Supervised Community Detection with Line Graph Neural Networks

Chen, Li, and Bruna

ICLR 2019

Presenter: Jack Lanchantin

`https://qdata.github.io/deep2Read`

Outline

1 Introduction

2 Power Graph Neural Networks

3 Line Graphs and Non-Backtracking Operator

4 Loss Function

5 Experiments

6 Results

Outline

1 Introduction

2 Power Graph Neural Networks

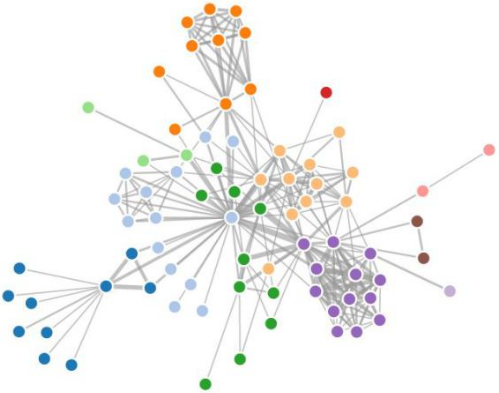
3 Line Graphs and Non-Backtracking Operator

4 Loss Function

5 Experiments

6 Results

Community Detection



Class of node classification tasks that attempt to discover a clustered, segmented structure within a graph.

Problem Setup

- **Given:** input graph $G = (V, E)$
- **Goal:** partition V into C groups. I.e. classify each node into one of C classes: $y : V \to \{1, C\}$
- We assume that a training set $\{(G_t, y_t)\}_{t \le T}$ is given, which we use to learn a model $\hat{y} = \Phi(G, \theta)$ trained by minimizing

$$L(\theta) = \frac{1}{T} \sum_{t \le T} \ell(\Phi(G_t, \theta), y_t) \ .$$

Outline

1 Introduction

2 Power Graph Neural Networks

3 Line Graphs and Non-Backtracking Operator

4 Loss Function

5 Experiments

6 Results

Graph Operators

Given vertices matrix $x \in \mathbb{R}^{|V| \times b}$, we consider the following graph intrinsic linear operators that act locally on x

- **adjacency operator** A is the linear map given by the adjacency matrix $A_{i,j} = 1$ iff $(i,j) \in E$.
- **degree operator** is the linear map $D : F \mapsto DF$ where $(Dx)_i := deg(i) \cdot x_i$, $D(x) = \text{diag}(A\mathbf{1})x$
- **power graph adjacency operator** $A_j = \min(1, A^{2^j})$ encodes 2^j-hop neighborhoods into a binary graph
    - J-th powers of A encode J-hop neighborhoods of each node, and allow us to combine and aggregate local information at different scales

Graph Neural Networks Using a Family of Graph Operators

Given a family of operators $\mathcal{F}_A^J = \left\{I, D, A, A_{(2)}, \ldots, A_{(J)}\right\}$ on graph G, we define a multiscale GNN for each $i \in V$:

$$z^{(k+1)} = \sigma\left(\sum_{O_i \in \mathcal{F}_A^J} O_i x^{(k)} \theta_i\right) \tag{1}$$

$$\overline{z}^{(k+1)} = \sum_{O_i \in \mathcal{F}_A^J} O_i x^{(k)} \theta_i \tag{2}$$

$$x^{(k+1)} = \left[z^{(k+1)}, \overline{z}^{(k+1)}\right] \in \mathbb{R}^{|V| \times b_{k+1}} \tag{3}$$

where $\theta_j \in \mathbb{R}^{b_k \times \frac{b_{k+1}}{2}}$ are trainable parameters, and σ is a ReLU

Outline

1 Introduction

2 Power Graph Neural Networks

3 Line Graphs and Non-Backtracking Operator

4 Loss Function

5 Experiments

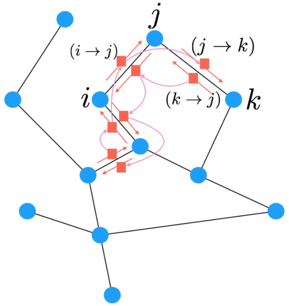
6 Results

Non-Backtracking Operator

- Previous methods (Krzakala et al., 2013) showed an improvement on spectral methods for community detection by using the *non-backtracking operator*
- This operator is defined over the edges of the graph and allows a directed flow of information even when the original graph is undirected

Line Graphs

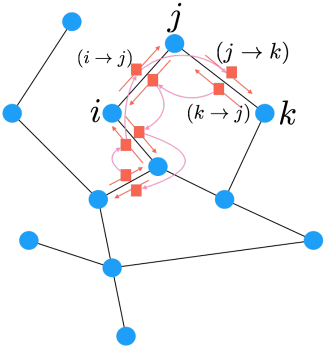
- The **line graph** $L(G) = (V_L, E_L)$ is the graph representing the edge adjacency structure of undirected graph $G = (V, E)$
- Vertices V_L of $L(G)$ are the **ordered edges** in E:
  $V_L = \{(i \to j);\ (i,j) \in E\} \cup \{(j \to i);\ (i,j) \in E\}$, so $|V_L| = 2|E|$



Non-Backtracking Operator

- The **non-backtracking operator** on the line graph is represented by $B \in \mathbb{R}^{2|E| \times 2|E|}$ encoding the edge adjacency structure. Two nodes in $L(G)$ are connected if:

$$B_{(i \to j),(i' \to j')} = \begin{cases} 1 & \text{if } j = i' \text{ and } j' \neq i \,, \\ 0 & \text{otherwise.} \end{cases}$$



Enables the propagation of directed information through the graph

GNN on Line Graphs with Non-Backtracking Operator

- A natural extension of the GNN architecture is thus to consider a second GNN defined on $L(G)$, generated by the corresponding non-backtracking operators $B_{(j)}$ and degree $D_B = \mathrm{diag}(B\mathbf{1})$
- This defines *edge features* that are diffused and updated according to the edge adjacency of G

GNN on Line Graphs with Non-Backtracking Operator

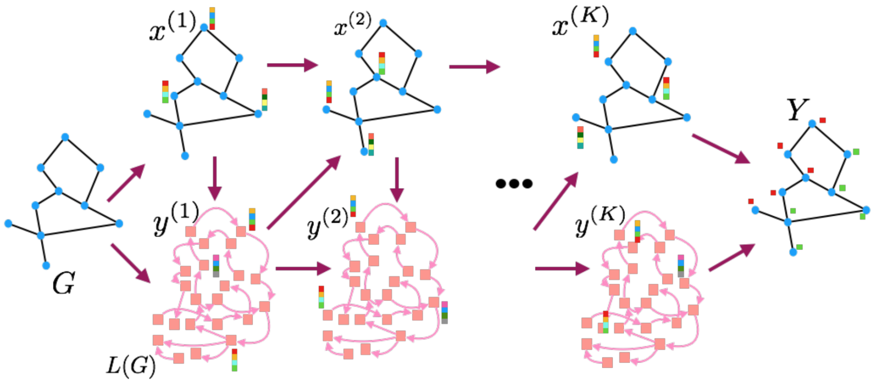
- Given graph operators $\mathcal{F}_A = \left\{I, D, A, A_{(2)}, \ldots, A_{(J)}\right\}$ and line graph operators $\mathcal{F}_B = \left\{I_B, D_B, B, B_{(2)}, \ldots, B_{(J)}\right\}$
- Edge and node features are combined at each layer using edge indicator matrices $\mathrm{Pm}, \mathrm{Pd} \in \{0,1\}^{|V| \times 2|E|}$, defined as $\mathrm{Pm}_{i,(i \to j)} = 1$, $\mathrm{Pd}_{j,(i \to j)} = 1$, $\mathrm{Pd}_{i,(i \to j)} = 1$, $\mathrm{Pd}_{j,(i \to j)} = -1$ and 0 otherwise. $\mathcal{F}_{AB} = \{\mathrm{P_m}, \mathrm{P_d}\}$

$$x^{(k+1)} = \sigma \left[\sum_{O_i \in \mathcal{F}_A} O_i x^{(k)} \theta_i + \sum_{O_j' \in \mathcal{F}_{AB}} O_j' y^{(k)} \theta_i' \right] \tag{4}$$

$$y^{(k+1)} = \sigma \left[\sum_{O_l'' \in \mathcal{F}_B} O_l'' y^{(k)} \theta_i'' + \sum_{O_j' \in \mathcal{F}_{AB}} \left(O_j'\right)^T x^{(k+1)} \theta_j''' \right] \tag{5}$$

where $\theta_i, \theta_i', \theta_i'' \in \mathbb{R}^{b_k \times b_{k+1}}$ and $\theta_i''' \in \mathbb{R}^{b_{k+1} \times b_{k+1}}$ are the learnable parameters. $x^{(0)} = \deg(A)$ and $y^{(0)} = \deg(B)$

Line Graph Neural Networks (LGNNs)



Relationship between LGNN and edge feature learning approaches

- **LGNNs:** learning *directed* edge features from an *undirected* graph
- If each node i contains two distinct sets of features $x_s(i)$ and $x_r(i)$, the non-backtracking operator constructs edge features from node features while preserving orientation
- GATs (Velickovic et. al.) and other similar models learn directed edge features on undirected graphs using stochastic matrices as adjacencies

Outline

1 Introduction

2 Power Graph Neural Networks

3 Line Graphs and Non-Backtracking Operator

4 Loss Function

5 Experiments

6 Results

Loss Function

No overlap in communities

- Let $\mathcal{C} = \{1, \ldots, C\}$ denote the possible community labelings that each node can take. $y \in \mathcal{C}^V$ is the ground truth community structure
- Softmax at each node's output gives the conditional probability that node i belongs to community c: $o_{i,c} = p(y_i = c \mid \theta, G)$
- Since community structure is defined up to global permutations of the labels, we can define a loss function w.r.t a given graph instance as:

$$\ell(\theta) = \inf_{\pi \in S_{\mathcal{C}}} - \sum_{i \in V} \log o_{i,\pi(y_i)} \ , \tag{6}$$

where $S_{\mathcal{C}}$ denotes the permutation group of C elements. This is essentially taking the the cross entropy loss minimized over all possible permutations of $S_{\mathcal{C}}$.

Loss Function

Overlap in communities (nodes can belong to multiple communities)

- If communities may overlap, we can enlarge $\mathcal{C}$ to include subsets of communities and define the permutation group accordingly
- For example, if there are two overlapping communities, we let $C = \{\{1\},\{2\},\{1,2\}\}$ and only allow the permutation between 1 and 2 when computing the loss function

Outline

1 Introduction

2 Power Graph Neural Networks

3 Line Graphs and Non-Backtracking Operator

4 Loss Function

5 Experiments

6 Results

Experiments

- We present experiments on synthetic community detection, as well as real-world detection
- Performance measure is the overlap between predicted ($\hat{y}$) and true labels (y), which quantifies how much better than random guessing a predicted labelling is
  - The overlap is given by $\left(\frac{1}{n}\sum_u \delta_{y(u),\hat{y}(u)}\right)$ where δ is the Kronecker delta function, and the labels are defined up to global permutation

Binary Stochastic Block Model

- Random graph model with planted community structure.
- Assign $|V| = n$ nodes to C classes at random with $y : V \to \{1, C\}$ and draw an edge connecting any two vertices u, v independently at random with probability p if $y(v) = y(u)$, and with probability q otherwise
- The sparse binary case $C = 2$ when $p, q \simeq 1/n$ is well understood and provides an initial platform to compare the GNN against provably optimal recovery algorithms

Outline

1 Introduction

2 Power Graph Neural Networks

3 Line Graphs and Non-Backtracking Operator

4 Loss Function

5 Experiments

6 Results

BSM Results

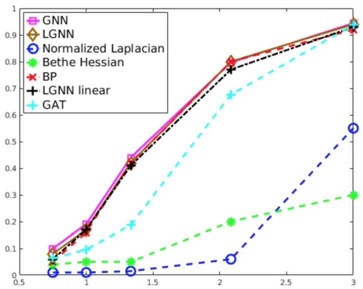
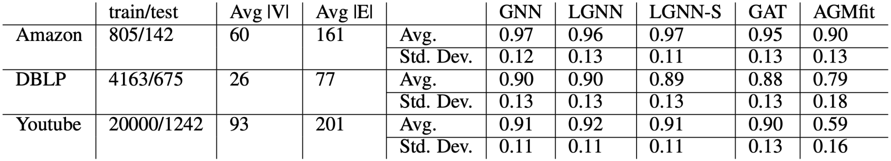


Figure 3. Binary associative SBM detection ($C = 2$, p > q). X-axis corresponds to SNR, and Y-axis to overlap between the prediction and the ground truth.

Real Datasets from SNAP



	train/test	Avg |V|	Avg |E|		GNN	LGNN	LGNN-S	GAT	AGMfit
Amazon	805/142	60	161	Avg.	0.97	0.96	0.97	0.95	0.90
				Std. Dev.	0.12	0.13	0.11	0.13	0.13
DBLP	4163/675	26	77	Avg.	0.90	0.90	0.89	0.88	0.79
				Std. Dev.	0.13	0.13	0.13	0.13	0.18
Youtube	20000/1242	93	201	Avg.	0.91	0.92	0.91	0.90	0.59
				Std. Dev.	0.11	0.11	0.11	0.13	0.16

Table 2: Comparison of the node classification accuracy by different models on the three SNAP datasets. Note that the average accuracy was computed graph-wise with each graph weighted by its size, while the standard deviation was computed graph-wise with equal weights among the graphs.

Conclusion

- Propose modifications to the GNN architecture, which allow it to exploit edge adjacency information, by incorporating the non-backtracking operator of the graph
- Con: didn't justify the use of line graphs and the non-backtracking operator enough
- Con: Assumes fixed number of communities