Graph Neural Network in Network Science

Applications of GNNs in clustering and community detection.

Michał Bujak

Jagiellonian University, Kraków, Poland

August 2, 2023

Outline

Types of underlying graphs and methods

GNN Framework

Structures:[1]

- **directed/undirected**;
- **homogeneous/heterogeneous** (nodes and links are of the same or different type);
- **static/dynamic** (fixed, evolving over time).

Loss function design:

- **node-level** (discrete classification of nodes or continuous assignment of values);
- **link-level** (classify edge type or predict its existence);
- **graph-level** (classification, regression, matching).

[1] Classification proposed by Zhou et al., "Graph neural networks: A review of methods and applications"

Supervision

GNN Framework

Supervision levels:

- **supervised learning** (labelled data);
- **semi-supervised learning** (a small amount of labelled nodes, a large amount of unlabelled nodes for training)
  - **transductive setting** (predict given unlabelled nodes);
  - **inductive setting** (provide new unlabelled nodes from the same distribution to infer);
- **unsupervised setting** (only unlabelled data).

Computational modules

GNN Framework

Computational modules:

- **Propagation module.** Propagate information between nodes: capture features and topological properties. **Convolution operator** and **recurrent operator** are used to aggregate information from neighbours, **skip connection** is used to gather information from historical representations (mitigation of over-smoothing).
- **Sampling module.** Sampling is usually needed for large graphs.
- **Pooling module.** Extract more general information from high-level graphs.

General pipeline

GNN Framework

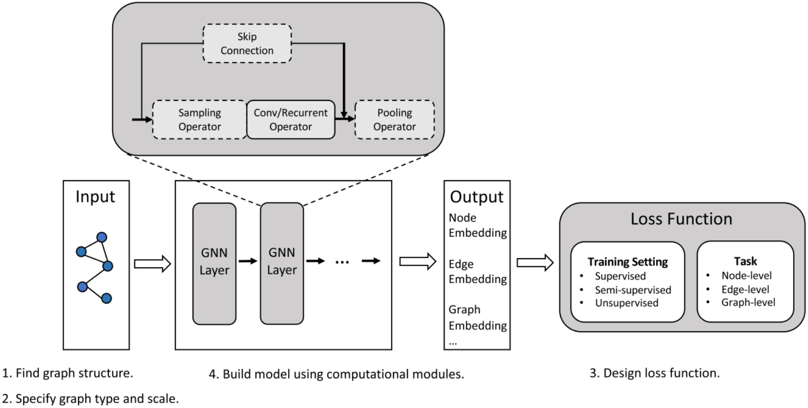


Figure: The general design pipeline for a GNN model. Zhou et al., "Graph neural networks: A review of methods and applications"

Propagation modules - convolution operator

Spectral approach

Degree matrix:

$$D = \{d_{ij}\}_{i,j \leq N}, d_{ii} = \deg(i), d_{ij} = 0 \text{ for } i \neq j$$

Graph Laplacian and normalised graph Laplacian:

$$\hat{L} := D - A, L := I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

L is a real symmetric matrix, it has a complete set of orthonormal eigenvectors, which we denote by $\{u_l\}_{l=1 \dots N}$. Associated real-non-negative eigenvalues $\{\lambda_l\}_{l=1 \dots N}$. Graph Fourier transform:

$$\hat{f}(\lambda_l) :=< f, u_l >= \sum_{i=1}^{N} f(i) u_l^*(i)$$

Factorisation $L = U \Lambda U^T$, where Λ is a diagonal matrix of eigenvalues. Convolution operation:

$$g \star x = \mathcal{F}^{-1}(\mathcal{F}(g) \odot \mathcal{F}(x)) = U(U^T g \odot U^T x)$$

Outline

Communities

Visualisation

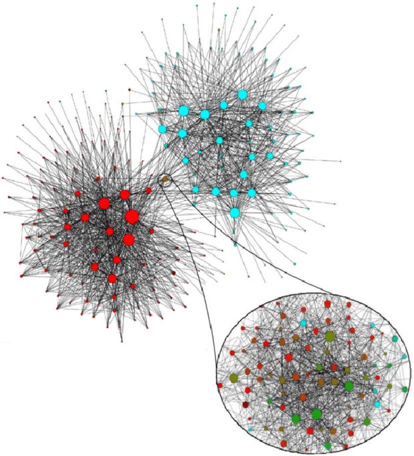


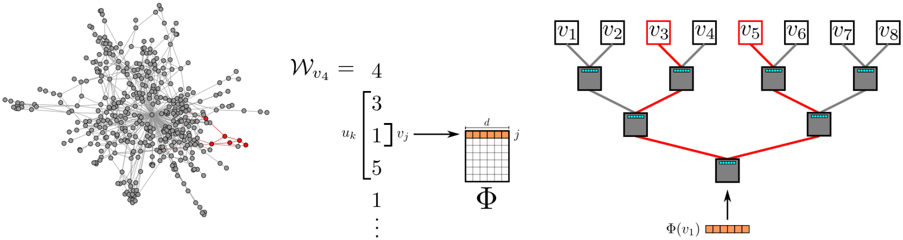
Figure: French and Dutch majorities in Belgium. Fortunato and Castellano, "Community Structure in Graphs"

DeepWalk

Online Learning of Social Representations

Perozzi, Al-Rfou, and Skiena, “Deepwalk: Online learning of social representations”

- Goal: embed the graph into Euclidean space.
- Based on random walks.
- Inspired by NLP (short walks as corpus, and nodes as vocabulary).
- Nodes close in the latent space have high probability to be close in the random walks.



(a) Random walk generation. (b) Representation mapping. (c) Hierarchical Softmax.

Neural Overlapping Community Detection

Shchur and Günnemann, “Overlapping community detection with graph neural networks”
Affiliation matrix $F \in \mathbb{R}_{\geq 0}^{|V| \times |C|}$, where C - communities. Bernoulli–Poisson (BP) graph generating model:

$$A_{uv} \sim \text{Bernoulli}(1 - \exp\left(-F_u F_v^T\right))$$

$F := \text{GNN}_\theta(A, X)$. The negative log-likelihood:

$$-\log p(A|F) = -\sum_{(u,v)\in E} \log(1 - \exp(-F_u F_v^T)) + \sum_{(u,v)\notin E} F_u F_v^T$$

Second term has much larger contribution[2]. Balanced loss function:

$$\mathcal{L}(F) = -\mathbb{E}_{(u,v)\sim P_E}(\log(1 - \exp(-F_u F_v^T))) + \mathbb{E}_{(u,v)\sim P_N}(F_u F_v^T)$$

$$\theta^\star = \arg\min_\theta \mathcal{L}(GNN_\theta(A, X))$$

[2] Real-world networks are usually sparse.

Communities

Modularity

Let d_v denote degree of a node v, m number of edges in the graph, and $\mathbf{A}$ graph's adjacency matrix. Modularity measures the partition of the graph into $c_i, i = 1 \dots k$ communities.

$$\mathcal{Q} = \frac{1}{2m} \sum_{ij} [\mathbf{A}_{ij} - \frac{d_i d_j}{2m}] \delta(c_i, c_j),$$

where $\delta(c_i, c_j)$ is a binary indicator variable. In the matrix form, $C \in \mathcal{M}(\{0, 1\})^{n,k}$ - cluster assignment, $\mathbf{B} := \mathbf{A} - \frac{dd^T}{2m}$, where d - degree vector:

$$\mathcal{Q} = \frac{1}{2m} \mathrm{Tr}(C^T \mathbf{B} C)$$

Relaxed, spectral version, computed efficiently: $C \in \mathcal{M}(\mathbb{R})^{n,k}$. Computation:

$$\mathbf{B}x = \mathbf{A}x - \frac{d^T x d}{2m}.$$

GNN construction

Google Research team

Model introduced in Müller, "Graph clustering with graph neural networks":

- Transductive GNN that outputs a single embedding per node.
- Start with $X^0 \in \mathbb{R}^{n \times s}$ - initial node features.
- $\hat{\mathbf{A}} = D^{-\frac{1}{2}} \mathbf{A} D^{-\frac{1}{2}}$ - normalised adjacency matrix.
- output of the t-th layer:

$$X^{t+1} = \mathrm{SeLU}(\hat{\mathbf{A}} X^t W + X W_{\mathrm{skip}}).$$

$$\mathrm{SeLU}(x) = \begin{cases} \lambda x, x > 0, \\ \lambda \alpha (e^x - 1), x \leq 0, \end{cases} \tag{1}$$

where $\lambda = 1.05070098, \alpha = 1.67326324$.

DMoN

Deep Modularity Networks

1. Encode cluster assignments:

$$C = \mathrm{softmax}(GCN(\hat{\mathbf{A}}, X))$$

$$\mathrm{softmax} : \mathbb{R}^K \ni z \to \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \in (0,1)^K$$

2. Loss function based on spectral modularity maximisation and regularisation (prevent trivial solutions)

$$\mathcal{L}_{\mathrm{DMoN}}(C, A) = \underbrace{-\frac{1}{2m}\mathrm{Tr}(C^T BC)}_{\text{modularity}} + \underbrace{\frac{\sqrt{k}}{n}\|\sum_i C_i^T\|_F - 1}_{\text{collapse regularisation}}$$

$\|\cdot\|_F$ is the Frobenius norm[3].

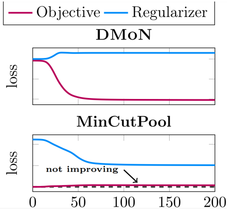
[3] $\|A\|_F = \sqrt{\sum_{i,j} |a_{ij}|^2} = \sqrt{\mathrm{tr}(A^* A)}$.

Modularity problem

Problem with a loss function based only on the modularity criterion

Problem: spectral clustering for modularity objective has spurious minima - assignment all nodes to the same cluster. Bianchi, Grattarola, and Alippi, "Spectral clustering with graph neural networks for graph pooling" suggested **MinCutPool**. Regularisation was based on the soft-orthogonal regularisation $\|C^T C - I\|_F$.

- Overly restrictive in combination with softmax class assignment.
- Regularisation dominates the clustering term (worse than random).



DMoN:

- normalised to range $[0, \sqrt{k}]$ (0 when perfectly balanced, $\sqrt{k}$ when all clusters are of size 1)
- applied dropout in GNN before the softmax (prevention of local optima).

Outline

GNN Framework

Community detection

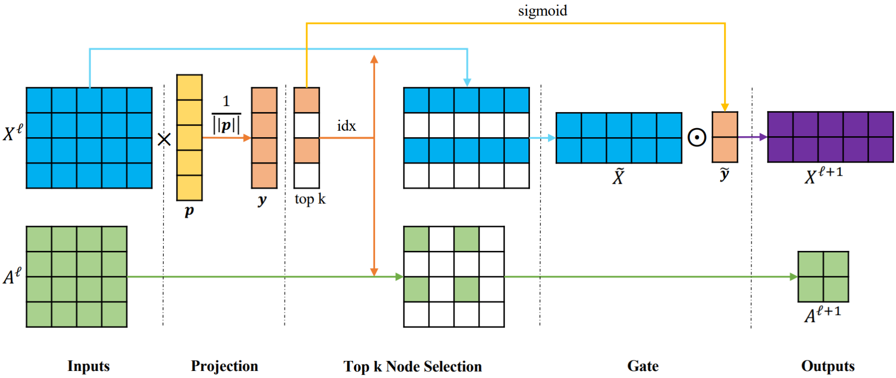
Graph autoencoder

Application in the urban mobility

Graph autoencoder

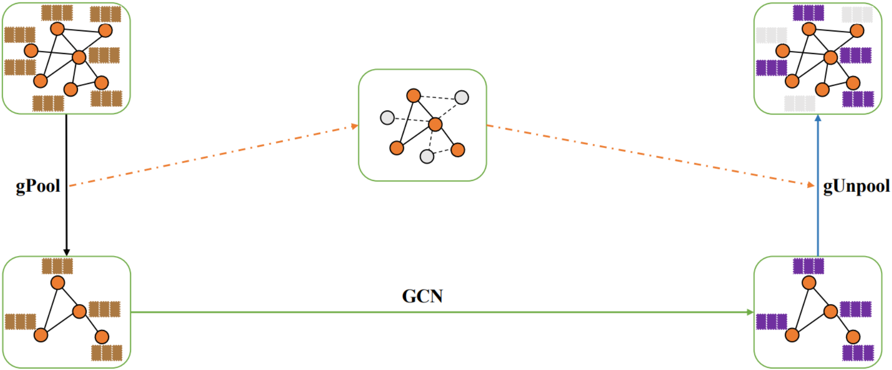
Encoding *gPool*

Gao and Ji, "Graph u-nets"



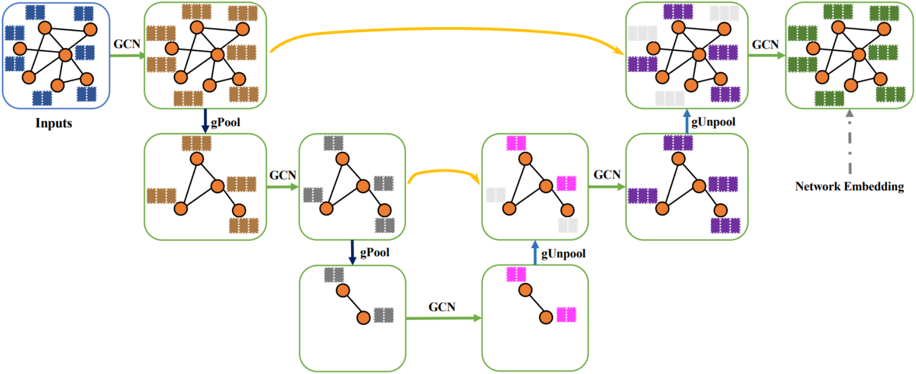
Graph autoencoder

Decoding *gUnpool*



Graph autoencoder

g-U-Nets framework



Outline

GNN Framework

Community detection

Graph autoencoder

Application in the urban mobility

Ride-pooling

Ride-pooling:

- Sharing a ride with different passengers;
- discomfort caused by delay (detour & waiting time);
- compensation with lower fare (sharing discount).

Benefits:

- reduced vehicle mileage (environment);
- decrease in fleet size (operator, city (congestion reduction));
- lower costs (users).

Two algorithmic stages:

- determining the set of the feasible rides;
- matching (finding optimal solution).

Ride-pooling

Ride-pooling:

- Sharing a ride with different passengers;
- discomfort caused by delay (detour & waiting time);
- compensation with lower fare (sharing discount).

Benefits:

- reduced vehicle mileage (environment);
- decrease in fleet size (operator, city (congestion reduction));
- lower costs (users).

Two algorithmic stages:

- determining the set of the feasible rides;
- matching (finding optimal solution).

Ride-pooling

Ride-pooling:

- Sharing a ride with different passengers;
- discomfort caused by delay (detour & waiting time);
- compensation with lower fare (sharing discount).

Benefits:

- reduced vehicle mileage (environment);
- decrease in fleet size (operator, city (congestion reduction));
- lower costs (users).

Two algorithmic stages:

- determining the set of the feasible rides;
- matching (finding optimal solution).

Ride-pooling

Determining the set of feasible rides

Presented algorithmic steps are according to the ExMAS algorithm[4].
Ride comprising travellers $T_1, \ldots, T_n$ is considered feasible if it is attractive for all of $T_1, \ldots, T_n$.
Utility formulas (traveller specific):

$$\begin{aligned} U_i^{ns} &= \rho l_i + \beta_t t_i \\ U_{i,r_k}^{s} &= (1-\lambda)\rho l_i + \beta_t \beta_s (\hat{t}_i + \beta_d \hat{t}_i^p), \end{aligned} \tag{2}$$

- ρ - price ($\$/\mathrm{km}$);
- λ - sharing discount;
- l_i - trip length;
- β_t - value of time;
- β_s - sharing discomfort;
- β_d - delay sensitivity;
- t_i, $\hat{t}_i$ - travel time with the non-shared ride and shared rides, respectively;
- $\hat{t}_i^p$ - pick-up delay.

[4] Kucharski and Cats, "Exact matching of attractive shared rides (ExMAS) for system-wide strategic evaluations".

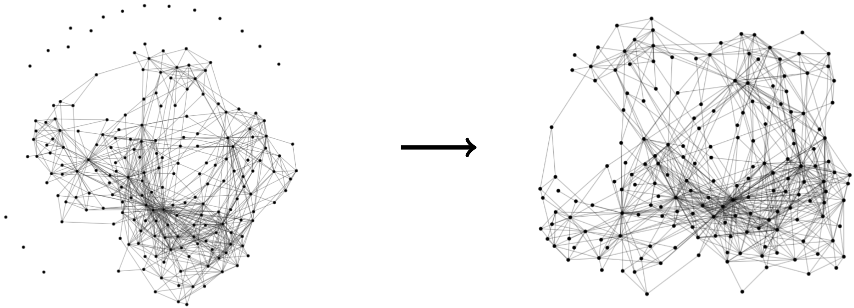
Potential shared rides

	pickup_datetime	origin	destination
0	2016-01-02 18:30:02	42440639	42440009
1	2016-01-02 18:30:29	42431106	42438798
2	2016-01-02 18:30:37	42438889	42430347
3	2016-01-02 18:30:47	42428179	42437343
4	2016-01-02 18:58:14	42442895	42437343
...	...	...	...

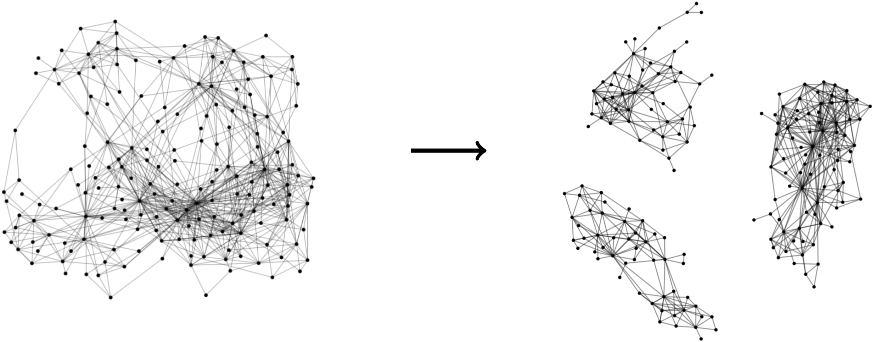
$\longrightarrow$

travellers_id	shared_time	solo_time
[54, 100]	1229	1715
[53, 100]	1118	1787
[19, 100]	1055	1817
[66, 100]	1269	1888
[20, 62]	1035	1293
...	...	...

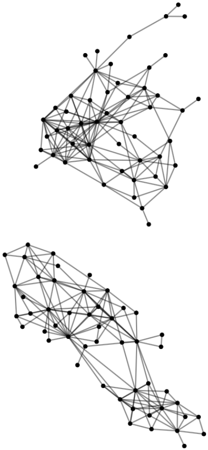
Graph representation



Frame Title



Benchmarks



Graph partition	Objective
Isolated nodes	124k
No partition	84.8k
Random partition (3)	110 - 120k
Classic algorithms	89.7 - 110k
Our	94.4k

New Method

- If u travels with v, v travels with u;
- u and v can only travel together if assigned to the same cluster;
- edge weight $s_{uv} = t_u^{ns} + t_v^{ns} - t_{\{u,v\}}^s$;
- $p_v = (p_v^1, \ldots, p_v^k) = (P(v \in c_1), \ldots, P(v \in c_k))$;
- denote r_{uv} attractiveness of matching u to v (in $[0,1]$), for example starting from $\sigma(s_{uv})$;
- edge uv attractiveness $q_{uv} = r_{uv} r_{vu}$;
- $P_{uv} = \sum_{c \in C} p_u^c p_v^c$;
- probability that u travels with q define as $w_{uv} = P_{uv} * q_{uv}$;
- loss function:

$$\sum_{u \in V} [\sum_{v \in N(u)} w_{uv} s_{uv} + (1 - \sum_{v \in N(u)} w_{uv}) s_{uu}] + \lambda (\sum_{c \in C} |c| \log_2 |c|)^\alpha$$

Current problems

- In reality, ride-pooling scheme admits high order rides (more than 2 travellers).
  Hyper-graphs, heterogeneous, bipartite representation
- Finding optimal representation, node and edge features.
- Structuring architecture.