = \sum_{u\in V}\sum_{v\in N_R(u)} -\log\left(\frac{\exp\left(\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_v\right)}{\sum_{n\in V}\exp\left(\mathbf{z}_u^{\mathrm{T}}\mathbf{z}_n\right)}\right) \quad (14)$$

Framing this embedding goal as a maximum likelihood optimization problem which is independent to the downstream prediction task, we use flexible and biased random walks to do the minimization to trade off between local an global structure of the network. Observe that flexible notion of network neighborhood $N_R(u)$ of $u$ leads to rich node embeddings, we develop biased random walk to generate network neighborhood $N_R(u)$ of node $u$. The process can be summarized as follows (7) (8).

1. Compute random walk probabilities.
2. Simulate $r$ random walks of length $l$ starting from each node $u$.
3. Optimize the node2vec objective using stochastic gradient descent.

The algorithm has linear-time complexity and all 3 steps are individually parallelizable. After obtaining the vector embedding of the nodes, conventional machine learning algorithms like logistic regression can be applied to predict the node labels. In our experiment, node2vec can achieve 72.6% test accuracy on the Cora dataset. The effect is not impressive, so we choose it as a baseline for subsequent comparisons.

### C. GNN

A GNN is an optimizable transformation on all attributes of the graph (nodes, edges, global-context) that preserves graph symmetries and permutation invariance. We build GNNs using

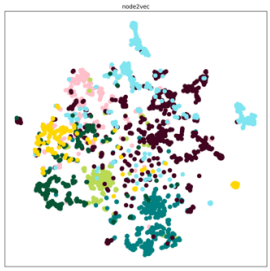

Fig. 28. Node classification of Cora predicted by node2vec.

the "message passing neural network" framework like Fig. 29. GNNs adopt a "graph-in, graph-out" architecture without changing the connectivity of the input graph (9).

GNN generate node embeddings based on local network neighborhoods and nodes aggregate information from their neighbors using neural networks. Neighborhood aggregation is the key distinction of different approaches (8).

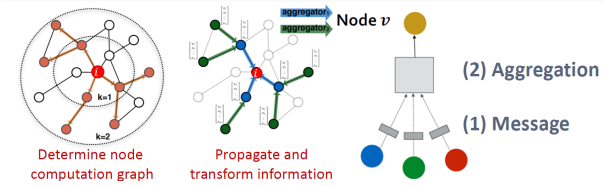


Fig. 29. Illustration of neighborhood aggregation in GNN.

*1) GCN:* Motivated by a localized first-order approximation of spectral graph convolution, GCN follows the following rules for propagation between layers

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{15}$$

where $\tilde{A} = A + I$ denotes the adjacency matrix of the undirected graph with added self-connections. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ denotes the degree matrix of the nodes. $W^{(l)}$ denotes the layer specific trainable weight matrix. $H^{(l)}$ denotes the output of the $l^{\text{th}}$ layer. $\sigma(\cdot)$ denotes an activation function like ReLU. Some researches indicate that combining local graph structure and node-level features yields good performance on node classification tasks, but the way GCN aggregates is structure-dependent, which can hurt its generalizability (10).

*2) GAT:* GAT uses weighted neighbor features with feature dependent and structure-free normalization, in the style of attention. The key difference between GAT and GCN is how the information from the one-hop neighborhood is aggregated (11).

Graph attention network (GAT) proposes a masked self-attentive layer to address the shortcomings of previous methods based on graph convolution or its approximations. The rules for propagation between layers for node $j$ are given by

$$\vec{h}'_i = \sigma\left(\sum_{j\in N_i}\alpha_{ij}\vec{h}_j\right) \tag{16}$$

where $\alpha_{ij} = \text{softmax}_j\left(e_{ij}\right) = \frac{\exp(e_{ij})}{\sum_{k\in Ni}\exp(e_{ik})}$ and $e_{ij}$ denotes LeakyReLU $\left(\vec{\mathbf{a}}'\left[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j\right]\right)$; $\vec{\mathbf{a}}$ denotes the role of a single-layer feedforward neural network. In fact, the high attention score indicates that the two nodes have the same class label.

*3) GraphSAGE:* For GraphSAGE, we use the NeighborSampler module in Pytorch-geometric to sample a fixed-size neighborhood of each node. This will allow user to operate on large graphs that do not fit into memory. As described in the research paper (12), the rules for propagation between layers are:

$$\begin{aligned} h^k_{N(v)} &\leftarrow \text{AGGREGATE}_k(\{h^{k-1}_u, \forall u \in N(v)\}) \\ h^k_v &\leftarrow \sigma(W^k * \text{CONCAT}(h^{k-1}_v, h^k_{N(v)})) \\ h^k_v &\leftarrow h^k_v / \|h^k_v\|_2, \forall v \in V \end{aligned} \tag{17}$$

## D. Comparison and Analysis

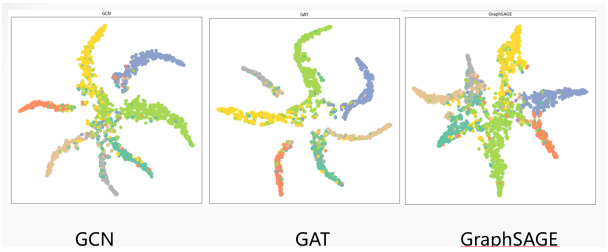


Fig. 30. Node classification on Cora predicted by various GNNs.

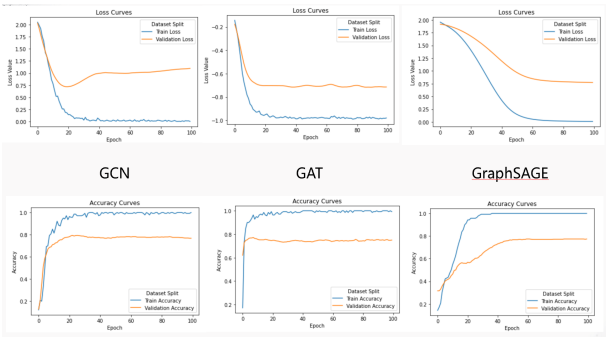


Fig. 31. Training result of various GNNs.

*a) Overfit:* From Fig. 31, we find that almost all GNNs models tested on the Cora dataset have accuracy almost 80%, all outperforming node2vec. However, their accuracy on the training data is close to 100%, which indicates that GNNs generally have overfitting problems.

*b) Over-smoothing:* We observe that stacking more layers results in a significant drop in accuracy. This is the over-smoothing problem (13) where neighboring nodes become increasingly similar when the number of layers of forward propagation accumulates, leading the model to be less discriminative.

### E. Model improvement

GCNII introduces two structures: initial residual and identity mapping, which solves the over-smoothing problem in deep networks; and the research shows the the difference between GCN and GCNII and prove its effectiveness (13). In terms of the rules for propagation between layers, updated $H^{(l+1)}$ can be expressed as

$$\sigma\left\{\left(\bar{\alpha}\tilde{P}H^{(l)}+\alpha H^{(0)}\right)\left(\bar{\beta}I+\beta W^{(l)}\right)\right\} \tag{18}$$

in the form of activated value of the matrix product between a smoothed $H^{(l)}$ and $W^{(l)}$. Here $\bar{\alpha}=1-\alpha$ and $\bar{\beta}=1-\beta$ are hyperparameters to control the proportion of initial residual $H^{(0)}$ and the identical mapping $I$.

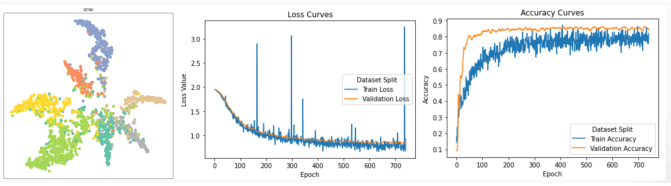


Fig. 32. GCNII performance on Cora.

TABLE VI
ACCURACY OF NODE CLASSIFICATION ON CORA

| Model | Accuracy |
|---|---|
| node2vec | 0.7260 |
| GCN | 0.7790 |
| GAT | 0.7830 |
| GraphSAGE | 0.7760 |
| GCNII | **0.8580** |

In the experiments, we implement 128-layer GCNII and achieve 85.5% accuracy in testing set and almost 90% accuracy on training set, which illustrates that both over-fitting and over-smoothing problems have been overcome, surpassing GCN on node classification task.

## VI. Conclusion And Future Work

In this paper we have taken a brief overview of several fundamental topics in the field of social network theory, including community detection, network analysis, link prediction and node classification. A few experiments are carried out on ego-Facebook and Cora dataset to review and compare the classic methods.

However, there are many more advanced models beyond the vision of this paper, for instance, GGNN (14). Also, there are greater and more complicated networks that pose challenges on computation. More work is needed to dive deeper into the investigation of social network.

## References


[1] Y. bo, LIU, *et al.*, "Complex network clustering algorithms," *Journal of software*, vol. 20, no. 1, pp. 54–66, 2009.

[2] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of community hierarchies in large networks," *J Stat Mech*, vol. abs/0803.0476, 2008.

[3] D. A. M. Rosvall and C. Bergstrom, "The map equation," 2010.

[4] G. Cordasco and L. Gargano, "Community detection via semi-synchronous label propagation algorithms," in *2010 IEEE International Workshop on: Business Applications of Social Network Analysis (BASNA)*, pp. 1–8, 2010.

[5] S. Fagan and R. Gençay, "An introduction to textual econometrics," *Handbook of empirical economics and finance*, pp. 133–153, 2010.

[6] T. Kipf and M. Welling, "Variational graph auto-encoders," *ArXiv*, vol. abs/1611.07308, 2016.

[7] G. Aditya and L. Jure, "node2vec: Scalable feature learning for networks," 2016.

[8] J. Leskovec, "Stanford cs224wmachine learning with graphs." http://cs224w.stanford.edu.

[9] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko, "A gentle introduction to graph neural networks," *Distill*, 2021. https://distill.pub/2021/gnn-intro.

[10] M. W. Thomas N. Kipf, "Semi-supervised classification with graph convolutional networks," 2017.

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2017.

[12] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[13] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 1725–1735, PMLR, 13–18 Jul 2020.

[14] F. Groh, L. Ruppert, P. Wieschollek, and H. Lensch, "Ggnn: Graph-based gpu nearest neighbor search," *IEEE Transactions on Big Data*, pp. 1–1, 2022.