EECE 571F: Deep Learning with Structures

Lecture 8:
Unsupervised Graph Representation Learning

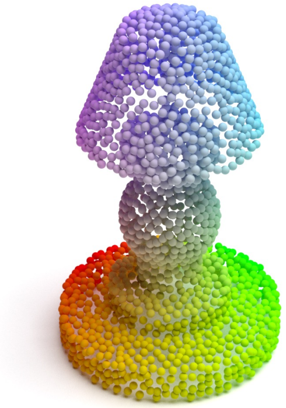
Renjie Liao

University of British Columbia

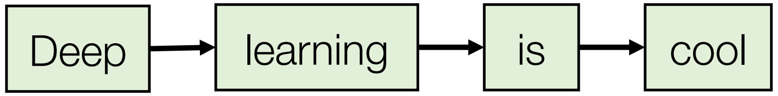
Winter, Term 2, 2021/22

Course Scope

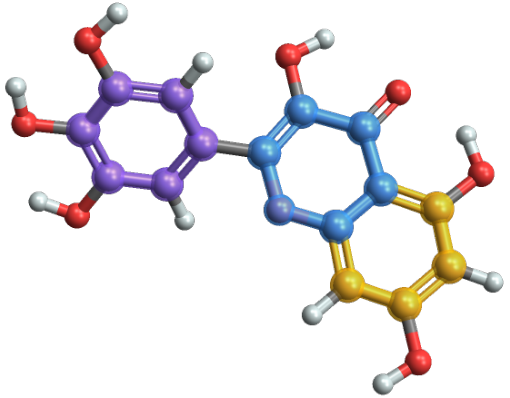
- Supervised Learning with Observable Structures
- **Unsupervised / Self-supervised Learning with Observable Structures**
- Supervised Learning with Latent Structures



Points/Sets



Lists/Sequences



Graphs

Image Credit: https://github.com/AnTao97/PointCloudDatasets
https://www.wolfram.com/language/12/molecular-structure-and-computation/molecule-graphs.html.en?product=mathematica

Unsupervised Graph Representation Learning

- Deep Generative Models of Graphs

  Building probabilistic distributions of graphs (e.g., adjacency matrix A) and node representations X

Unsupervised Graph Representation Learning

- Deep Generative Models of Graphs

  Building probabilistic distributions of graphs (e.g., adjacency matrix A) and node representations X

- In this lecture, we focus on methods that learn good node representations

  Given graphs (e.g., adjacency matrix A), the goal is to learn node representations X

  The learned X is useful for supervised fine-tuning, e.g., node classification

Unsupervised / Self-Supervised Learning

Since only data is given, we need a learning criterion:

Unsupervised / Self-Supervised Learning on Graphs

Since only data is given, we need a learning criterion:

- Likelihood (Autoregressive models)
- Reconstruction Loss (Auto-encoders)
- Contrastive Loss (Noise contrastive estimation, Self-supervised learning)
- Min-max Loss (Generative adversarial networks)

……

Unsupervised / Self-Supervised Learning on Graphs

Since only data is given, we need a learning criterion:

- **Likelihood (Autoregressive models)** DeepWalk [1] Node2Vec[2]
- Reconstruction Loss (Auto-encoders)
- Contrastive Loss (Noise contrastive estimation, Self-supervised learning)
- Min-max Loss (Generative adversarial networks)

……

Unsupervised / Self-Supervised Learning on Graphs

Since only data is given, we need a learning criterion:

- **Likelihood (Autoregressive models)** DeepWalk [1] Node2Vec[2]
- Reconstruction Loss (Auto-encoders)
- **Contrastive Loss (Noise contrastive estimation, Self-supervised learning)** DeepGraphInfoMax [3]
- **Min-max Loss (Generative adversarial networks)** DeepGraphInfoMax [3]

……

Language Models

Model the probability of words given its context (e.g., a fixed-size window):

$$p(w_{i+1}|w_i, \cdots, w_{i-K})$$

Language Models

Model the probability of words given its context (e.g., a fixed-size window):

$$p(w_{i+1}|w_i, \cdots, w_{i-K})$$

$$p(w_i|\{w_{i-k}, \cdots, w_{i-1}, w_{i+1}, \cdots, w_{i+k}\})$$

Language Models

Model the probability of words given its context (e.g., a fixed-size window):

$$p(w_{i+1}|w_i, \cdots, w_{i-K})$$

$$p(w_i|\{w_{i-k}, \cdots, w_{i-1}, w_{i+1}, \cdots, w_{i+k}\})$$

- One can use RNNs or CNNs to construct the probability
- The model can be learned via *maximum likelihood*

Skip-Gram Models

Model the probability of its context (e.g., a fixed-size window) given the word:

$$p(\{w_{i-k}, \cdots, w_{i-1}, w_{i+1}, \cdots, w_{i+k}\}|w_i) = \prod_{j \neq i, j \in \mathcal{N}_i} p(w_j|w_i)$$

Skip-Gram Models

Model the probability of its context (e.g., a fixed-size window) given the word:

$$p(\{w_{i-k}, \cdots, w_{i-1}, w_{i+1}, \cdots, w_{i+k}\}|w_i) = \prod_{j \neq i, j \in \mathcal{N}_i} p(w_j|w_i)$$

- Assumes conditional independence, i.e., the context is orderless
- The model only takes one word as input, thus being efficient
- Interpolation (context => word) vs. extrapolation (word => context)

Skip-Gram Models

Model the probability of its context (e.g., a fixed-size window) given the word:

$$p(\{w_{i-k}, \cdots, w_{i-1}, w_{i+1}, \cdots, w_{i+k}\} | w_i) = \prod_{j \neq i, j \in \mathcal{N}_i} p(w_j | w_i)$$

- Assumes conditional independence, i.e., the context is orderless
- The model only takes one word as input, thus being efficient
- Interpolation (context => word) vs. extrapolation (word => context)

Can we generalize this model to graphs?

Skip-Gram Models

Model the probability of its context (e.g., a fixed-size window) given the word:

$$p(\{w_{i-k}, \cdots, w_{i-1}, w_{i+1}, \cdots, w_{i+k}\}|w_i) = \prod_{j \neq i, j \in \mathcal{N}_i} p(w_j|w_i)$$

- Assumes conditional independence, i.e., the context is orderless
- The model only takes one word as input, thus being efficient
- Interpolation (context => word) vs. extrapolation (word => context)

Words -> Nodes

Can we generalize this model to graphs?

Skip-Gram Models

Model the probability of its context (e.g., a fixed-size window) given the word:

$$p(\{w_{i-k}, \cdots, w_{i-1}, w_{i+1}, \cdots, w_{i+k}\}|w_i) = \prod_{j \neq i, j \in \mathcal{N}_i} p(w_j|w_i)$$

- Assumes conditional independence, i.e., the context is orderless
- The model only takes one word as input, thus being efficient
- Interpolation (context => word) vs. extrapolation (word => context)

Can we generalize this model to graphs?

Words -> Nodes

How about edges?

Random Walks

A special Markov Chain

Starting at a node, one can randomly choose a neighboring node at a time to walk

Random Walks

A special Markov Chain

Starting at a node, one can randomly choose a neighboring node at a time to walk

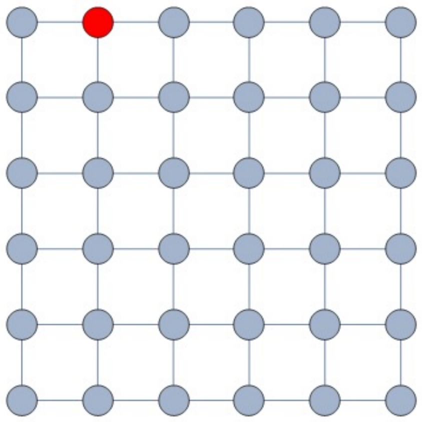
$$p_{ij} = \frac{1}{|\mathcal{N}_i|}$$

Random Walks

A special Markov Chain

Starting at a node, one can randomly choose a neighboring node at a time to walk

$$p_{ij} = \frac{1}{|\mathcal{N}_i|}$$



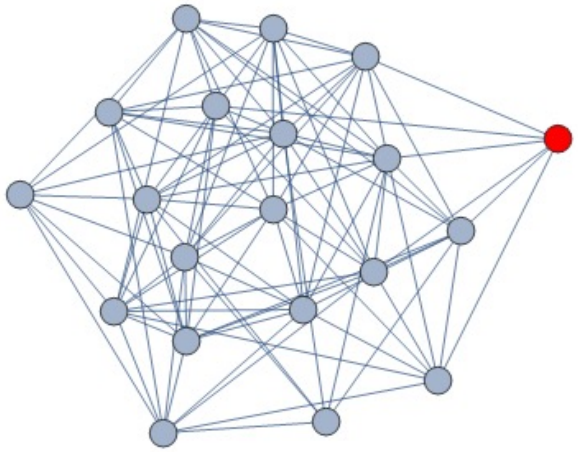
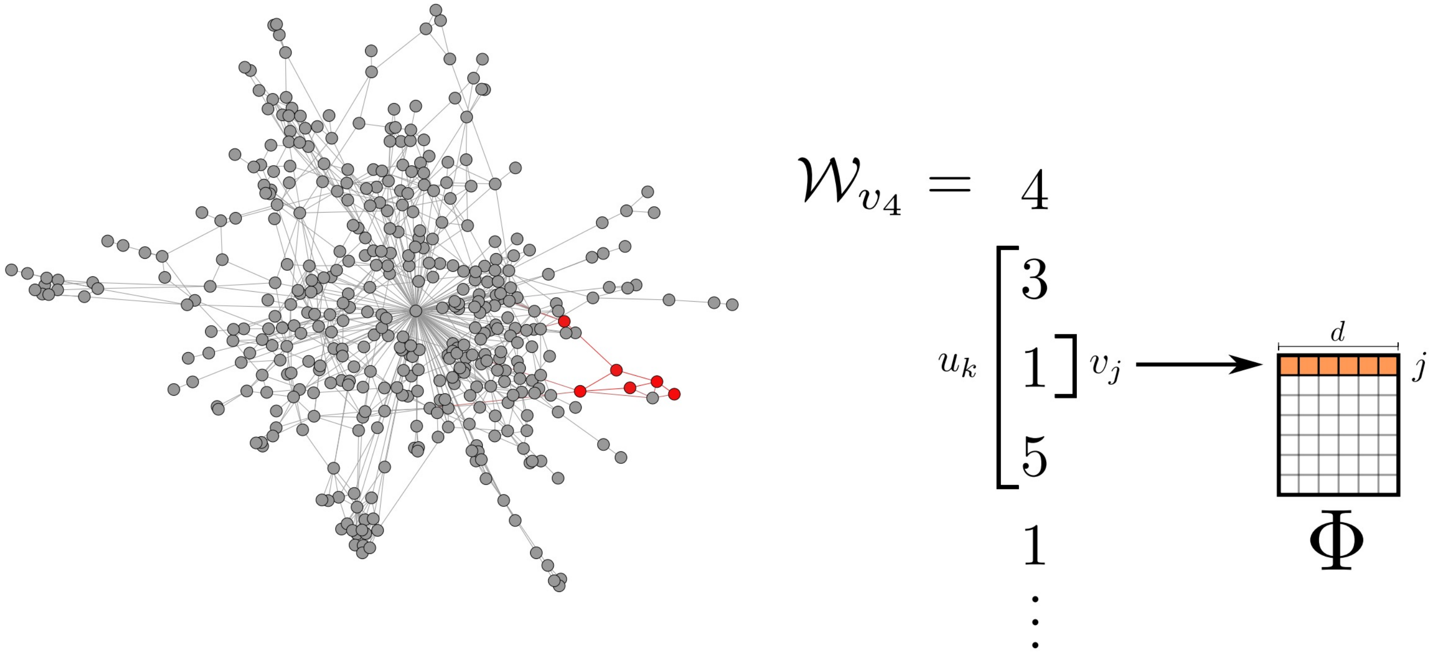


Image Credit: https://mathematica.stackexchange.com/questions/156626/generate-random-walk-on-a-graph

Deep Walks

Model the probability of context (random walks) given its word (vertex):



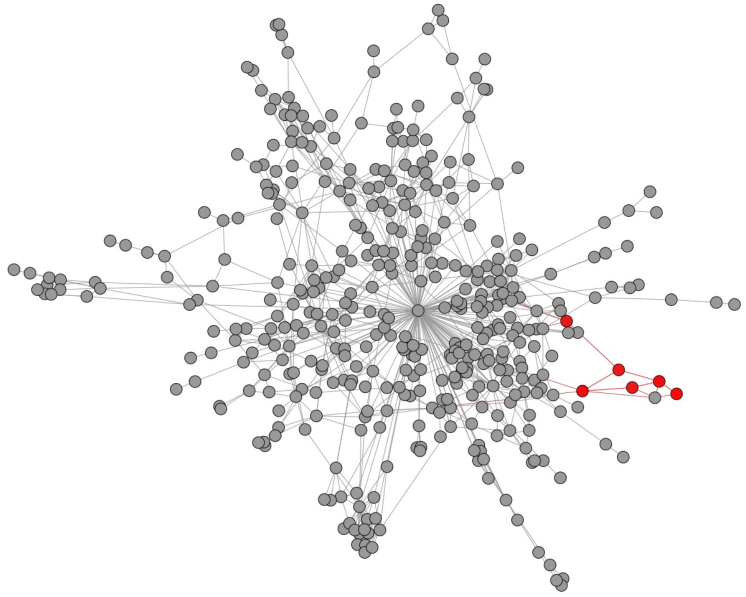
(a) Random walk generation. (b) Representation mapping.

Image Credit: [1]

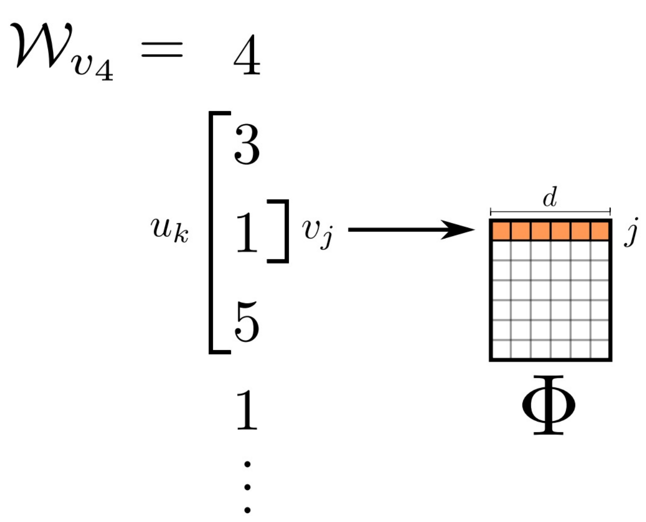
Deep Walks

Model the probability of context (random walks) given its word (vertex):

$$\max \quad \log p\left(\{v_{i-w}, \cdots, v_{i-1}, v_{i+1}, \cdots, v_{i+w}\} \mid \Phi(v_i)\right)$$



(a) Random walk generation.



(b) Representation mapping.

Image Credit: [1]

Deep Walks

Model the probability of context (random walks) given its word (vertex):

Algorithm 1 DeepWalk(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$
Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
  $\mathcal{O} = \text{Shuffle}(V)$
  for each $v_i \in \mathcal{O}$ **do**
    $\mathcal{W}_{v_i} = RandomWalk(G, v_i, \text{t})$
    SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
  end for
end for

Image Credit: [1]

Deep Walks

Skip-Gram Algorithm

Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

for each $v_j \in \mathcal{W}_{v_i}$ **do**
    for each $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ **do**
        $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
        $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
    end for
end for

Image Credit: [1]

Deep Walks

Skip-Gram Algorithm

Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

for each $v_j \in \mathcal{W}_{v_i}$ **do**
  for each $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ **do**
    $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
    $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
  end for
end for

Conditional Independence

$$p(\{u_{j-w}, \cdots, u_{j-1}, u_{j+1}, \cdots, u_{j+w}\} | \Phi(v_j)) = \prod_{k \neq j, k \in \mathcal{N}_j} p(u_k | \Phi(v_j))$$

Image Credit: [1]

Deep Walks

Model the probability of context (random walks) given its word (vertex)

Construct probability using Softmax:

$$p(u_k = i|\Phi(v_j)) = \frac{\exp(w_i^T \Phi(v_j))}{\sum_{m=1}^{|V|} \exp(w_m^T \Phi(v_j))}$$

Image Credit: [1]

Deep Walks

Model the probability of context (random walks) given its word (vertex)

Construct probability using Softmax:

$$p(u_k = i|\Phi(v_j)) = \frac{\exp(w_i^T \Phi(v_j))}{\sum_{m=1}^{|V|} \exp(w_m^T \Phi(v_j))}$$

One weight per vertex, i.e., a huge softmax on large graphs

Can we improve the efficiency?

Image Credit: [1]

Deep Walks

Hierarchical Softmax

Build a binary tree over vocabulary (the set of all vertices)

$$p(u_k \mid \Phi(v_j)) = \prod_{l=1}^{\lceil \log |V| \rceil} p(b_l \mid \Phi(v_j))$$

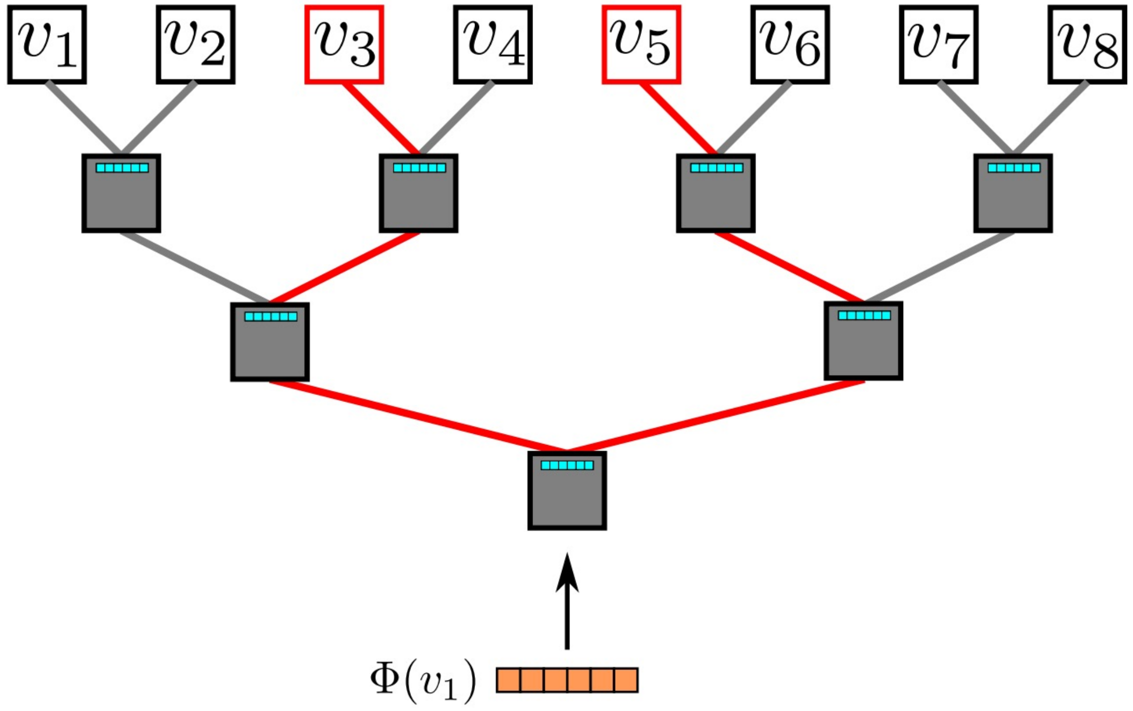


Image Credit: [1]

Deep Walks

Multi-Label Classification (BlogCatalog)

Two-stage pipeline: 1) learn node embeddings unsupervisedly; 2) learn node classifier supervisedly

	% Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1(%)	DeepWalk	**36.00**	**38.20**	**39.60**	**40.30**	**41.00**	**41.30**	41.50	41.50	42.00
	SpectralClustering	31.06	34.95	37.27	38.93	39.97	40.99	**41.66**	**42.42**	**42.62**
	EdgeCluster	27.94	30.76	31.85	32.99	34.12	35.00	34.63	35.99	36.29
	Modularity	27.35	30.74	31.77	32.97	34.09	36.13	36.08	37.23	38.18
	wvRN	19.51	24.34	25.62	28.82	30.37	31.81	32.19	33.33	34.28
	Majority	16.51	16.66	16.61	16.70	16.91	16.99	16.92	16.49	17.26
Macro-F1(%)	DeepWalk	**21.30**	**23.80**	25.30	26.30	27.30	27.60	27.90	28.20	28.90
	SpectralClustering	19.14	23.57	**25.97**	**27.46**	**28.31**	**29.46**	**30.13**	**31.38**	**31.78**
	EdgeCluster	16.16	19.16	20.48	22.00	23.00	23.64	23.82	24.61	24.92
	Modularity	17.36	20.00	20.80	21.85	22.65	23.41	23.89	24.20	24.97
	wvRN	6.25	10.13	11.64	14.24	15.86	17.18	17.98	18.86	19.57
	Majority	2.52	2.55	2.52	2.58	2.58	2.63	2.61	2.48	2.62

Image Credit: [1]

Deep Graph InfoMax

Mutual Information between X and Y:

$$\begin{aligned} \mathrm{MI}(X, Y) &= \mathrm{KL}(p(X, Y) \| p(X)p(Y)) \\ &= \int\int p(X, Y) \log \frac{p(X, Y)}{p(X)p(Y)} dX dY \end{aligned}$$

Deep Graph InfoMax

Mutual Information between X and Y:

$$
\begin{aligned}
\mathrm{MI}(X, Y) &= \mathrm{KL}(p(X, Y) \| p(X)p(Y)) \\
&= \int\int p(X, Y) \log \frac{p(X, Y)}{p(X)p(Y)} dX dY
\end{aligned}
$$

It quantifies the "amount of information" obtained about one random variable by observing the other random variable

The higher the mutual information => knowing one would give you more information about the other

Deep Graph InfoMax

How to estimate Mutual Information?

$$
\begin{aligned}
\mathrm{MI}(X, Y) &= \mathrm{KL}(p(X, Y) \| p(X)p(Y)) \\
&= \int\int p(X, Y) \log \frac{p(X, Y)}{p(X)p(Y)} dX dY \\
&= \int\int p(X, Y) \log \frac{p(Y|X)}{p(Y)} dX dY \\
&= \mathbb{E}_{p(X,Y)} \left[\log \frac{p(Y|X)}{p(Y)} \right]
\end{aligned}
$$

Deep Graph InfoMax

How to estimate Mutual Information?

Suppose we generate Y from a mixture distribution:

$$q(C=1) = q(C=0) = \frac{1}{2}$$
$$p(Y|X) = q(Y|C=1)$$
$$p(Y) = q(Y|C=0)$$

Deep Graph InfoMax

How to estimate Mutual Information?

Suppose we generate Y from a mixture distribution:

$$q(C=1) = q(C=0) = \frac{1}{2}$$
$$p(Y|X) = q(Y|C=1)$$
$$p(Y) = q(Y|C=0)$$

$$q(Y) = q(Y|C=0)q(C=0) + q(Y|C=1)q(C=1)$$

Deep Graph InfoMax

How to estimate Mutual Information?

Suppose we generate Y from a mixture distribution:

$$q(C=1) = q(C=0) = \frac{1}{2}$$
$$p(Y|X) = q(Y|C=1)$$
$$p(Y) = q(Y|C=0)$$

$$q(Y) = q(Y|C=0)q(C=0) + q(Y|C=1)q(C=1)$$

Density Ratio Trick:

$$\frac{p(Y|X)}{p(Y)} = \frac{q(Y|C=1)}{q(Y|C=0)} = \frac{\frac{q(C=1|Y)q(Y)}{q(C=1)}}{\frac{q(C=0|Y)q(Y)}{q(C=0)}} = \frac{q(C=1|Y)}{q(C=0|Y)}$$

Deep Graph InfoMax

How to estimate Mutual Information?

Suppose we generate Y from a mixture distribution:

$$q(C=1) = q(C=0) = \frac{1}{2}$$
$$p(Y|X) = q(Y|C=1)$$
$$p(Y) = q(Y|C=0)$$

$$q(Y) = q(Y|C=0)q(C=0) + q(Y|C=1)q(C=1)$$

Density Ratio Trick:

$$\frac{p(Y|X)}{p(Y)} = \frac{q(Y|C=1)}{q(Y|C=0)} = \frac{\frac{q(C=1|Y)q(Y)}{q(C=1)}}{\frac{q(C=0|Y)q(Y)}{q(C=0)}} = \frac{\boxed{q(C=1|Y)}}{q(C=0|Y)}$$

Binary Classifier!

Deep Graph InfoMax

How to estimate Mutual Information?

Suppose we generate Y from a mixture distribution:

$$q(C=1) = q(C=0) = \frac{1}{2}$$
$$p(Y|X) = q(Y|C=1)$$
$$p(Y) = q(Y|C=0)$$

$$q(Y) = q(Y|C=0)q(C=0) + q(Y|C=1)q(C=1)$$

Density Ratio Trick:

$$\frac{p(Y|X)}{p(Y)} = \frac{q(Y|C=1)}{q(Y|C=0)} = \frac{\frac{q(C=1|Y)q(Y)}{q(C=1)}}{\frac{q(C=0|Y)q(Y)}{q(C=0)}} = \frac{\boxed{q(C=1|Y)}}{q(C=0|Y)}$$

Binary Classifier!

We only need samplers to train a classifier, no exact probability densities are required!

Deep Graph InfoMax

Given samples $Y, C \sim q(Y, C)$

$$q(C=1) = q(C=0) = \frac{1}{2}$$

$$p(Y|X) = q(Y|C=1)$$

$$p(Y) = q(Y|C=0)$$

We build a classifier $q(C=1|Y) \propto f_\theta(X, Y)$

Deep Graph InfoMax

Given samples $Y, C \sim q(Y, C)$

$$q(C=1) = q(C=0) = \frac{1}{2}$$

$$p(Y|X) = q(Y|C=1)$$

$$p(Y) = q(Y|C=0)$$

We build a classifier $q(C=1|Y) \propto f_\theta(X, Y)$

Minimize Binary Cross-entropy: $$\min_q \quad -\frac{1}{N}\left(\sum_{Y,C} C \log q(C=1|Y) + (1-C)\log(1-q(C=1|Y))\right)$$

Deep Graph InfoMax

Given samples $Y, C \sim q(Y, C)$

$$q(C=1) = q(C=0) = \frac{1}{2}$$
$$p(Y|X) = q(Y|C=1)$$
$$p(Y) = q(Y|C=0)$$

We build a classifier $q(C=1|Y) \propto f_\theta(X, Y)$

Minimize Binary Cross-entropy: $$\min_q \quad -\frac{1}{N}\left(\sum_{Y,C} C \log q(C=1|Y) + (1-C)\log(1-q(C=1|Y))\right)$$

The optimal q is: $$q^*(C=1|Y) = \frac{\frac{p(Y|X)}{p(Y)}}{1+\frac{p(Y|X)}{p(Y)}}$$

Optimal Bayesian Classifier!

Deep Graph InfoMax

The optimal q is $$q^*(C=1|Y) = \frac{\frac{p(Y|X)}{p(Y)}}{1+\frac{p(Y|X)}{p(Y)}}$$

Binary Cross-entropy: $$\mathcal{L}_{\mathrm{BCE}}(q) = -\mathbb{E}_{C,Y}\left[C\log q(C=1|Y) + (1-C)\log(1-q(C=1|Y))\right]$$

Deep Graph InfoMax

The optimal q is $$q^*(C=1|Y) = \frac{\frac{p(Y|X)}{p(Y)}}{1+\frac{p(Y|X)}{p(Y)}}$$

Binary Cross-entropy: $$\mathcal{L}_{\mathrm{BCE}}(q) = -\mathbb{E}_{C,Y}\left[C \log q(C=1|Y) + (1-C)\log(1-q(C=1|Y))\right]$$

One can show [4] that $$\mathcal{L}_{\mathrm{BCE}}(q^*) \geq -\mathrm{MI}(X,Y)$$

One can also generalize this lower bound to multiple variables [4,5]

Deep Graph InfoMax

We want to learn node representations that capture global context information of the graph, i.e., maximize *local mutual information*

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

- Graph input

Deep Graph InfoMax

We want to learn node representations that capture global context information of the graph, i.e., maximize *local mutual information*

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

- Graph input
- Corrupted/Noisy graph input

Deep Graph InfoMax

We want to learn node representations that capture global context information of the graph, i.e., maximize *local mutual information*

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

- Graph input
- Corrupted/Noisy graph input
- Node representation

Deep Graph InfoMax

We want to learn node representations that capture global context information of the graph, i.e., maximize *local mutual information*

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

- Graph input
- Corrupted/Noisy graph input
- Node representation
- Graph representation

Deep Graph InfoMax

We want to learn node representations that capture global context information of the graph, i.e., maximize *local mutual information*

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

- Graph input
- Corrupted/Noisy graph input
- Node representation
- Graph representation
- Discriminator / Binary classifier

Deep Graph InfoMax

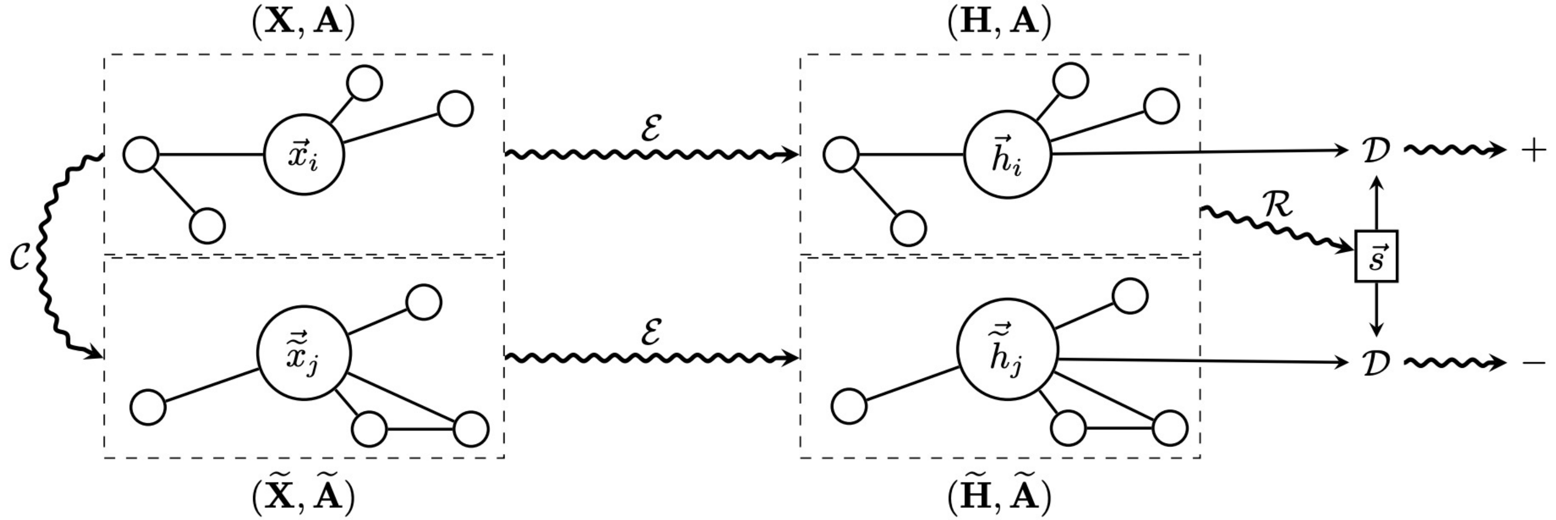
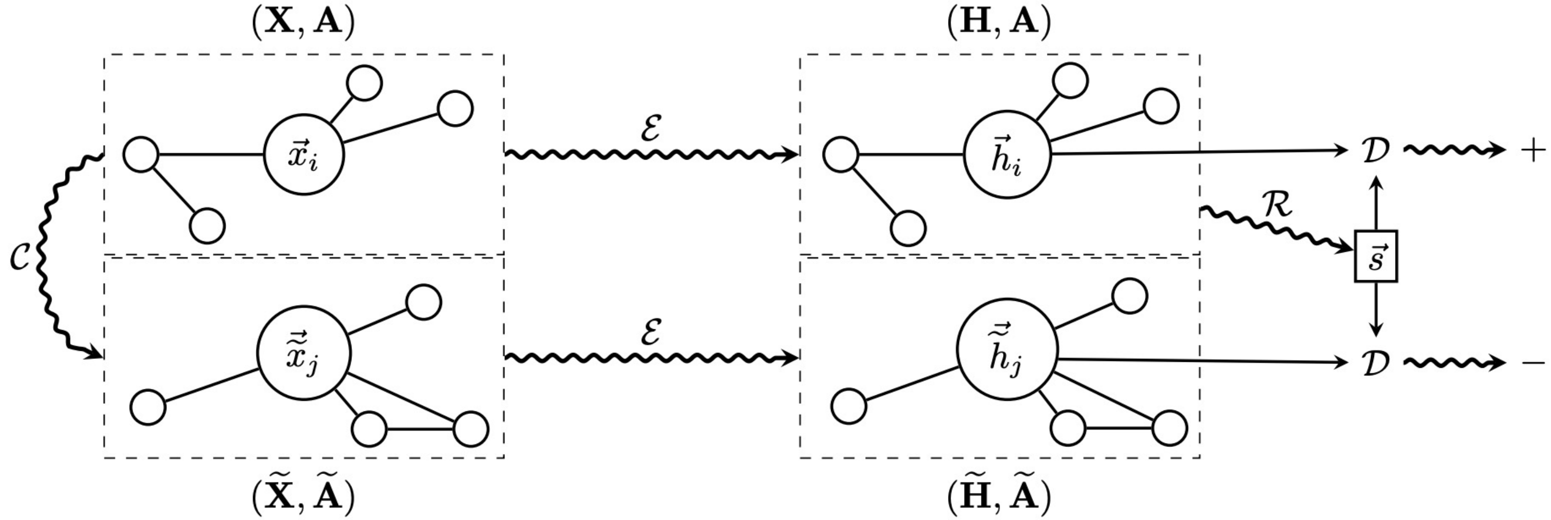


Image Credit: [3]

Deep Graph InfoMax



1. Sample a negative example by using the corruption function: $(\widetilde{\mathbf{X}}, \widetilde{\mathbf{A}}) \sim \mathcal{C}(\mathbf{X}, \mathbf{A})$.
2. Obtain patch representations, $\vec{h}_i$ for the input graph by passing it through the encoder: $\mathbf{H} = \mathcal{E}(\mathbf{X}, \mathbf{A}) = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}$.
3. Obtain patch representations, $\vec{\widetilde{h}}_j$ for the negative example by passing it through the encoder: $\widetilde{\mathbf{H}} = \mathcal{E}(\widetilde{\mathbf{X}}, \widetilde{\mathbf{A}}) = \{\vec{\widetilde{h}}_1, \vec{\widetilde{h}}_2, \ldots, \vec{\widetilde{h}}_M\}$.
4. Summarize the input graph by passing its patch representations through the readout function: $\vec{s} = \mathcal{R}(\mathbf{H})$.
5. Update parameters of $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{D}$ by applying gradient descent to maximize Equation 1.

Image Credit: [3]

Deep Graph InfoMax

If a huge graph is presented, sampling subgraphs is necessary

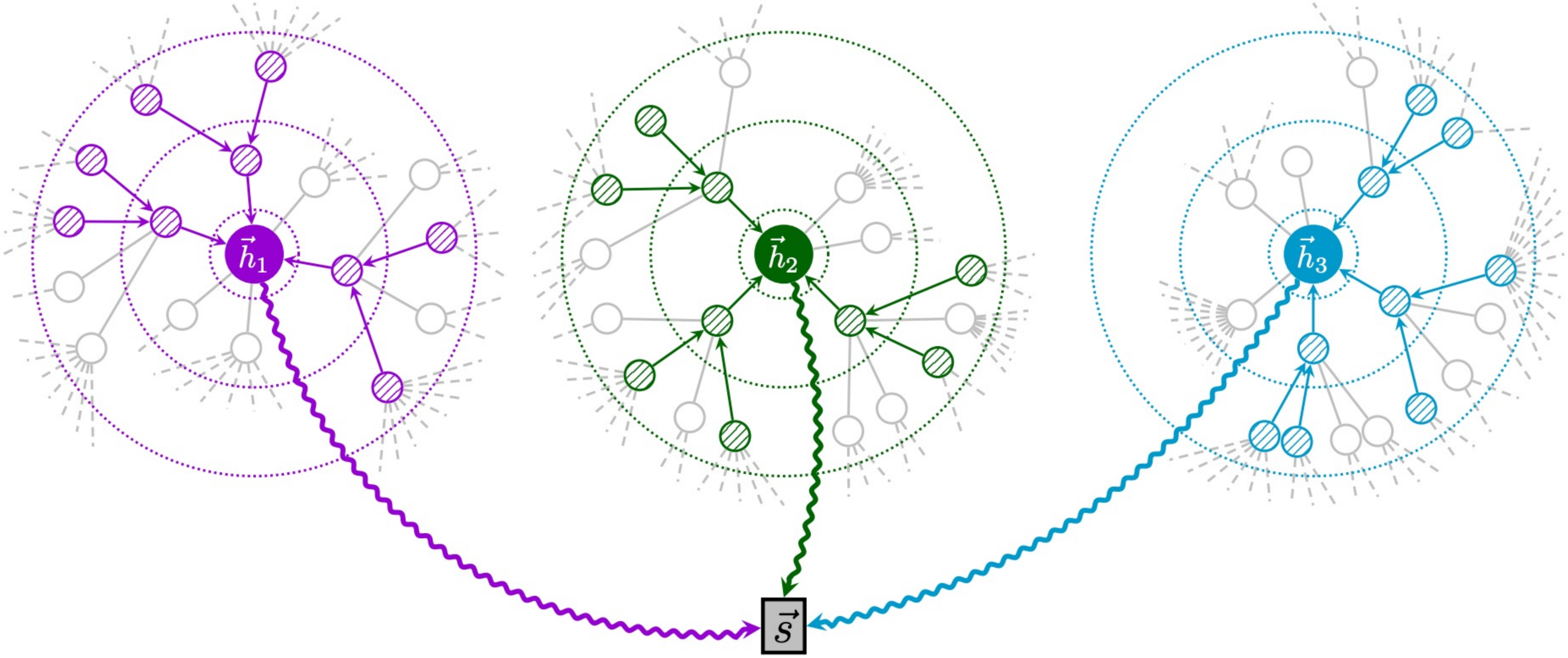


Image Credit: [3]

Deep Graph InfoMax

Transductive

Available data	Method	Cora	Citeseer	Pubmed
$\mathbf{X}$	Raw features	$47.9 \pm 0.4\%$	$49.3 \pm 0.2\%$	$69.1 \pm 0.3\%$
$\mathbf{A}, \mathbf{Y}$	LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
$\mathbf{A}$	DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
$\mathbf{X}, \mathbf{A}$	DeepWalk + features	$70.7 \pm 0.6\%$	$51.4 \pm 0.5\%$	$74.3 \pm 0.9\%$
$\mathbf{X}, \mathbf{A}$	Random-Init (ours)	$69.3 \pm 1.4\%$	$61.9 \pm 1.6\%$	$69.6 \pm 1.9\%$
$\mathbf{X}, \mathbf{A}$	**DGI** (ours)	$\mathbf{82.3} \pm 0.6\%$	$\mathbf{71.8} \pm 0.7\%$	$\mathbf{76.8} \pm 0.6\%$
$\mathbf{X}, \mathbf{A}, \mathbf{Y}$	GCN (Kipf & Welling, 2016a)	81.5%	70.3%	79.0%
$\mathbf{X}, \mathbf{A}, \mathbf{Y}$	Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%

Inductive

Available data	Method	Reddit	PPI
$\mathbf{X}$	Raw features	0.585	0.422
$\mathbf{A}$	DeepWalk (Perozzi et al., 2014)	0.324	—
$\mathbf{X}, \mathbf{A}$	DeepWalk + features	0.691	—
$\mathbf{X}, \mathbf{A}$	GraphSAGE-GCN (Hamilton et al., 2017a)	0.908	0.465
$\mathbf{X}, \mathbf{A}$	GraphSAGE-mean (Hamilton et al., 2017a)	0.897	0.486
$\mathbf{X}, \mathbf{A}$	GraphSAGE-LSTM (Hamilton et al., 2017a)	0.907	0.482
$\mathbf{X}, \mathbf{A}$	GraphSAGE-pool (Hamilton et al., 2017a)	0.892	0.502
$\mathbf{X}, \mathbf{A}$	Random-Init (ours)	0.933 ± 0.001	0.626 ± 0.002
$\mathbf{X}, \mathbf{A}$	**DGI** (ours)	$\mathbf{0.940} \pm 0.001$	$\mathbf{0.638} \pm 0.002$
$\mathbf{X}, \mathbf{A}, \mathbf{Y}$	FastGCN (Chen et al., 2018)	0.937	—
$\mathbf{X}, \mathbf{A}, \mathbf{Y}$	Avg. pooling (Zhang et al., 2018)	0.958 ± 0.001	0.969 ± 0.002

Image Credit: [3]

References

[1] Perozzi, B., Al-Rfou, R. and Skiena, S., 2014, August. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 701-710).

[2] Grover, A. and Leskovec, J., 2016, August. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 855-864).

[3] Veličković, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y. and Hjelm, R.D., 2018. Deep graph infomax. arXiv preprint arXiv:1809.10341.

[4] Van den Oord, A., Li, Y. and Vinyals, O., 2018. Representation learning with contrastive predictive coding. arXiv e-prints, pp.arXiv-1807.

[5] Hjelm, R.D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A. and Bengio, Y., 2018. Learning deep representations by mutual information estimation and maximization. arXiv preprint arXiv:1808.06670.

Questions?