A Survey on Unsupervised Graph Representation Learning

Yuanhao Xiong, Xiangning Chen, Li-Cheng Lan, Lucas Tecot

Background

- Learning good latent representations from raw features is key to machine learning problems.
- A more practical scenario: unsupervised setting
- Great progresses in computer vision such as MoCo and SimCLR, and natural language such as word2vec and BERT
- How about unsupervised graph representation learning (GRL)?

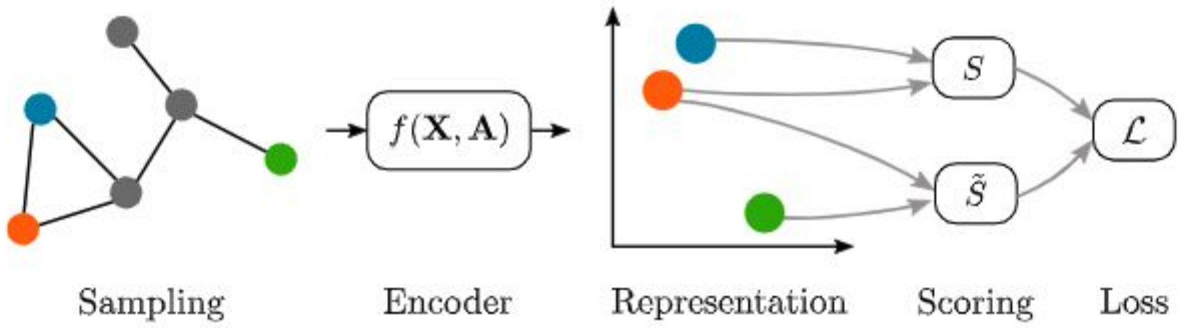
Selected Methods

- DeepWalk
- Node2vec
- GraphSAGE
- ARGA & ARVGA
- DGI
- GMI
- GCN + SS

Problem Formulation

- Undirected graph G = {V, E}
  - V is a set of nodes, E is a set of edges
  - e_k is in the form of (v_i, v_j)
- Feature matrix **X**
  - **X** = $\{x_1, \ldots, x_N\}$, N is the number of nodes in the graph
  - $x_i \in R^F$ represents the raw features of node i
- Adjacency matrix **A**
- Objective
  - Learn an encoder, E, $\mathbb{R}^{N\times F} \times \mathbb{R}^{N\times N} \rightarrow \mathbb{R}^{N\times F'}$
  - $E(X,A) = Z = \{z_1,z_2,\ldots,z_N\}$
  - No label information is leveraged

A General Framework

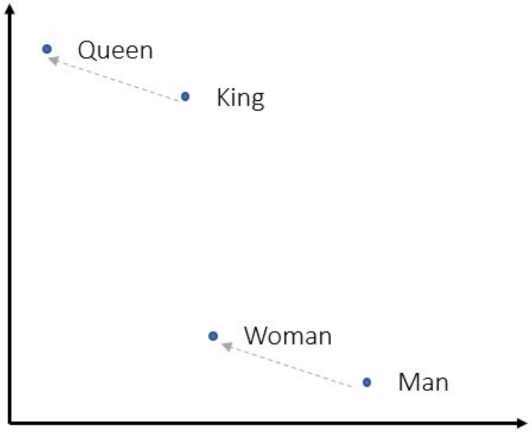


A General Framework

Table 1: A comparison of different methods.

Method	Encoder	Score	Loss	Sampling
DeepWalk (node2vec)	Embedding	$\sigma(\mathbf{z}_i^{\mathrm{T}}\mathbf{z}_j)$	$-\log(S) - \log(1 - \widetilde{S})$	$+$: random walk neighbors $-$: non-neighboring nodes[1]
GraphSAGE	GCN	$\sigma(\mathbf{z}_i^{\mathrm{T}}\mathbf{z}_j)$	$-\log(S) - \log(1 - \widetilde{S})$	$+$: random walk neighbors $-$: non-neighboring nodes
ARGE	GCN	$\sigma(\mathbf{z}_i^{\mathrm{T}}\mathbf{z}_j)$	$-\log(S) - \log(1 - \widetilde{S})$	$+$: prior distribution $-$: graph encoder
ARVGE	GCN	$\sigma(\mathbf{z}_i^{\mathrm{T}}\mathbf{z}_j)$	$-\log(S) - \log(1 - \widetilde{S}) + KL(p(\cdot)\|\|q(\cdot))$	$+$: prior distribution $-$: graph encoder
DGI	GCN	$\sigma(\mathbf{z}_i^{\mathrm{T}}\mathbf{W}\mathbf{s})$	$-\log(S) - \log(1 - \widetilde{S})$	$+$: original graph $-$: corrupted graph
GMI	GCN	$\sigma(\mathbf{z}_i^{\mathrm{T}}\mathbf{W}\mathbf{x}_j)$	See Eq. (2)	$+$: original graph $-$: corrupted graph

DeepWalk

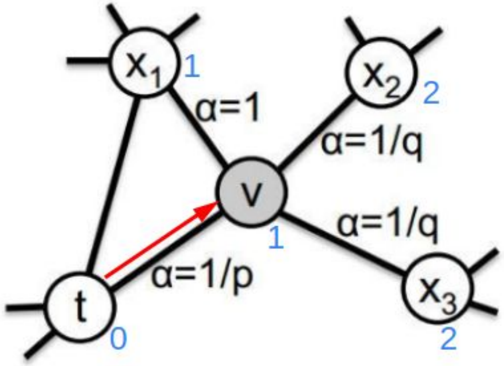


- Similar to the idea of word2vec that words appearing in the same context should be close in the high-dimensional embedding space

- DeepWalk defines the context of a certain node ***v*** as a sequence of nodes present in a random wandering starting from ***v***

- Use negative sampling to maximize the co-occurrence probability with a positive node in the context, and minimize the probability for another randomly sampled negative node

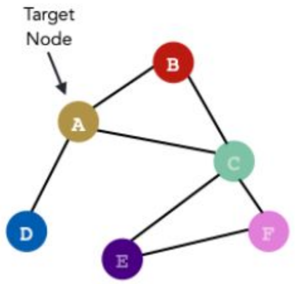
Node2Vec

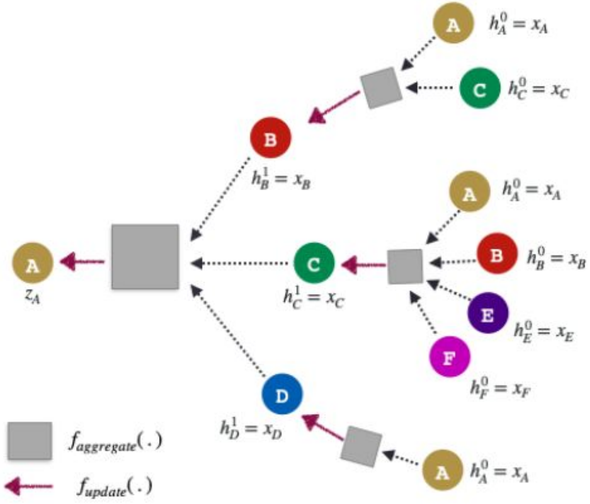


- Node2Vec uses biased random walk
  - Instead of BFS or DFS
- Assume we just random walk from node t to node v
  - The probability to each node is decided according to the distance with node t
    - distance=0: go back to the node t
    - distance=1: stay around the node t
    - distance=2: leave away from the node t
  - The probability is controlled by two parameters p and q
    - If p is small, the random walk be more local
    - If q is small, the random walk will walk further

GraphSAGE

- Method:
  - Generates the node embedding by using its neighbors
  - The neighbors are sampled to reduce time complexity
- Aggregate functions: MEAN, Pooling, or LSTM
- Inductive: Able to generate decent embedding for unseen nodes





ARGA & ARVGA

Adversarially Regularized Graph (Variational) Autoencoder

Encoder

$$f(\mathbf{Z}^{(l)}, \mathbf{A}|\mathbf{W}^{(l)}) = \phi(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}^{(l)}\mathbf{W}^{(l)})$$

Variational $\longrightarrow$ $q(\mathbf{z_i}|\mathbf{X}, \mathbf{A}) = \mathcal{N}(\mathbf{z}_i|\boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}^2))$

Decoder

$$p(\hat{\mathbf{A}}_{ij} = 1|\mathbf{z}_i, \mathbf{z}_j) = \mathrm{sigmoid}(\mathbf{z}_i^\top, \mathbf{z}_j)$$

Objective

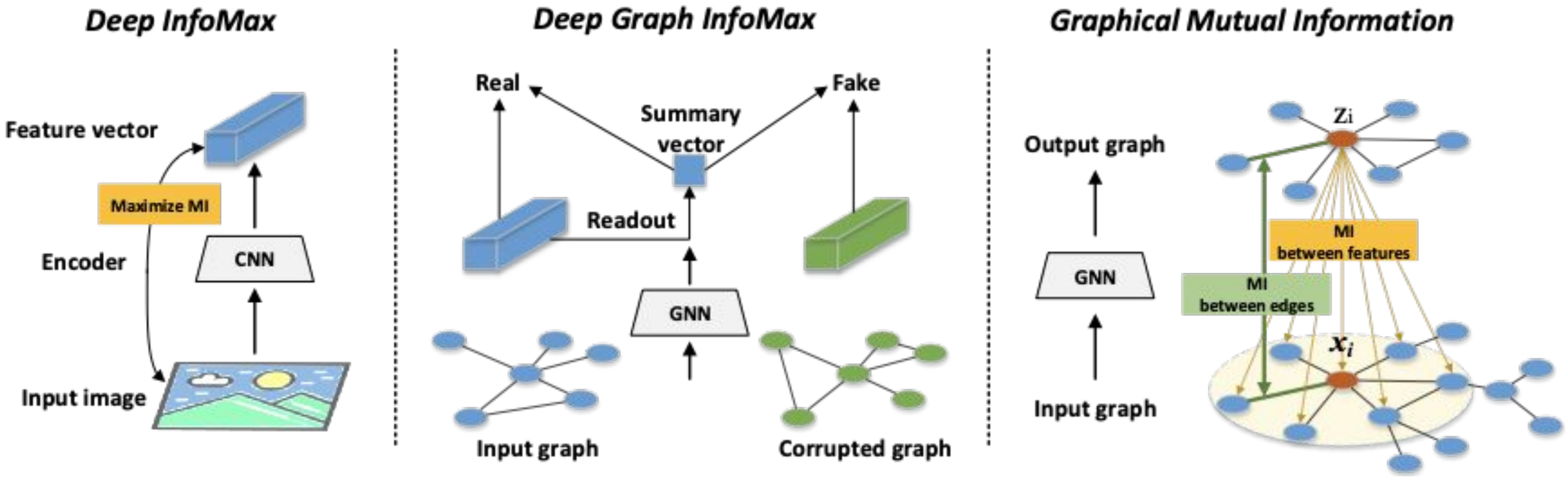
$$\mathbb{E}_{q(\mathbf{Z}|(\mathbf{X},\mathbf{A}))}[\log p(\hat{\mathbf{A}}|\mathbf{Z})]$$

Variational $\longrightarrow$ $\mathbb{E}_{q(\mathbf{Z}|(\mathbf{X},\mathbf{A}))}[\log p(\hat{\mathbf{A}}|\mathbf{Z})] - \mathbf{KL}[q(\mathbf{Z}|\mathbf{X}, \mathbf{A}) \parallel p(\mathbf{Z})]$

Adversarial Regulation

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathbb{E}_{\mathbf{z}\sim p_z}[\log \mathcal{D}(\mathbf{Z})] + \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[\log(1 - \mathcal{D}(\mathcal{G}(\mathbf{X}, \mathbf{A})))]$$

DGI & GMI



Deep InfoMax
Feature vector
Maximize MI
Encoder
CNN
Input image
Deep Graph InfoMax
Real
Fake
Summary vector
Readout
GNN
Input graph
Corrupted graph
Graphical Mutual Information
Output graph
GNN
Input graph
Zi
MI between features
MI between edges
x_i

DGI & GMI

- Objective of DGI:

$$\mathcal{L} = \frac{1}{N+M}\left(\sum_{i=1}^{N}\mathbb{E}_{(\mathbf{X},\mathbf{A})}\left[\log\mathcal{D}\left(\mathbf{z}_i,\mathbf{s}\right)\right] + \sum_{j=1}^{M}\mathbb{E}_{(\widetilde{\mathbf{X}},\widetilde{\mathbf{A}})}\left[\log\left(1-\mathcal{D}\left(\widetilde{\mathbf{z}}_j,\mathbf{s}\right)\right)\right]\right),$$

- Objective of GMI:

$$I(\mathbf{z}_i;\mathbf{x}_j) = -sp(-\mathcal{D}(\mathbf{z}_i,\mathbf{x}_j)) - \mathbb{E}_{\tilde{\mathbb{P}}}[sp(\mathcal{D}(\mathbf{z}_i,\mathbf{x'}_j))],$$

$$I(w_{ij};a_{ij}) = a_{ij}logw_{ij} + (1-a_{ij})log(1-w_{ij}).$$

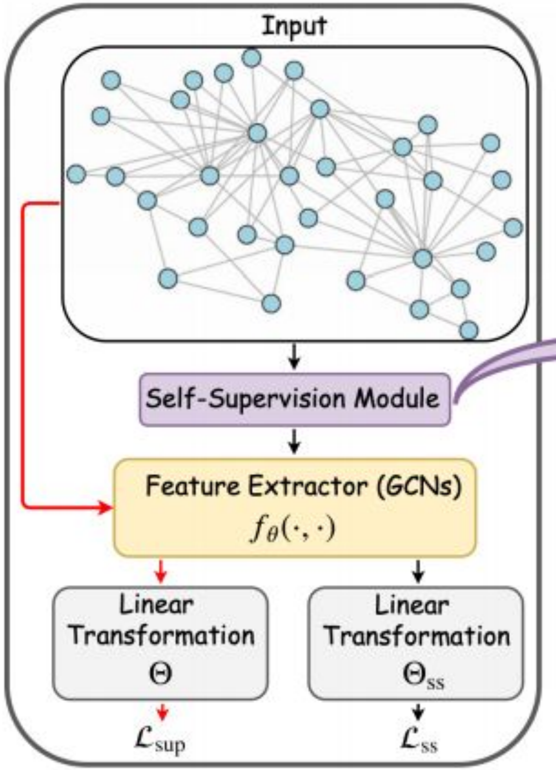
$$I(\mathbf{z}_i;\mathcal{G}_i) = \sum_{j}^{i_n} w_{ij}I(\mathbf{z}_i;\mathbf{x}_j) + I(w_{ij};a_{ij}),$$

$$with \quad w_{ij} = \sigma(\mathbf{z}_i^{\mathrm{T}}\mathbf{z}_j),$$

GCN with SS

- Self-Supervision Scheme
  - Pretrain-Finetune: little gain, especially for large datasets
  - Multi-Task: more effective as regularization term

- Pretext Tasks
  - Node clustering: simply based on the feature matrix
    - less informative when the dataset is large and the feature dimension is low
  - Graph partitioning: group nodes based on the topology
    - ensure that the number of connecting edges between subsets is minimized, general and effective



Pros and Cons

- DeepWalk & node2vec:
  - Simple but do not fully leverage the graph structure
  - Use look up table, can not generalize to other nodes
- ARGA & ARVGA
  - Encoder-decoder structure to obtain hidden features by reconstructing the graph
  - Focus more on adjacent relationships, less suitable for node classification
- DGI & GMI
  - Based on mutual information maximization, theoretical support
  - Sensitive to the mutual information estimator
- GCN with SS
  - Combined with supervised learning by fine-tuning the pretrained embeddings
  - More flexible

Evaluation Metric

- Node classification: accuracy
- Link prediction: average precision

Table 2: Statistics of the datasets used in experiments.

Dataset	Type	#Nodes	#Edges	#Features	#Classes
Cora	Citation network	2,708	5,429	1,433	7
Citeseer	Citation network	3,327	4,732	3,703	6
PubMed	Citation network	19,717	44,338	500	3

Experimental Results

Table 3: Summary of results in terms of classification accuracies. In the first column, we highlight the kind of data available to each method during training ($\mathbf{X}$: features, $\mathbf{A}$: adjacency matrix, $\mathbf{Y}$: labels). "GCN" corresponds to a two-layer GCN encoder trained in a supervised manner.

Available data	Method	Cora	Citeseer	Pubmed
$\mathbf{X}$	Raw features	$47.9 \pm 0.4\%$	$49.3 \pm 0.2\%$	$69.1 \pm 0.3\%$
$\mathbf{A}$	DeepWalk	$67.2 \pm 1.4\%$	$43.2 \pm 1.4\%$	$65.3 \pm 0.3\%$
$\mathbf{A}$	Node2Vec	$70.9 \pm 1.0\%$	$48.1 \pm 1.4\%$	$68.7 \pm 1.0\%$
$\mathbf{X}, \mathbf{A}$	GraphSAGE	$76.6 \pm 1.6\%$	$67.4 \pm 1.0\%$	$78.7 \pm 0.7\%$
$\mathbf{X}, \mathbf{A}$	ARGE	$78.6 \pm 0.2\%$	$70.2 \pm 0.7\%$	$76.7 \pm 0.6\%$
$\mathbf{X}, \mathbf{A}$	ARVGE	$80.5 \pm 0.9\%$	$70.0 \pm 1.3\%$	$77.7 \pm 0.9\%$
$\mathbf{X}, \mathbf{A}$	DGI	$81.8 \pm 0.6\%$	$70.7 \pm 0.8\%$	$76.8 \pm 0.8\%$
$\mathbf{X}, \mathbf{A}$	GMI	$\mathbf{82.0 \pm 0.1\%}$	$\mathbf{71.2 \pm 0.4\%}$	$\mathbf{79.5 \pm 0.4\%}$
$\mathbf{X}, \mathbf{A}, \mathbf{Y}$	Planetoid	75.7%	64.7%	77.2%
$\mathbf{X}, \mathbf{A}, \mathbf{Y}$	GCN	81.5%	70.3%	79.0%
$\mathbf{X}, \mathbf{A}, \mathbf{Y}$	GCN + SS (Clustering)	$81.6 \pm 0.6\%$	$70.7 \pm 0.8\%$	$78.8 \pm 0.4\%$
$\mathbf{X}, \mathbf{A}, \mathbf{Y}$	GCN + SS (Partition)	$\mathbf{81.8 \pm 0.7\%}$	$\mathbf{71.3 \pm 0.7\%}$	$\mathbf{80.0 \pm 0.7\%}$

Experimental Results

Table 4: Results of the link prediction task in citation networks. Average precision is reported.

Method	Cora	Citeseer	Pubmed
DeepWalk	92.0 ± 0.6	91.3 ± 0.5	91.3 ± 0.3
Node2Vec	91.7 ± 0.2	85.1 ± 1.0	76.3 ± 0.3
GraphSAGE	88.6 ± 1.1	84.8 ± 1.3	83.8 ± 0.5
ARGE	92.1 ± 0.3	88.0 ± 0.6	96.4 ± 0.9
ARVGE	92.5 ± 0.2	89.4 ± 0.7	$\mathbf{96.6} \pm \mathbf{0.4}$
DGI	91.7 ± 1.3	92.4 ± 0.6	94.9 ± 0.1
GMI	$\mathbf{95.5} \pm \mathbf{1.0}$	$\mathbf{94.3} \pm \mathbf{0.5}$	96.0 ± 0.2

Future Direction

- A more fair evaluation protocol.
- Negative sampling strategies.
- Generalization to large and realistic graphs.