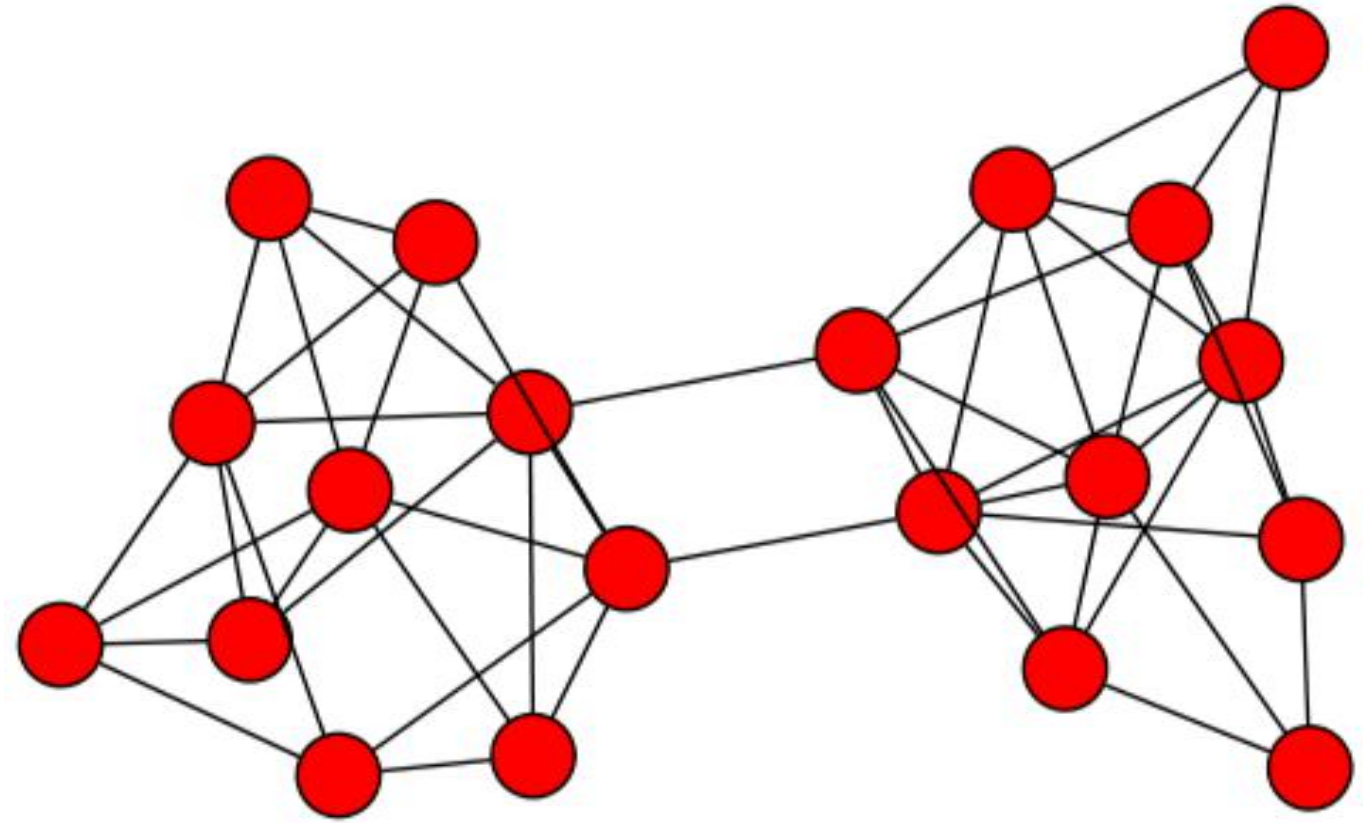
Frontiers in Network Embedding and GCN

Peng Cui

Tsinghua University

Network (Graph)

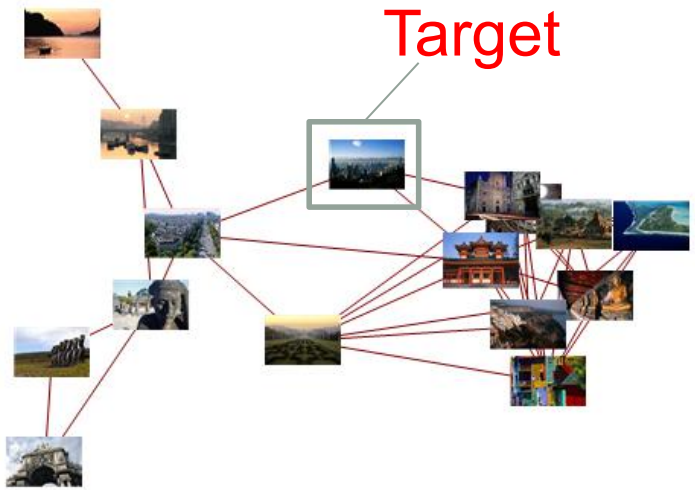
The general description of data and their relations



Why network is important?

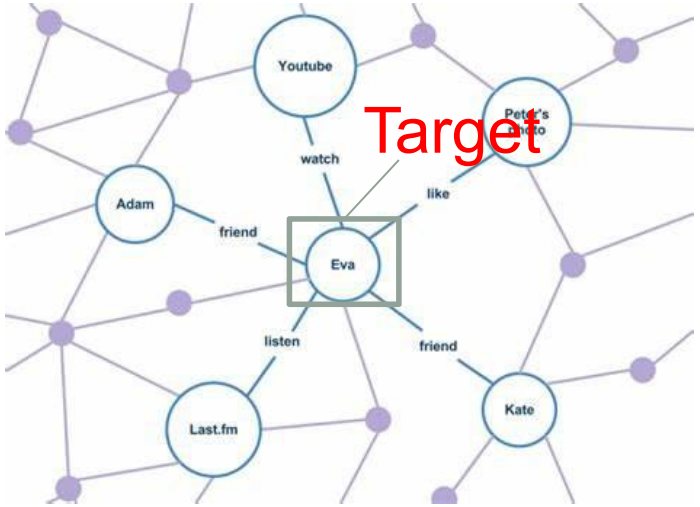
Can you name a case where you only care about an object but not its relations with other subjects?

Image Characterization



Reflected by relational subjects

Social Capital

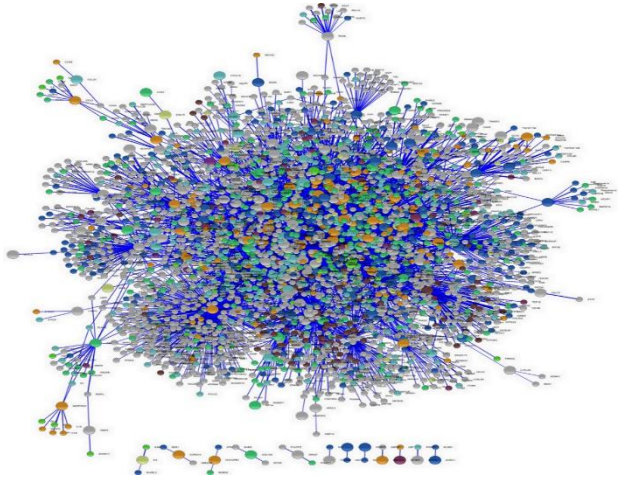


Decided by relational subjects

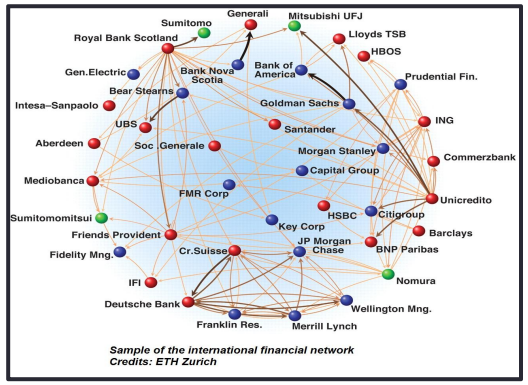
Graph/network data is everywhere

Social Networks

Biology Networks

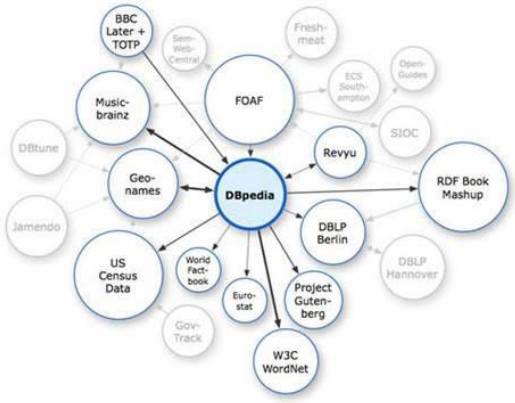


Finance Networks



Internet of Things

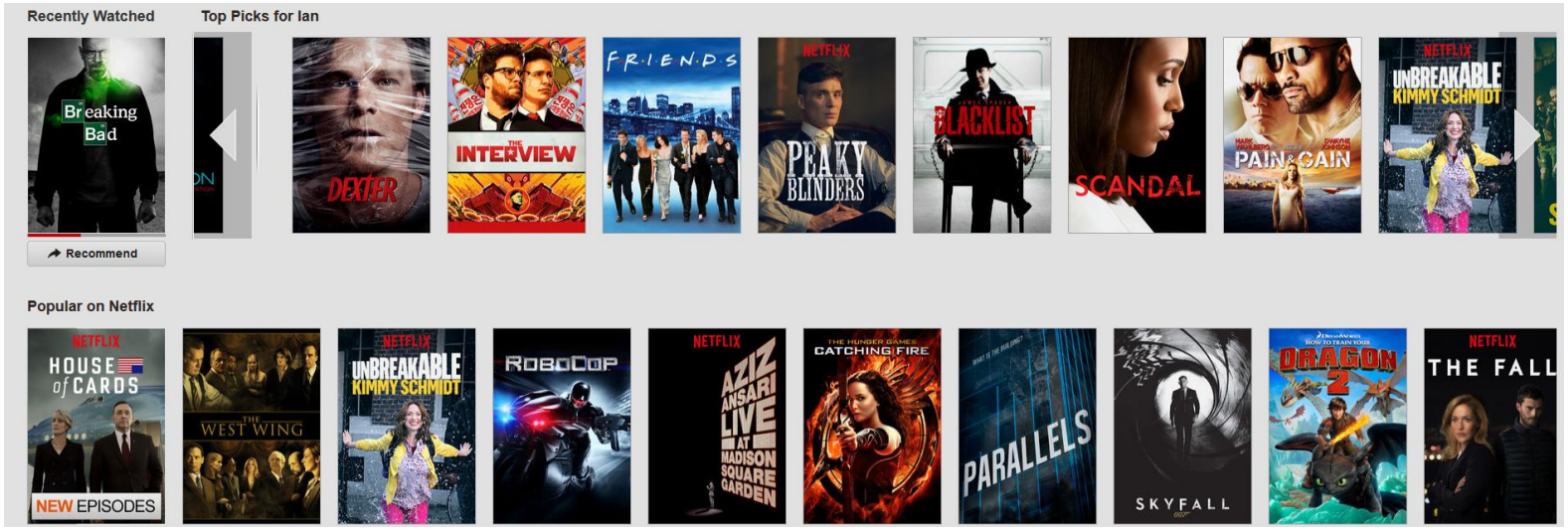
Information Networks



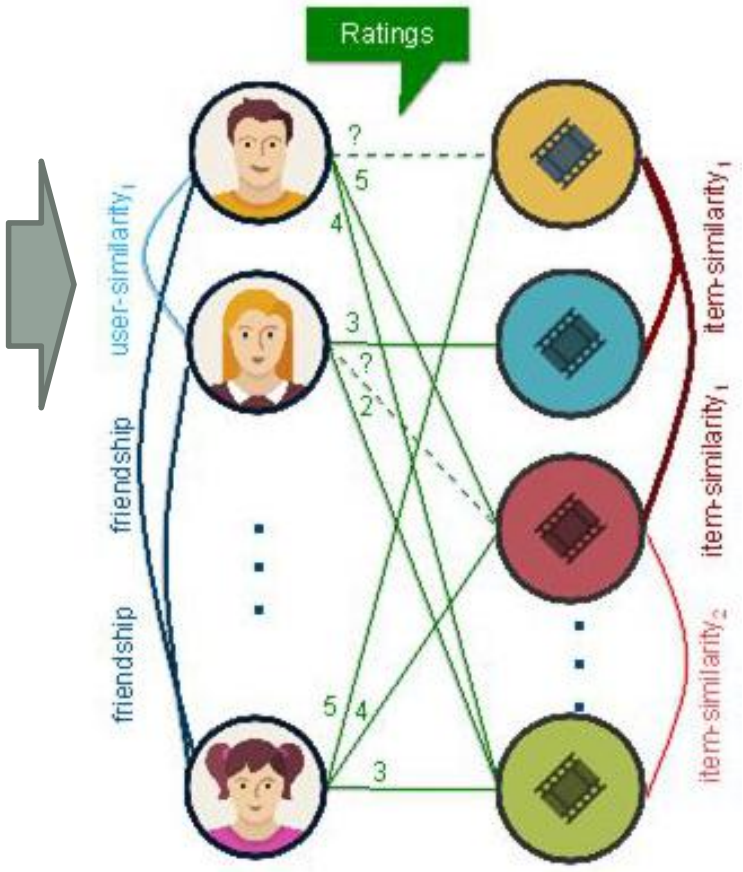
Logistic Networks

Many applications are intrinsically network problems

Recommendation Systems



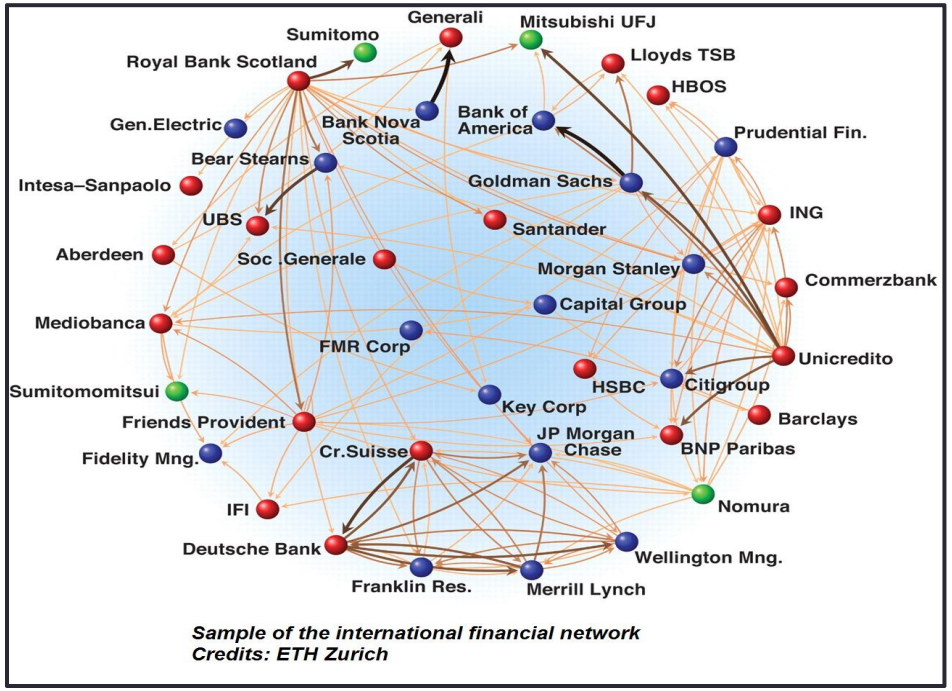
Link prediction in bipartite graphs



Many applications are intrinsically network problems

Financial credit & risk management

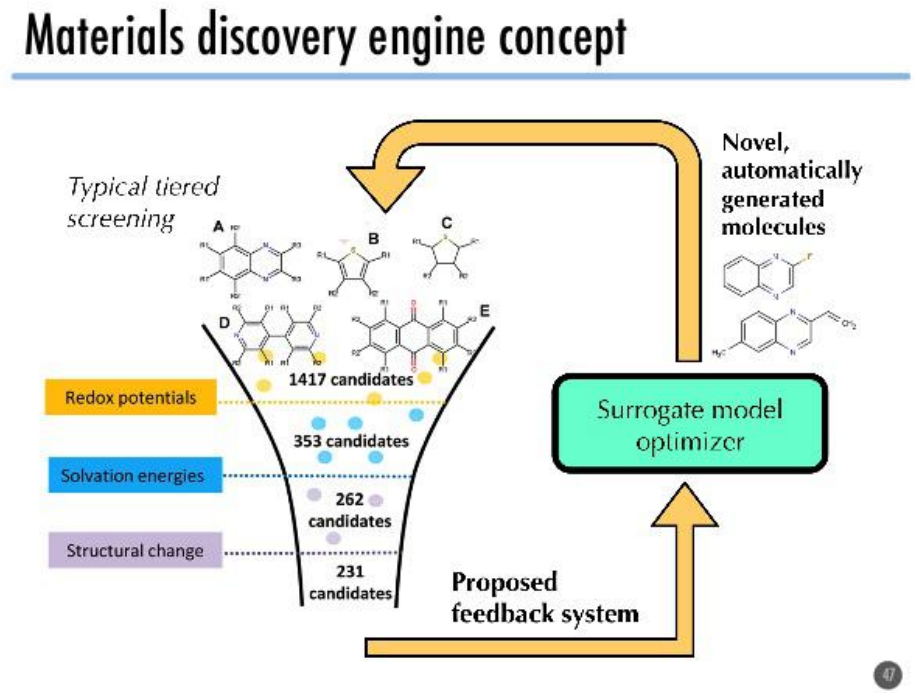
Node importance & classification



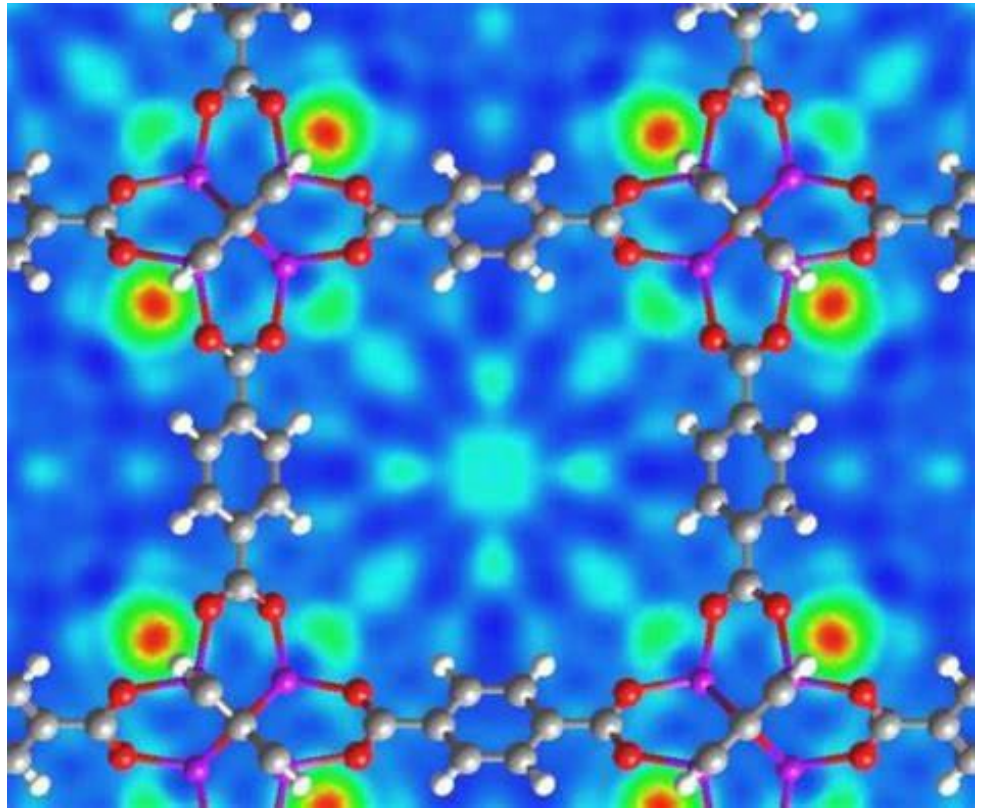
Sample of the international financial network
Credits: ETH Zurich

Many applications are intrinsically network problems

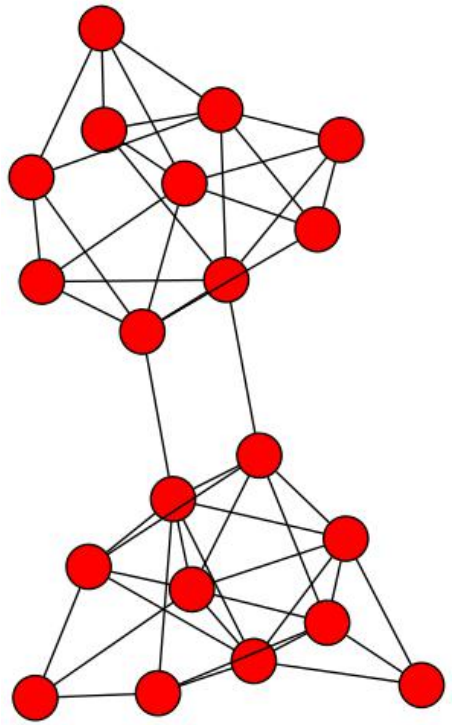
New material discovery



Subgraph pattern discovery



Traditional methods – graph theory



Graph Theory

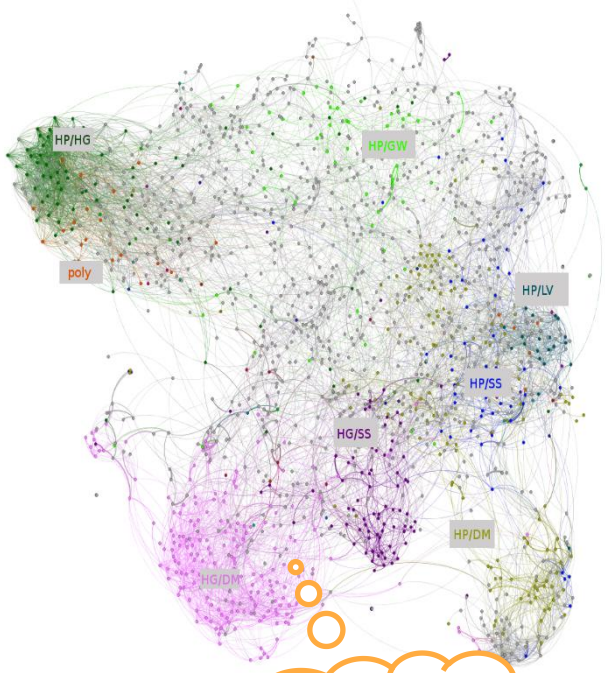
Centrality Problem

Isomorphism Problem

Routing Problem

…

Real networks

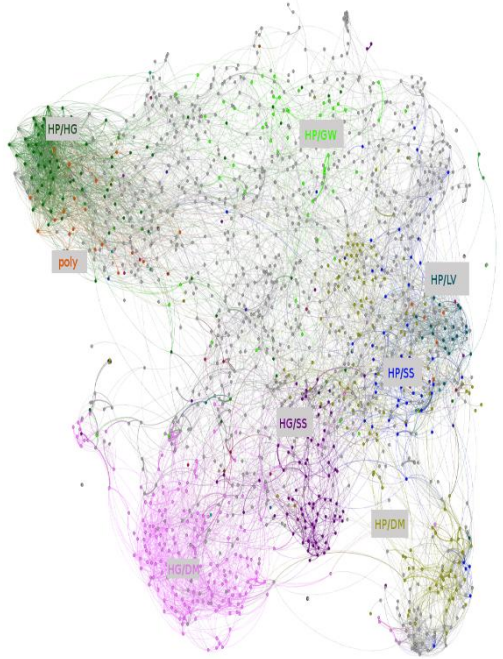


Billion nodes

Challenge 1: Scale

Traditional methods – graph analysis

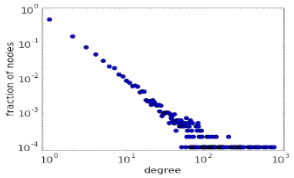
Real Networks



Graph Analysis

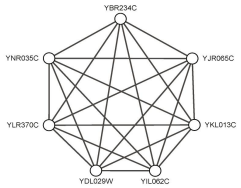
Graph Patterns

Power-law



Triadic Closure

Clustering effect



Applications

Link prediction

Community detection

Anomaly detection

…

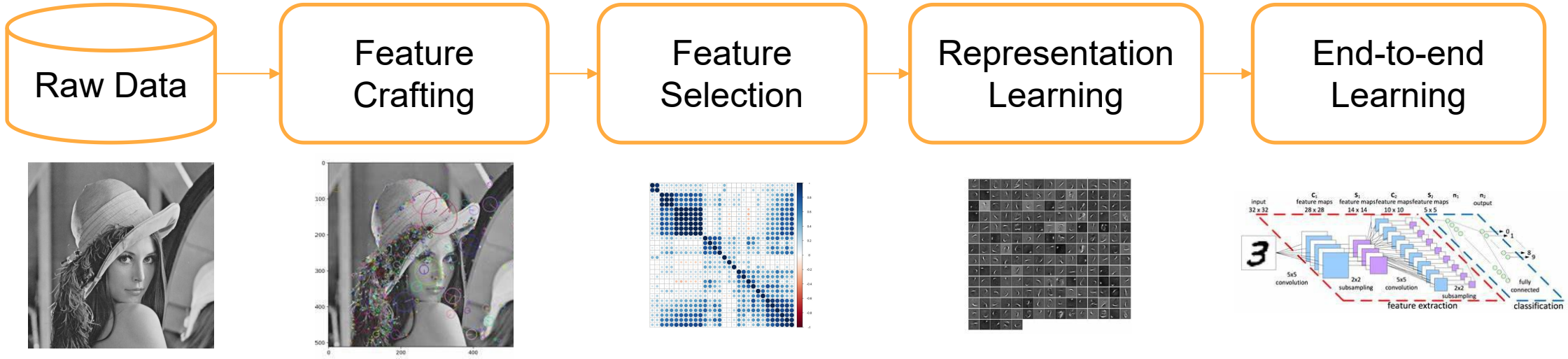
High Complexity

High Diversity

Challenge 2: Complexity and Diversity

From graph theory and analysis to learning

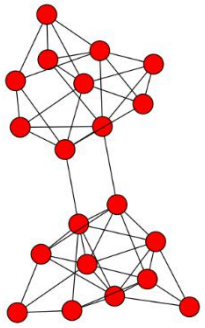
Progressive development of learning related fields

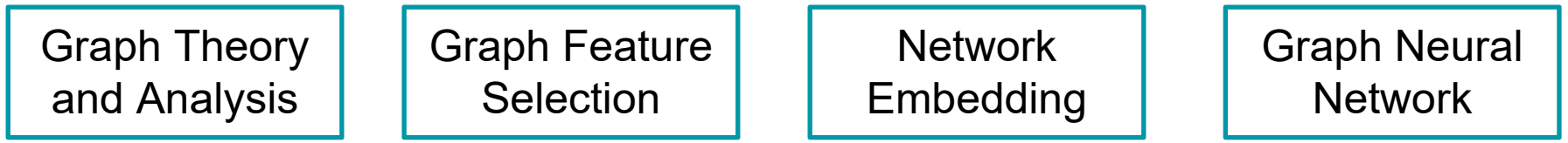


Go through a similar development path?

From graph theory and analysis to learning

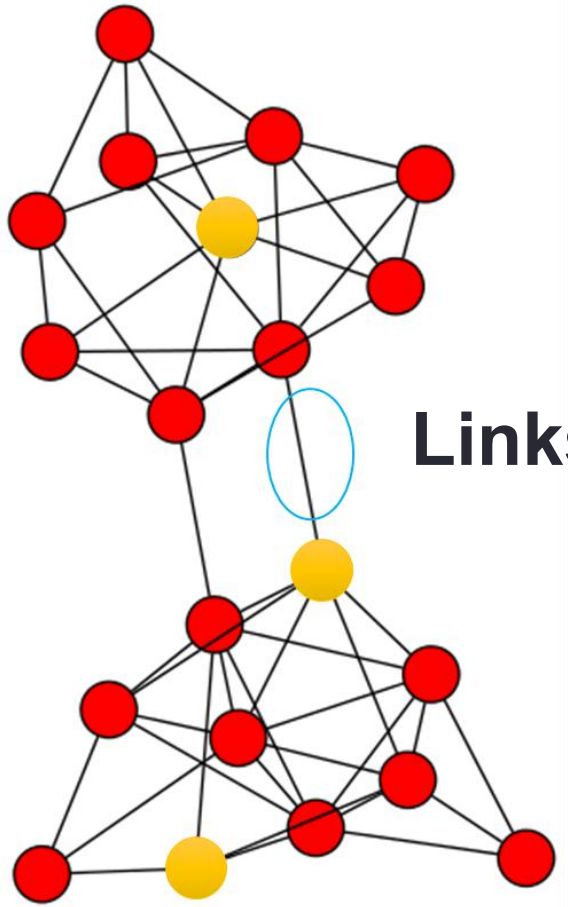
Provide general learning solutions to various tasks over a diverse range of complex networks.





Networks are not *learning-friendly*

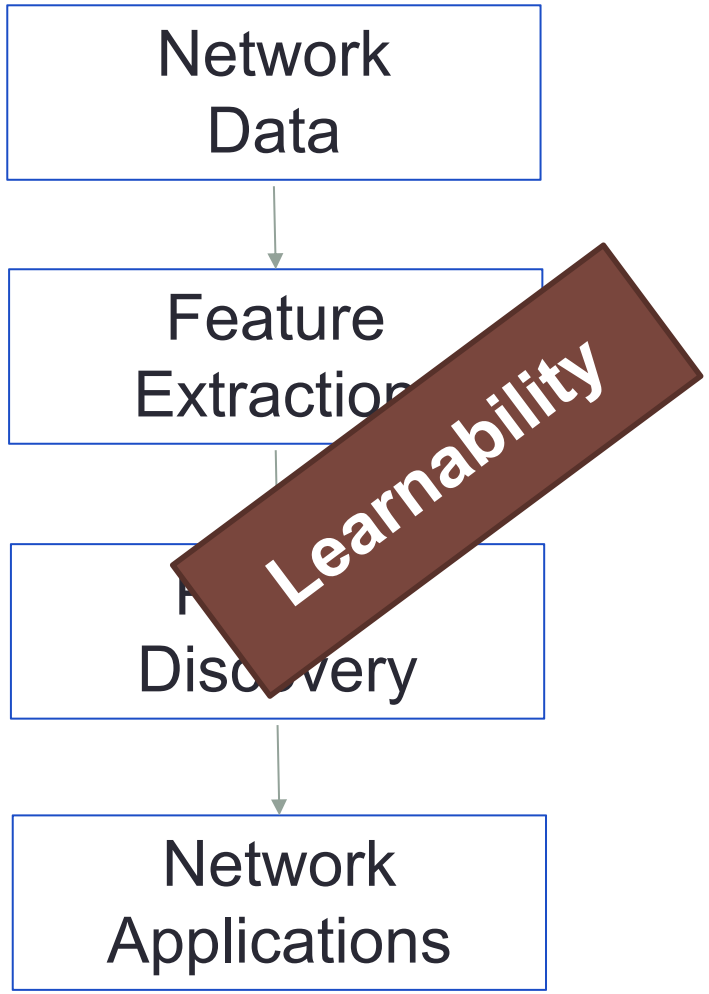
G = (V, E)



Links ⇨ **Topology**

Inapplicability of ML methods

Pipeline for network analysis



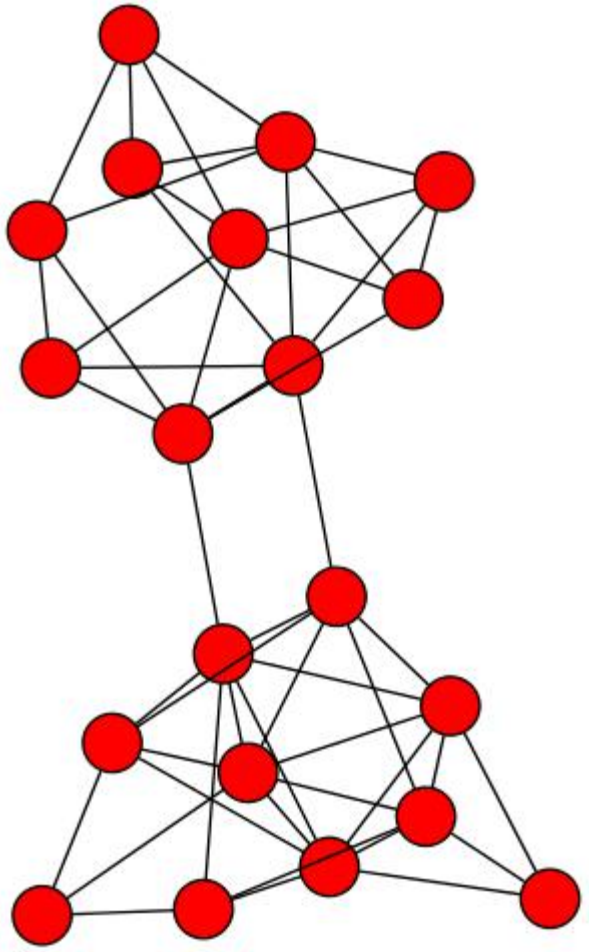
Learning from networks

Network Embedding

GCN

Network Embedding

G = (V, E)

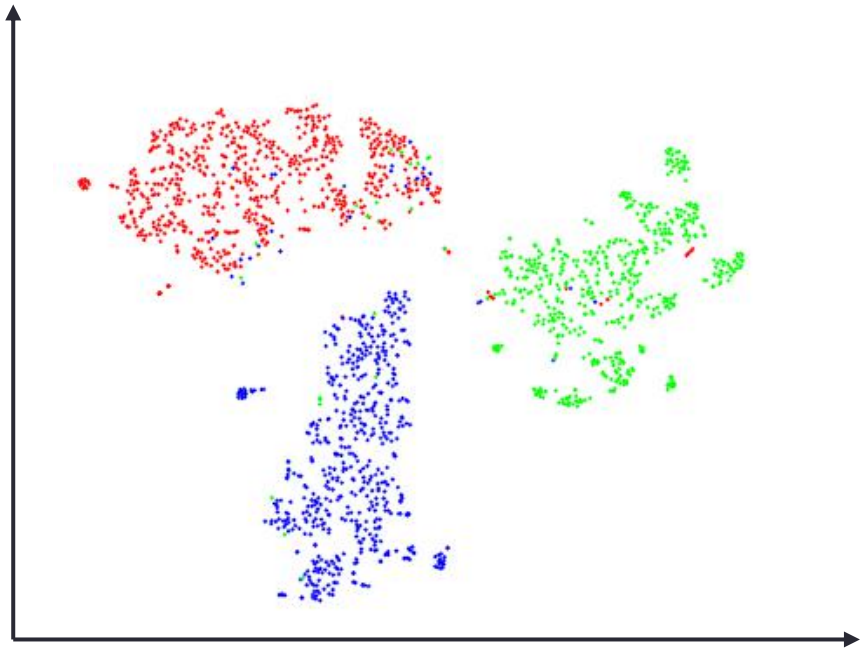


generate

embed

G = (V)

Vector Space



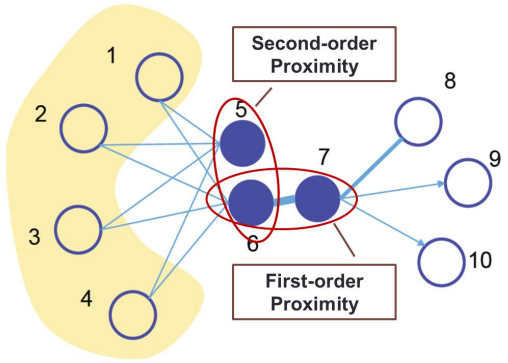
- Easy to parallel
- Can apply classical ML methods

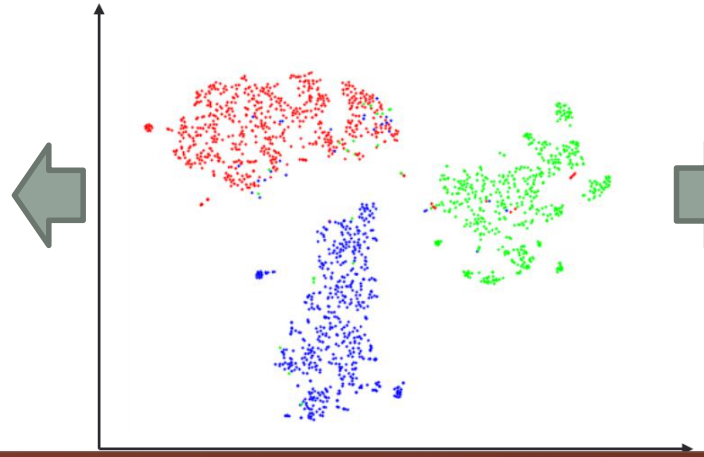
The goal of network embedding

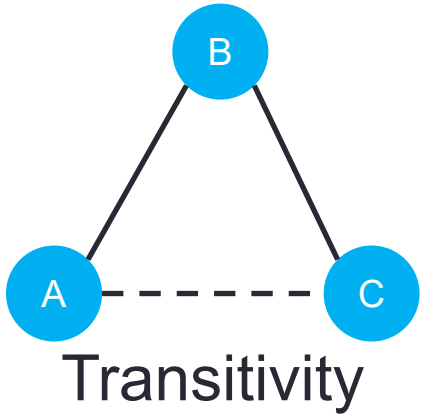
Goal Support network inference in vector space

Reflect network structure

Maintain network properties







Transform network nodes into vectors that are fit for off-the-shelf machine learning models.

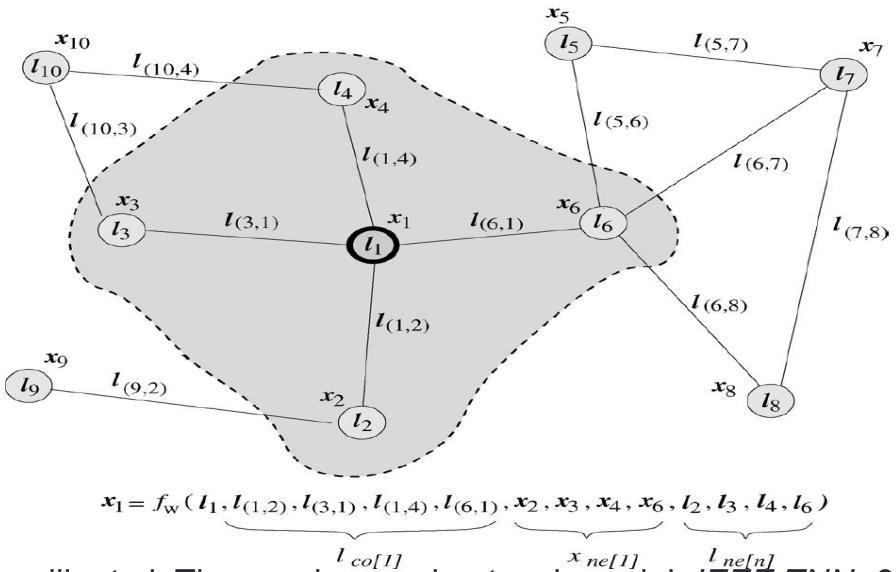
Graph Neural Networks

Design a learning mechanism on graph.

- Basic idea: recursive definition of states

$$\mathbf{s}_i = \sum_{j \in \mathcal{N}(i)} \mathcal{F}\left(\mathbf{s}_i, \mathbf{s}_j, \mathbf{F}_i^V, \mathbf{F}_j^V, \mathbf{F}_{i,j}^E\right)$$

- A simple example: PageRank



F. Scarselli, et al. The graph neural network model. *IEEE TNN, 2009.*

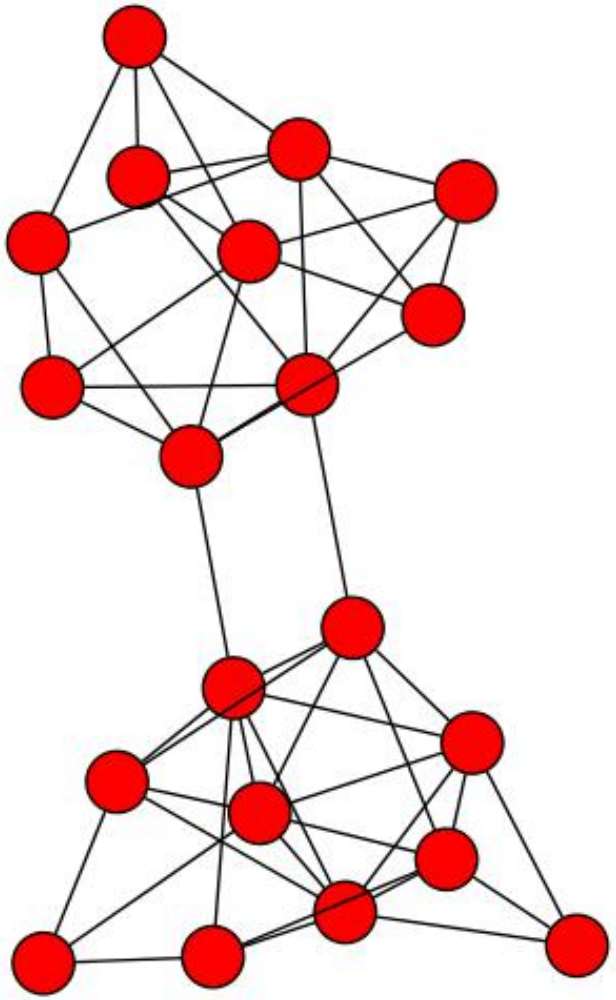
Network Embedding

- **Structure-preserved network embedding**
- **Property-preserved network embedding**
- **Dynamic network embedding**

Network Embedding

- **Structure-preserved network embedding**
- **Property-preserved network embedding**
- **Dynamic network embedding**

Network Structures



Nodes & Links

Pair-wise Proximity

Community Structures

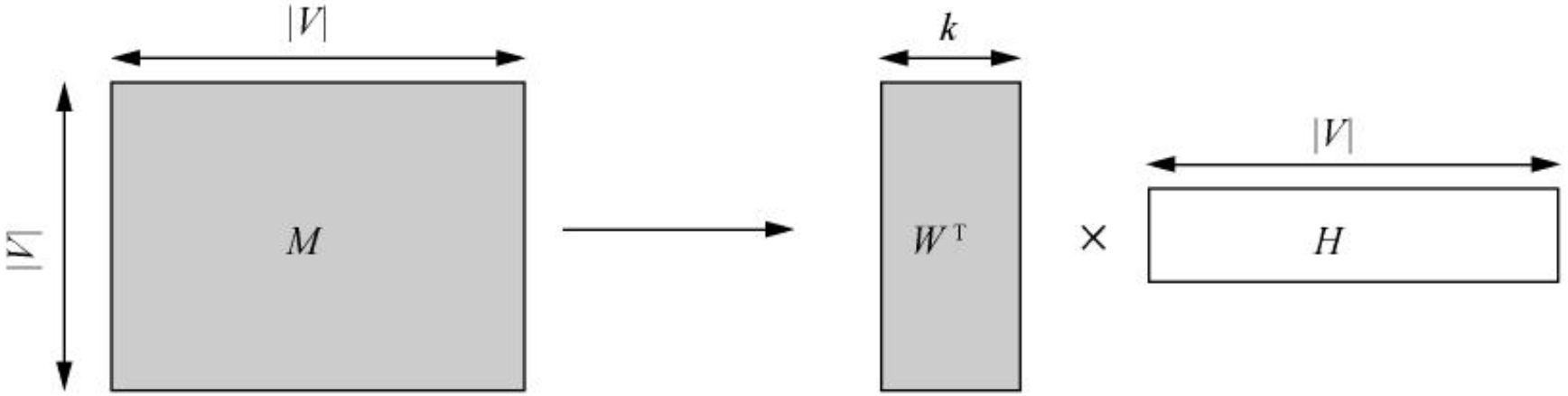
Hyper Edges

Global Structure

Nodes & Links

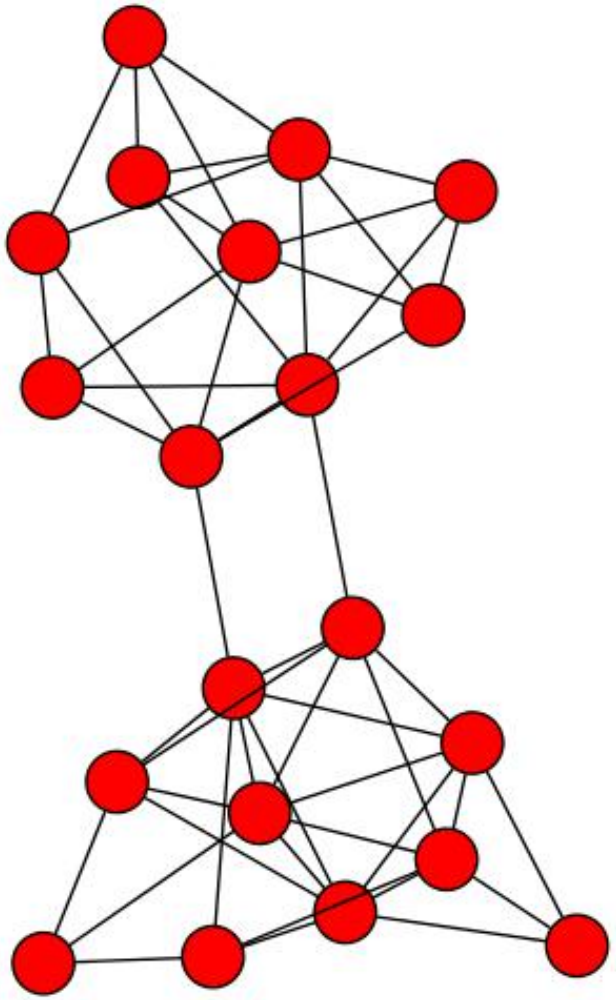
Reconstruct the original network

Matrix Factorization



Reconstruct all the links? May cause overfitting.
The network inference ability is seriously limited.

Network Structures



Nodes & Links

Pair-wise Proximity

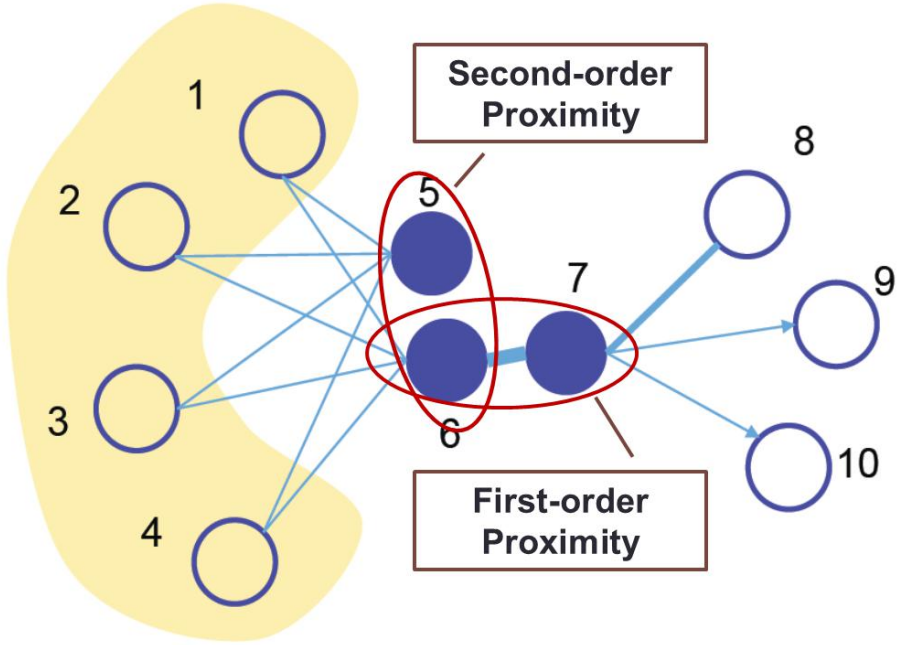
Community Structures

Hyper Edges

Global Structure

High-Order Proximity

- Capturing the underlying structure of networks



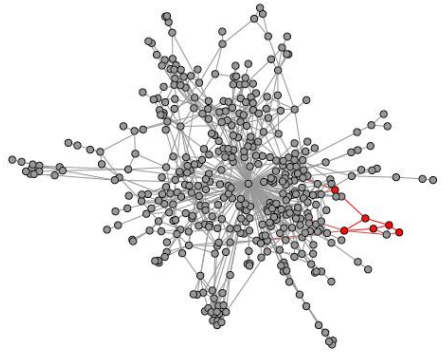
- Advantages:
  - Solve the sparsity problem of network connections
  - Measure indirect relationship between nodes

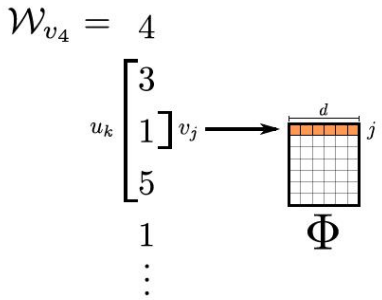
Deepwalk

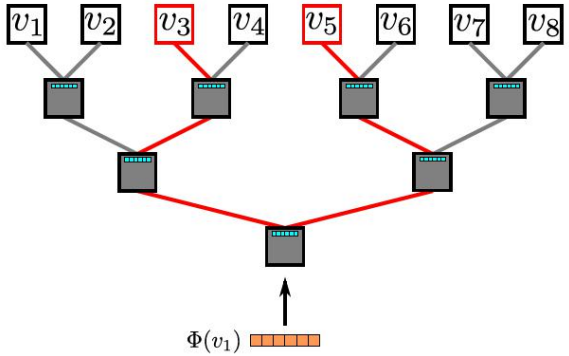
- Exploit truncated random walk to define neighborhood of a node.

Random Walks on Graph

- $V_{26} - V_{25} - V_{32} - V_3 - V_{10} \cdots$
- $V_5 - V_7 - V_{17} - V_6 - V_{11} \cdots$
- $V_{31} - V_{33} - V_{21} - V_{33} - V_{15}$



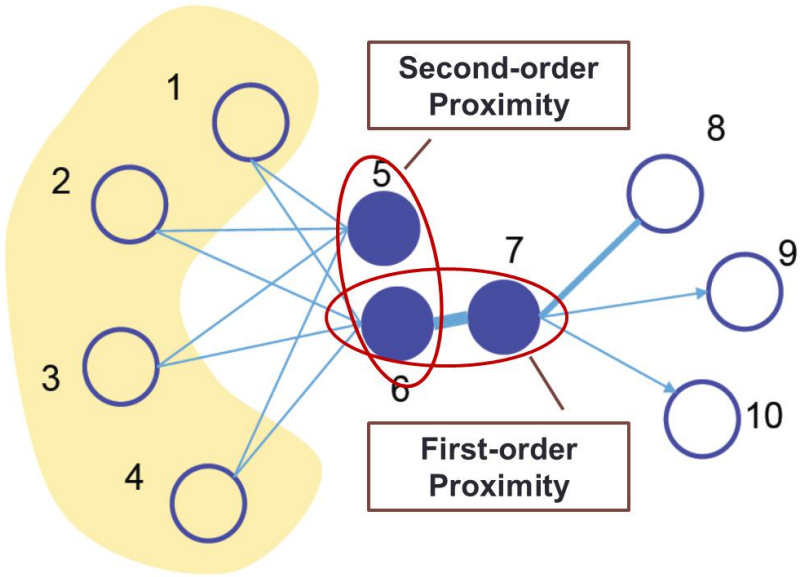




(a) Random walk generation. (b) Representation mapping. (c) Hierarchical Softmax.

B. Perozzi et al. Deepwalk: Online learning of social representations. *KDD 2014.*

LINE



LINE with First-order Proximity:
local pairwise

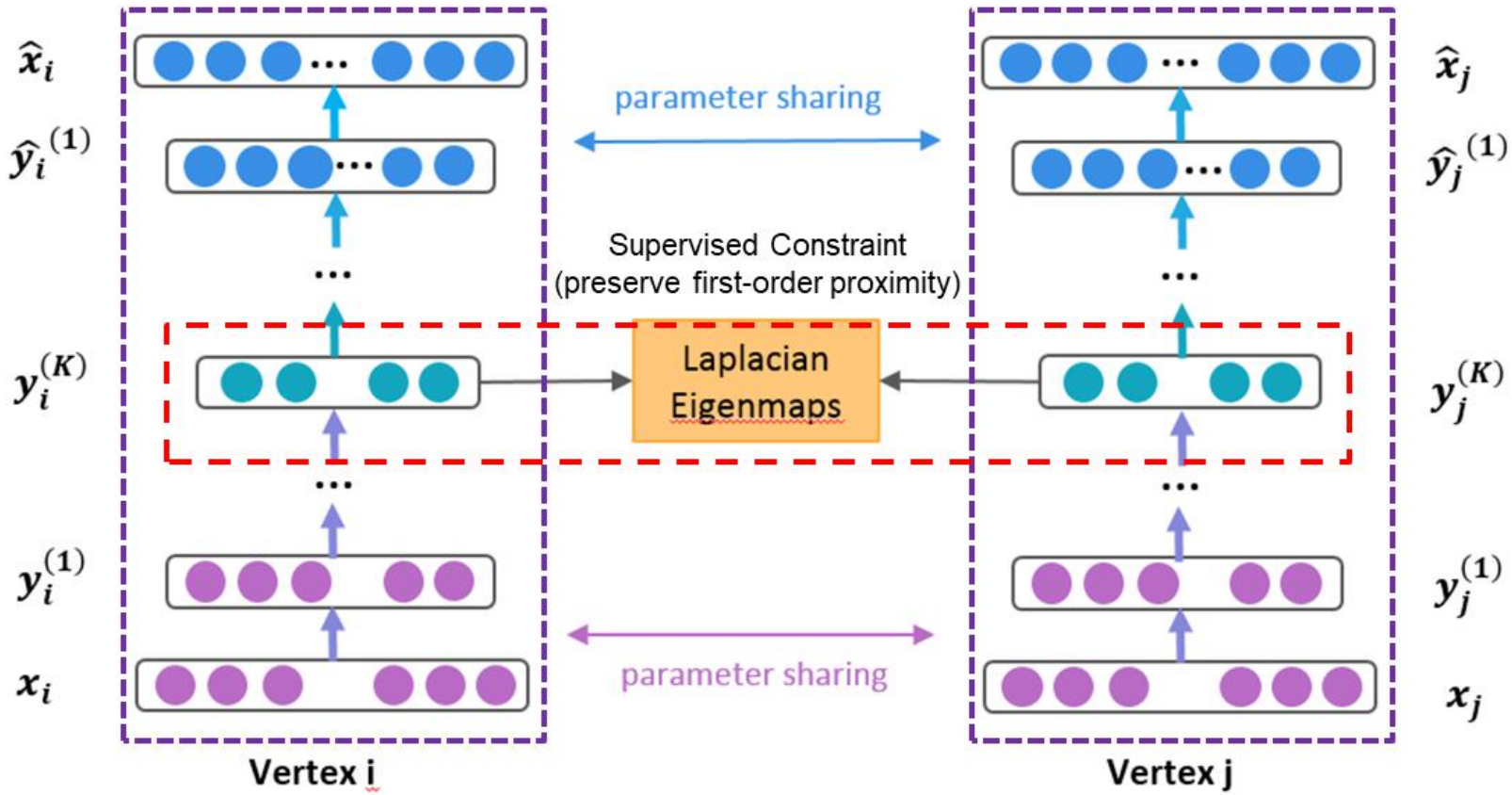
$$O_1 = -\sum_{(i,j)\in E} w_{ij} \log p_1(v_i, v_j)$$

LINE with Second-order Proximity:
neighborhood structures

$$O_2 = \sum_{i\in V} \lambda_i d(\hat{p}_2(\cdot|v_i), p_2(\cdot|v_i))$$

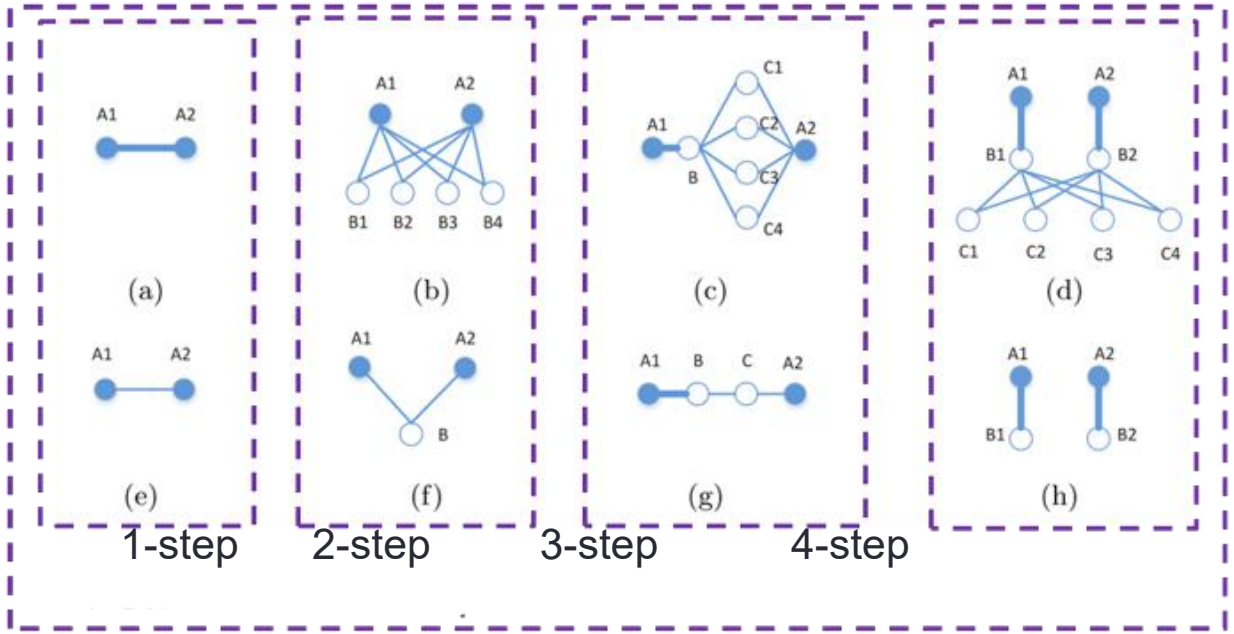
Jian Tang et al. LINE: Large-scale Information Network Embedding. *WWW 2015*.

SDNE – Structural Deep Network Embedding

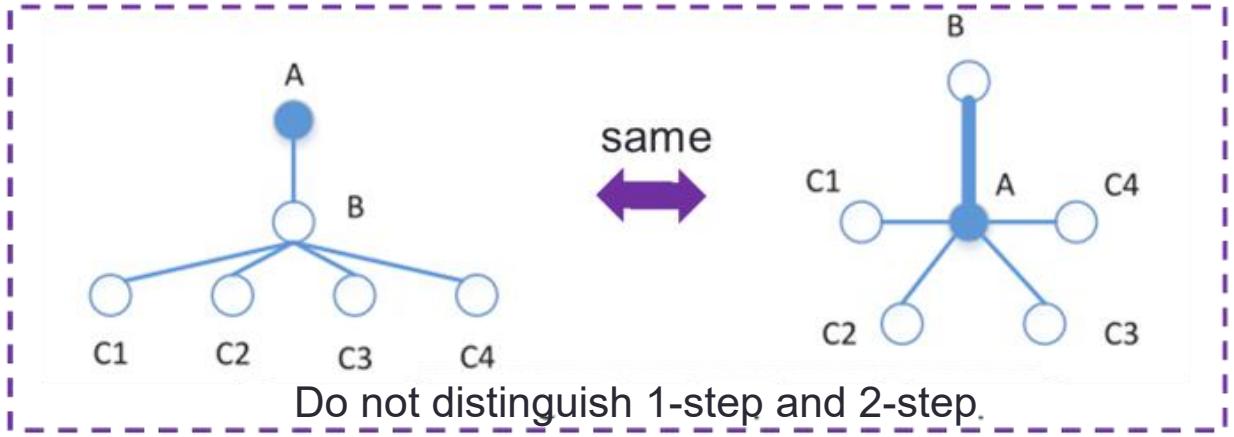


Daixin Wang et al. Structural Deep Network Embedding. *KDD*, 2016.

GraRep



capturing different k-step information

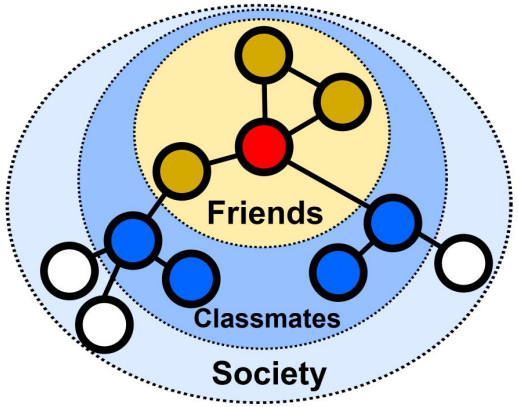


maintaining different k-step information

Shaosheng Cao et al. GraRep: Learning Graph Representations with Global Structural Information. *CIKM 2015*.

What is the *right* order?

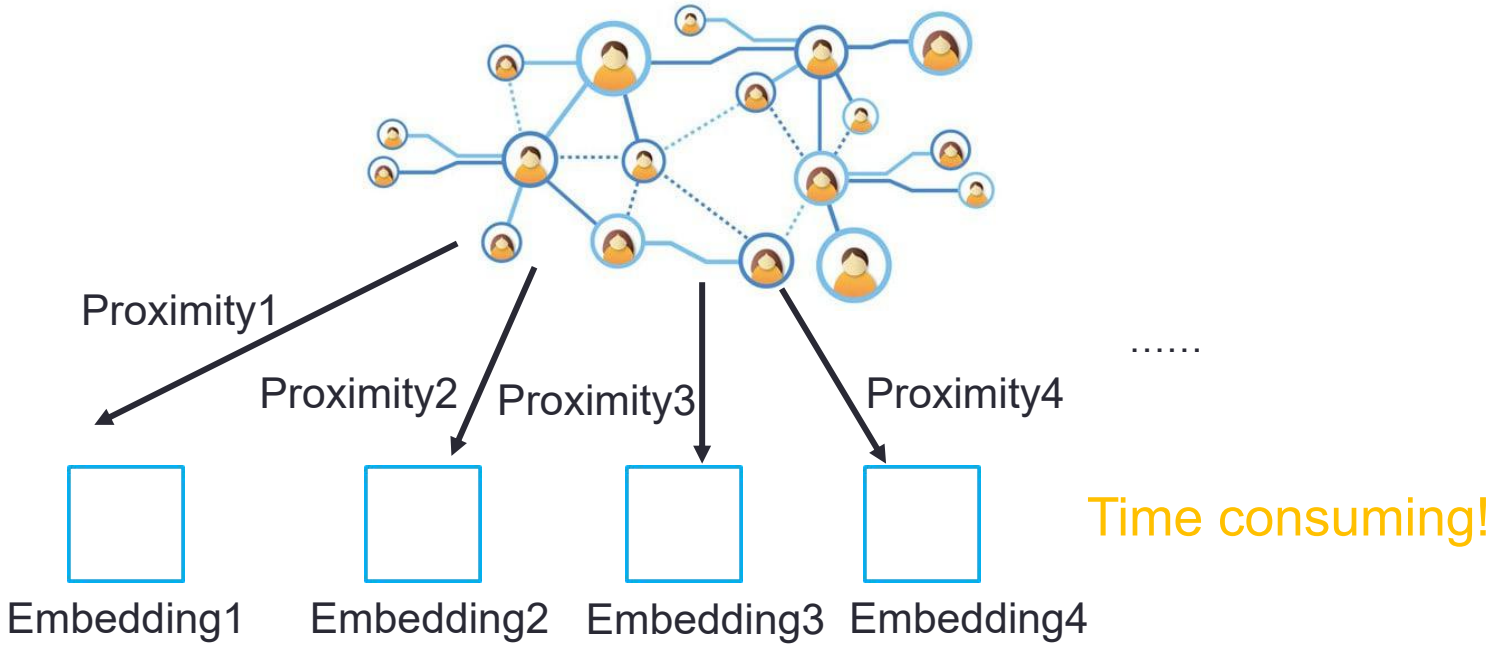
- Different networks/tasks require different high-order proximities
  - E.g., multi-scale classification (Bryan Perozzi, et al, 2017)



  - E.g., networks with different scales and sparsity
- Proximities of different orders can also be arbitrarily weighted
  - E.g., equal weights, exponentially decayed weights (Katz)

What is the *right* order?

- Existing methods can only preserve one fixed high-order proximity
  - Different high-order proximities are calculated separately



- -> How to preserve arbitrary-order proximity while guaranteeing accuracy and efficiency?

Problem Formulation

- High-order proximity: a polynomial function of the adjacency matrix

$$S = f(A) = w_1 A^1 + w_2 A^2 + \cdots + w_q A^q$$

  - q: order; $w_1 \ldots w_q$: weights, assuming to be non-negative
  - A: could be replaced by other variations (such as the Laplacian matrix)
- Objective function: matrix factorization

$$\min_{U^*, V^*} \left\| S - U^* {V^*}^T \right\|_F^2$$

  - $U^*, V^* \in \mathbb{R}^{N \times d}$: left/right embedding vectors
  - d: dimensionality of the space
- Optimal solution: Singular Value Decomposition (SVD)
  - $[U, \Sigma, V]$: top-d SVD results

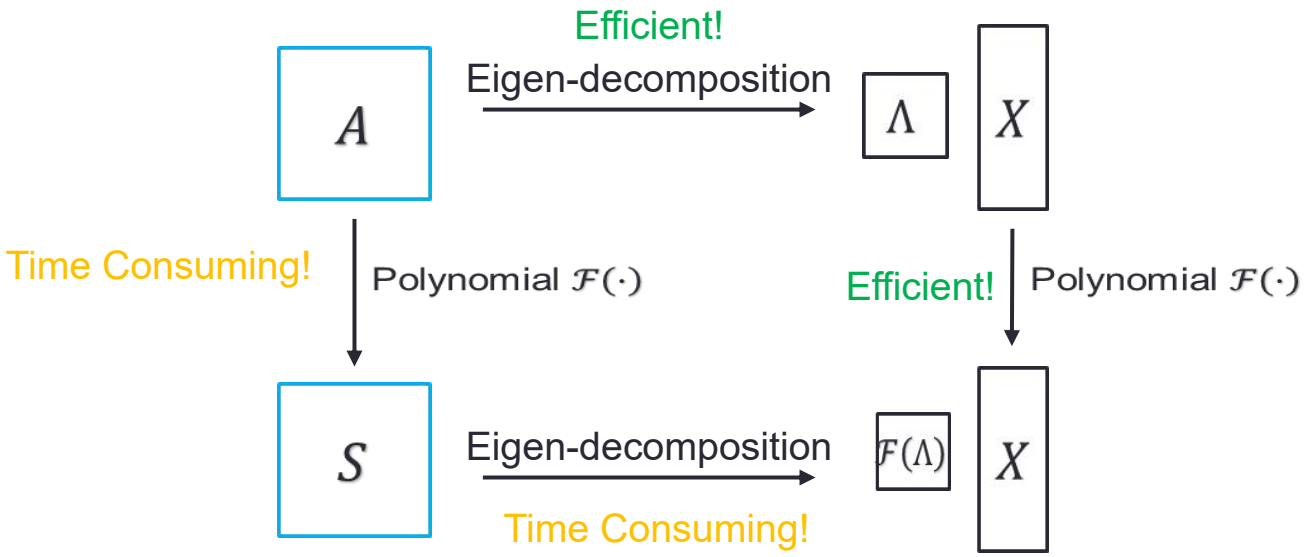
$$U^* = U\sqrt{\Sigma}, \qquad V^* = V\sqrt{\Sigma}$$

Z. Zhang, et al. Arbitrary-Order Proximity Preserved Network Embedding. *KDD, 2018.*

Eigen-decomposition Reweighting

- Eigen-decomposition reweighting

THEOREM 4.2 (EIGEN-DECOMPOSITION REWEIGHTING). *If* $[\lambda, \mathbf{x}]$ *is an eigen-pair of* $\mathbf{A}$, *then* $[\mathcal{F}(\lambda), \mathbf{x}]$ *is an eigen-pair of* $\mathbf{S} = \mathcal{F}(\mathbf{A})$.

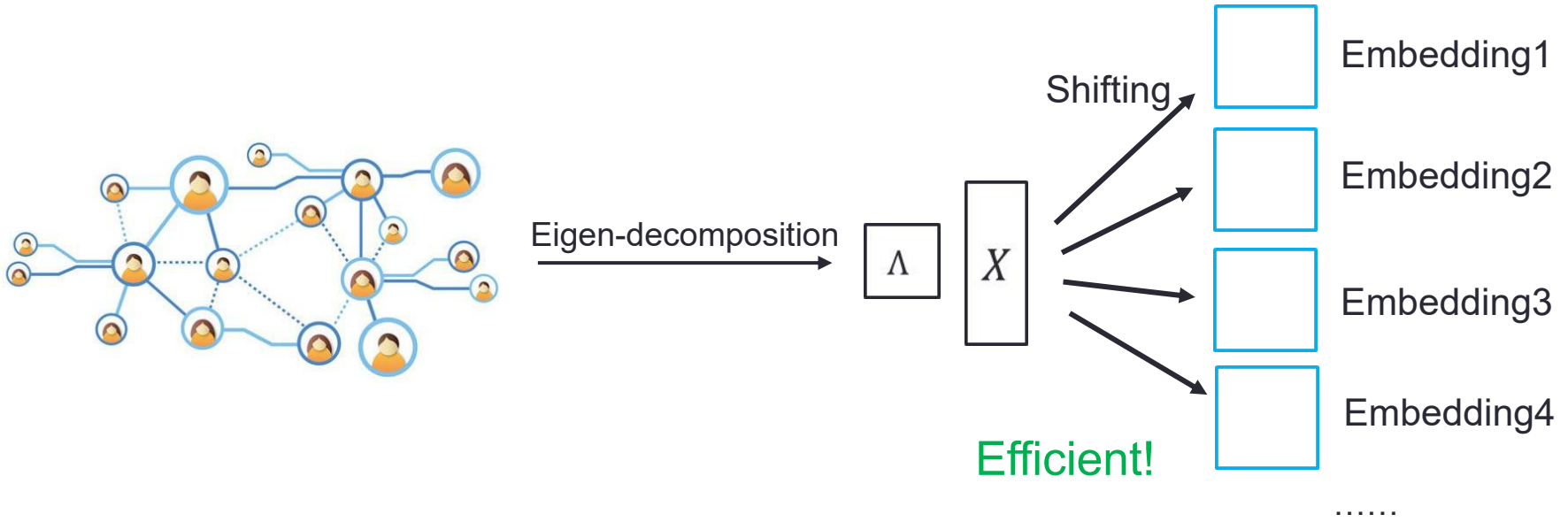


- **Insights**: high-order proximity is simply re-weighting dimensions!

Z. Zhang, et al. Arbitrary-Order Proximity Preserved Network Embedding. *KDD, 2018.*

Preserving Arbitrary-Order Proximity

- Shifting across different orders/weights:

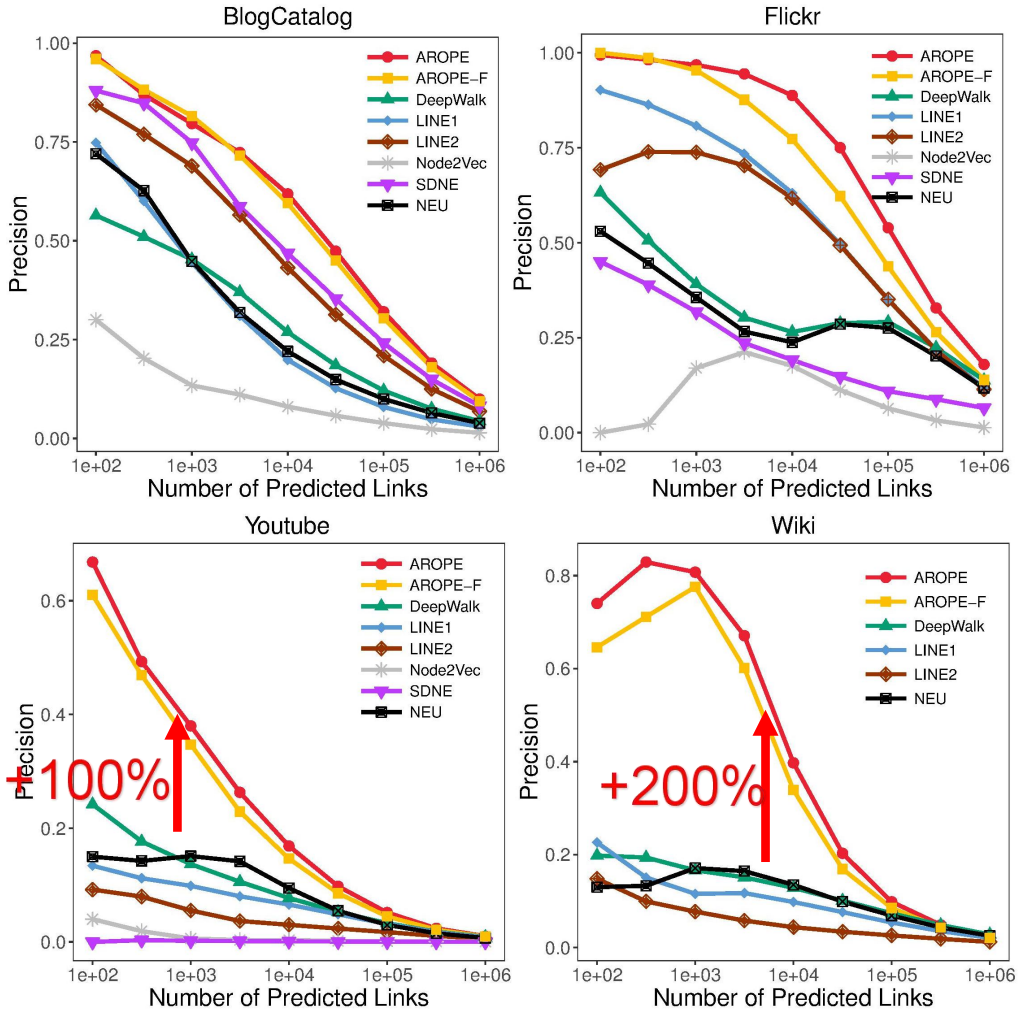


- Preserving arbitrary-order proximity
- Low marginal cost
- Accurate and efficient

Z. Zhang, et al. Arbitrary-Order Proximity Preserved Network Embedding. *KDD, 2018.*

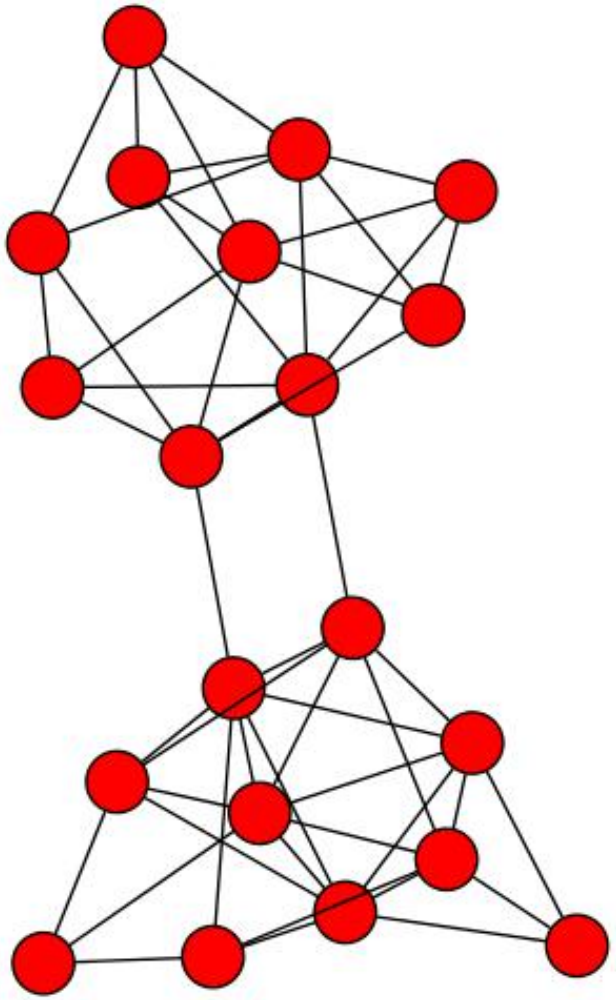
Experimental Results

- Link Prediction



Z. Zhang, et al. Arbitrary-Order Proximity Preserved Network Embedding. *KDD, 2018.*

Network Structures



Nodes & Links

Pair-wise Proximity

Community Structures

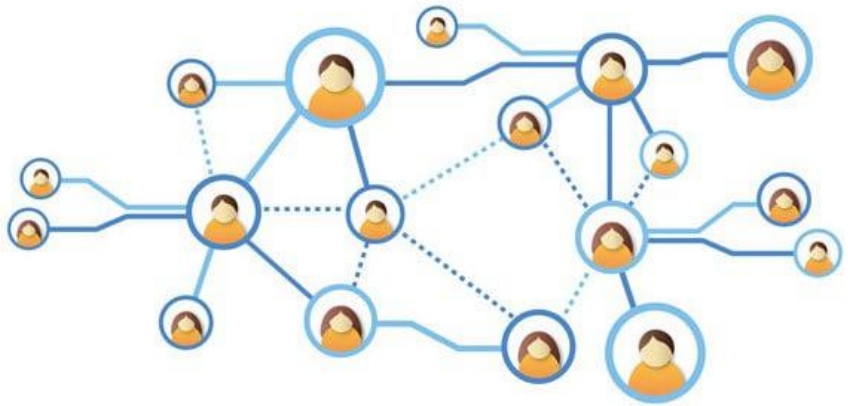
Hyper Edges

Global Structure

Motivation

- Vertexes in different parts of the network may have similar roles(global position)

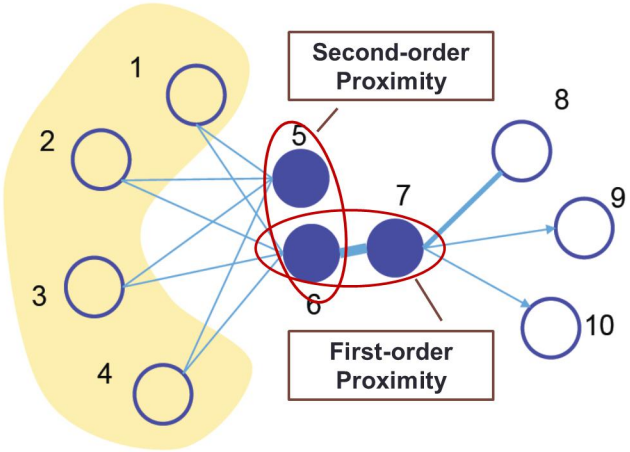
- Example：
  - Managers in the social network of a company
  - Outliers in a network in the task of anomaly detection

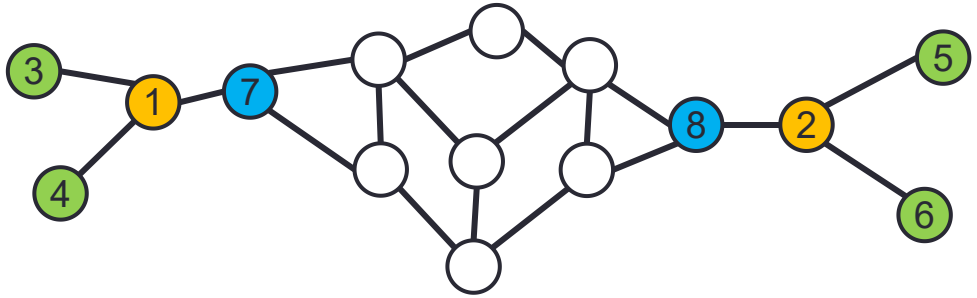


Social network with different positio

How to reflect the role or importance of a vertex in embedding space?

Existing embedding methods

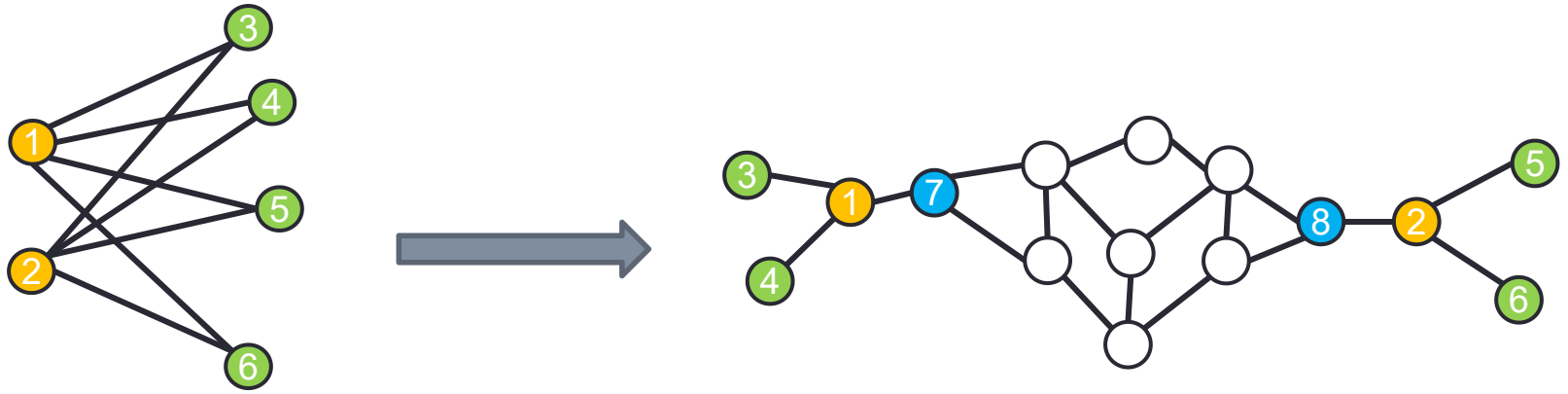




- They can only preserve local proximity(Structural equivalence), can not reflect the global position
  - Embeddings of node 5,6 in left network will be similar but embeddings of node 1, 2 in right network will not be similar.

Regular Equivalence

Two nodes are regularly equivalent if their network neighbors are themselves similar (i.e. regularly equivalent).



- Structural equivalence s
  - $N(u) = N(v)$
    - Direct way
    - Common neighbors

- Regular equivalence r
  - $\{r(i)|i \in N(u)\} = \{r(j)|j \in N(v)\}$
    - Recursive way
    - Similar global position

Regular equivalence is largely ignored in network embedding

Naïve Solutions

- Basis: two regularly equivalent nodes should have similar embeddings
  1. Explicitly calculate the regular equivalence of all vertex pairs
     - infeasible for large-scale networks due to the high complexity of calculating regular equivalence
  2. Replace regular equivalence into simpler graph theoretic metrics
     - centrality measures
     - one centrality can only capture a specific aspect of network role
     - some centrality measures also bear high computational complexity

Ke Tu, et al. Deep Recursive Network Embedding with Regular Equivalence. ***KDD***, 2018.

Deep Recursive Network Embedding

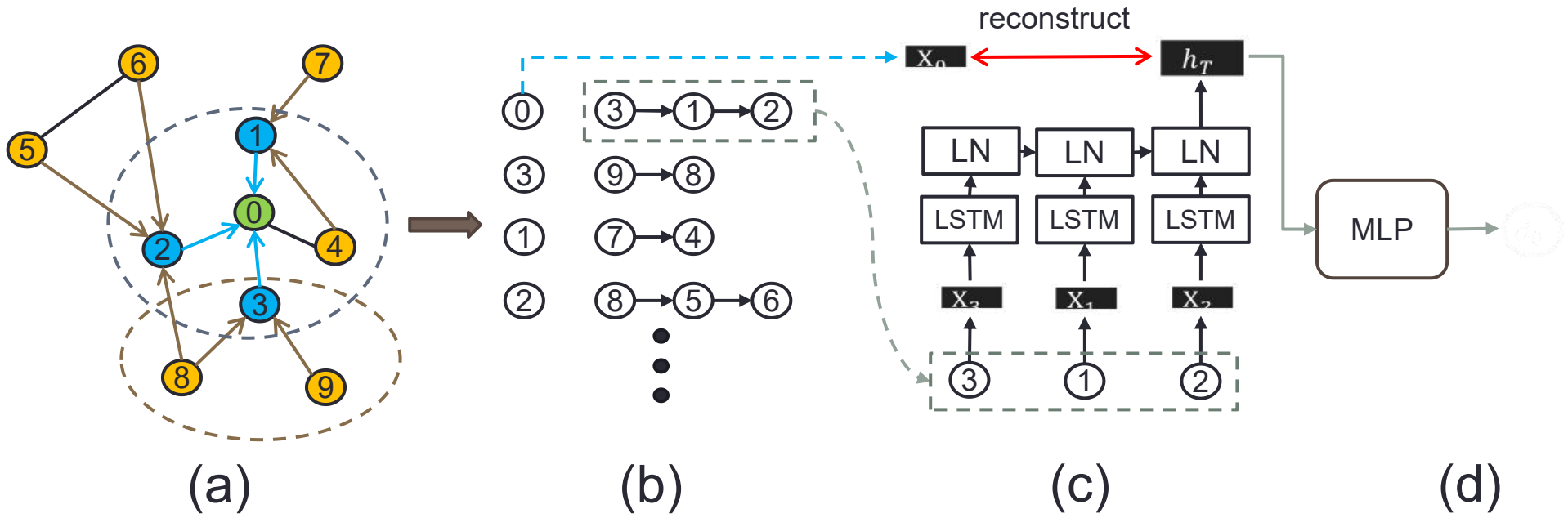
- The definition of regular equivalence is recursive
  - Aggregating neighbors' information in a **recursive** way

$$\mathcal{L}_1 = \sum_{v \in V} ||\mathbf{X}_v - Agg(\{\mathbf{X}_u | u \in \mathcal{N}(v)\})||_F^2,$$

- How to design the aggregating function
  - Variable length of neighbors
  - Highly nonlinear
  - → Layer-normalized LSTM

Ke Tu, et al. Deep Recursive Network Embedding with Regular Equivalence. ***KDD***, 2018.

Deep Recursive Network Embedding



(a) Sampling neighborhoods
(b) Sorting neighborhoods by their degree
(c) Aggregate neighbors
(d) A Weakly guided regularizer

Ke Tu, et al. Deep Recursive Network Embedding with Regular Equivalence. ***KDD***, 2018.

Theoretical Analysis

THEOREM 3.5. *If the centrality $C(v)$ of node v satisfies that $C(v) = \sum_{u \in \mathcal{N}(v)} F(u)C(u)$ and $F(v) = f(\{F(u), u \in \mathcal{N}(v)\})$ where f is any computable function, then $C(v)$ is one of the optimal solutions of our model.*

Centrality	Definition $C(v)$	$F(v)$	$f(\{x_i\})$
Degree	$d_v = \sum_{u \in \mathcal{N}(v)} I(d_u)$	$1/d_v$	$1/(\sum I(x_i))$
Eigenvector	$1/\lambda * \sum_{u \in \mathcal{N}(v)} C(u)$	$1/\lambda$	mean
PageRank	$\sum_{u \in \mathcal{N}(v)} 1/d_u * C(u)$	$1/d_v$	$1/(\sum I(x_i))$

Ke Tu, et al. Deep Recursive Network Embedding with Regular Equivalence. ***KDD***, 2018.

Experiment --- predict centrality

centrality	closeness	betweenness	eignvector	k-core
DeepWalk	0.6016	3.7188	2.1543	13.2755
LINE	0.5153	4.3919	1.5072	15.8179
node2vec	1.0489	3.4065	3.9436	39.2156
struc2vec	0.2365	0.25371	1.0544	9.0858
DRNE	**0.1909**	**0.1261**	**0.5267**	**5.5683**

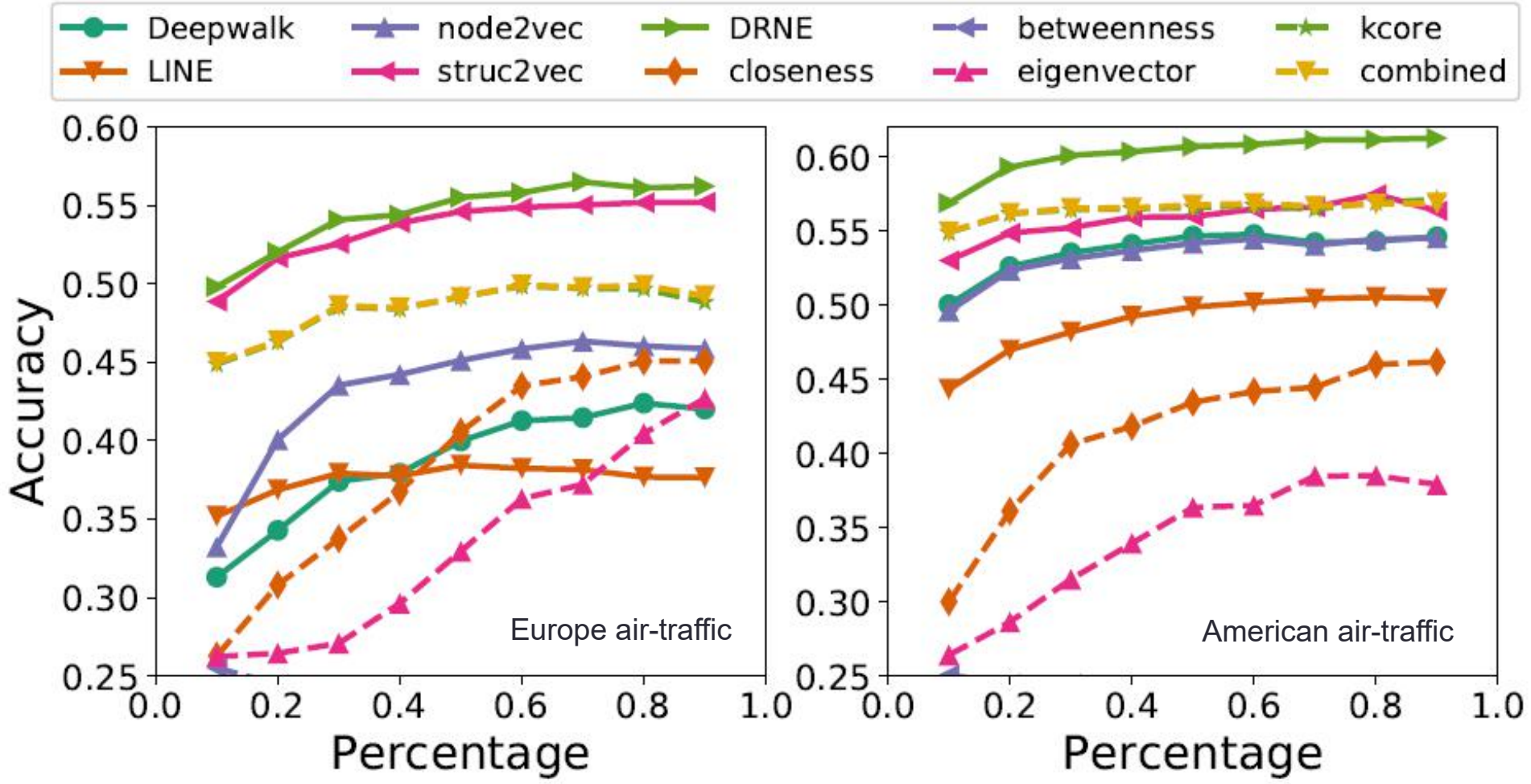
The MSE value of predicting centralities on Jazz dataset (∗ 10−2)

centrality	closeness	betweenness	eignvector	k-core
DeepWalk	0.2982	1.7836	1.1194	19.7016
LINE	0.3979	1.8425	1.5167	34.9079
node2vec	0.3573	1.6958	1.1432	24.1704
struc2vec	0.2947	1.6018	1.0445	25.3047
DRNE	**0.1101**	**0.6676**	**0.3108**	**7.7210**

The MSE value of predicting centralities on BlogCatalog dataset (∗ 10−2)

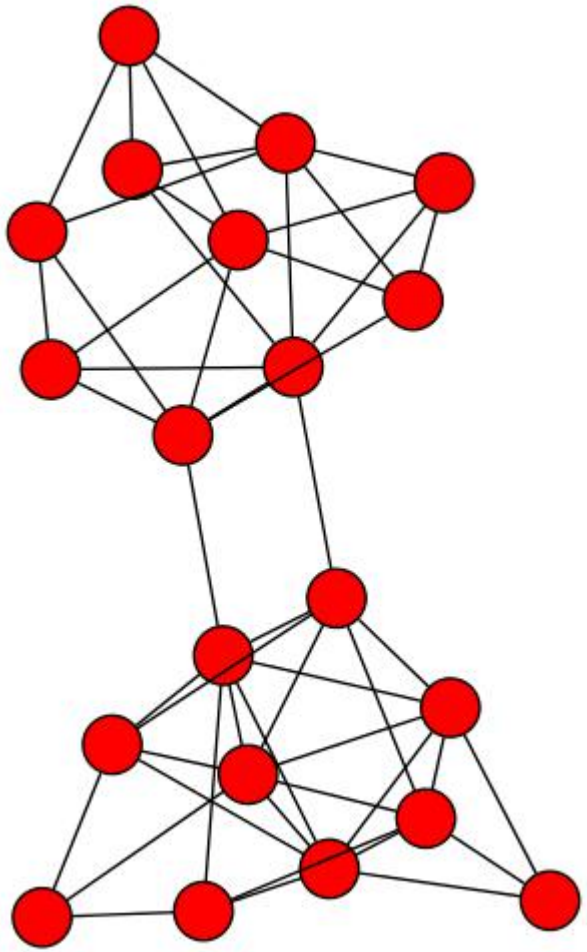
Ke Tu, et al. Deep Recursive Network Embedding with Regular Equivalence. ***KDD***, 2018.

Experiment - Structural Role Classification



Ke Tu, et al. Deep Recursive Network Embedding with Regular Equivalence. ***KDD***, 2018.

Section Summary



Nodes & Links

Node Neighborhood

Pair-wise Proximity

Community Structures

Hyper Edges

Global Structure

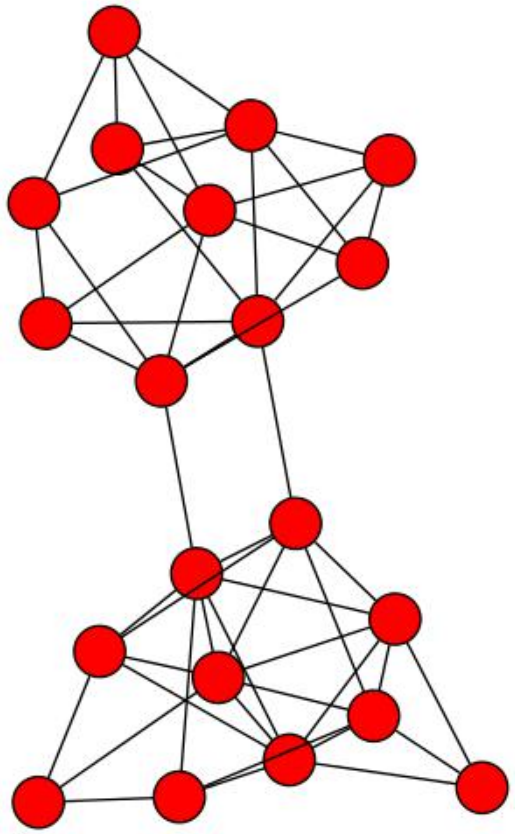
Network Characteristics

Application Characteristics

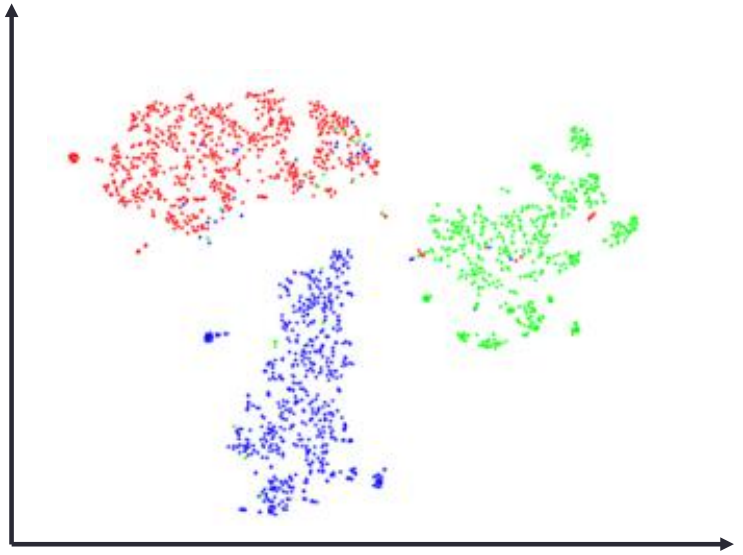
Outline

- Structure-preserved network embedding
- Property-preserved network embedding
- Dynamic network embedding

Why preserve network properties?



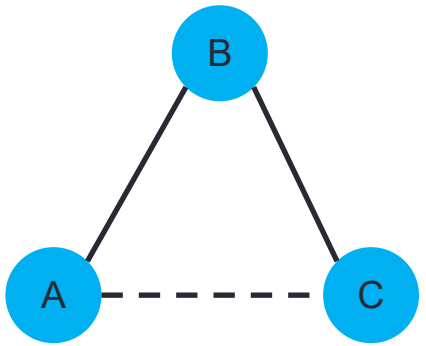
Heterogeneity



Transitivity

The Transitivity Phenomenon

Network



Embedding Space

Triangle Inequality: $D(A,B) + D(B,C) > D(A,C)$

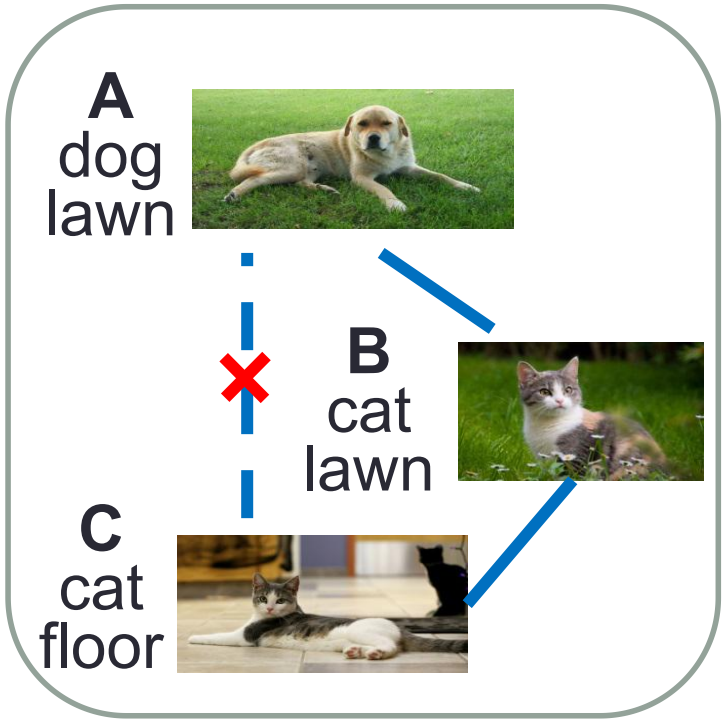
A close to ***B***, ***B*** close to ***C***, ⟶ ***A*** relatively close to ***C***

However, real network data is complex…

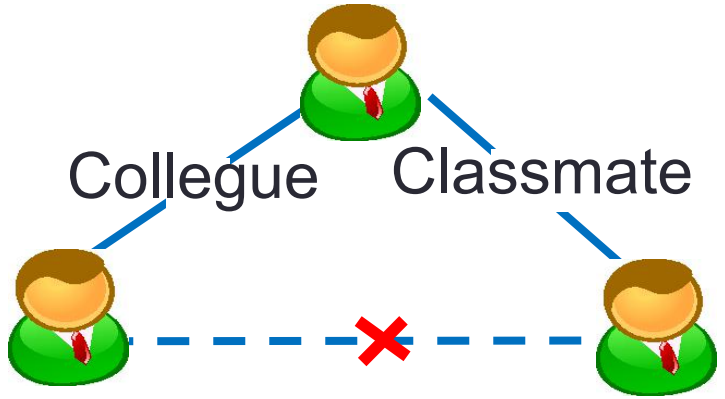
Non-transitivity

The Co-existence of *Transitivity* and *Non-transitivity*

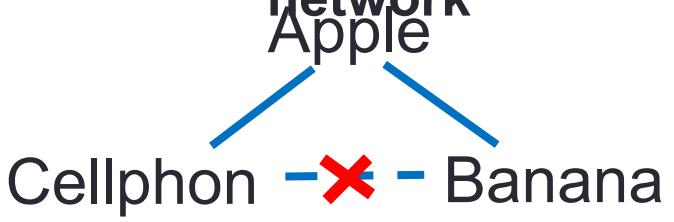
Image network



Social network



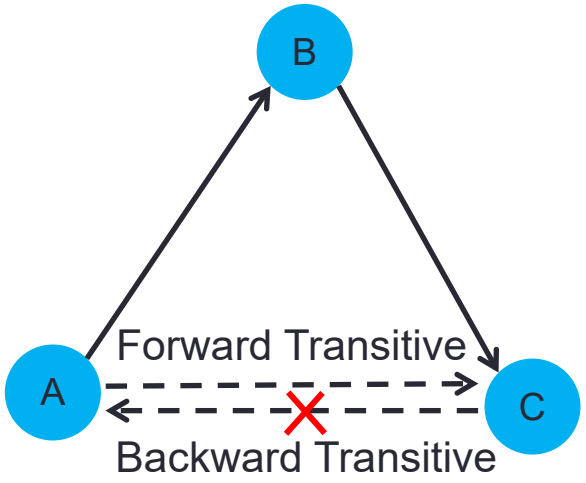
Word network



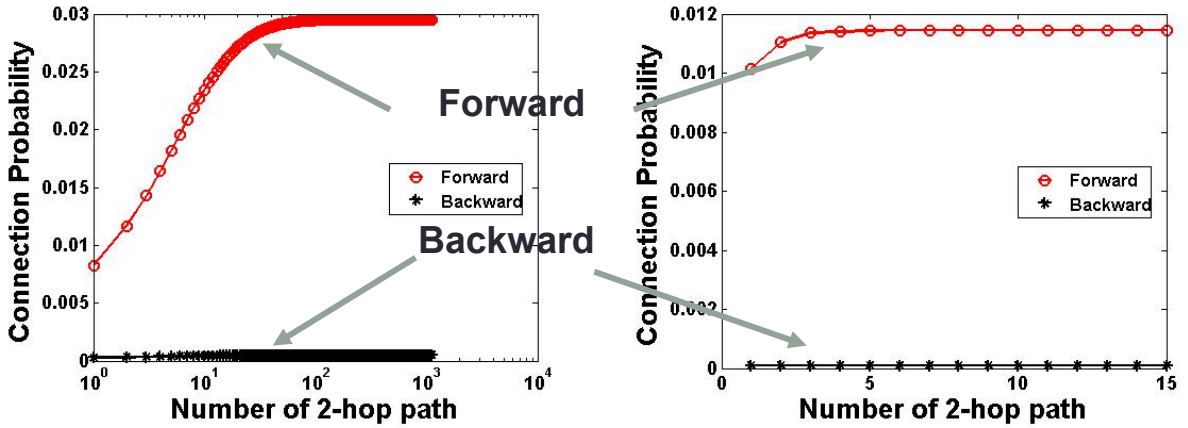
How to incorporate non-transitivity in embedding space?

Asymmetric Transitivity

Directed Network



A→B, B → C => A → C, but not C →A



Tencent Microblog

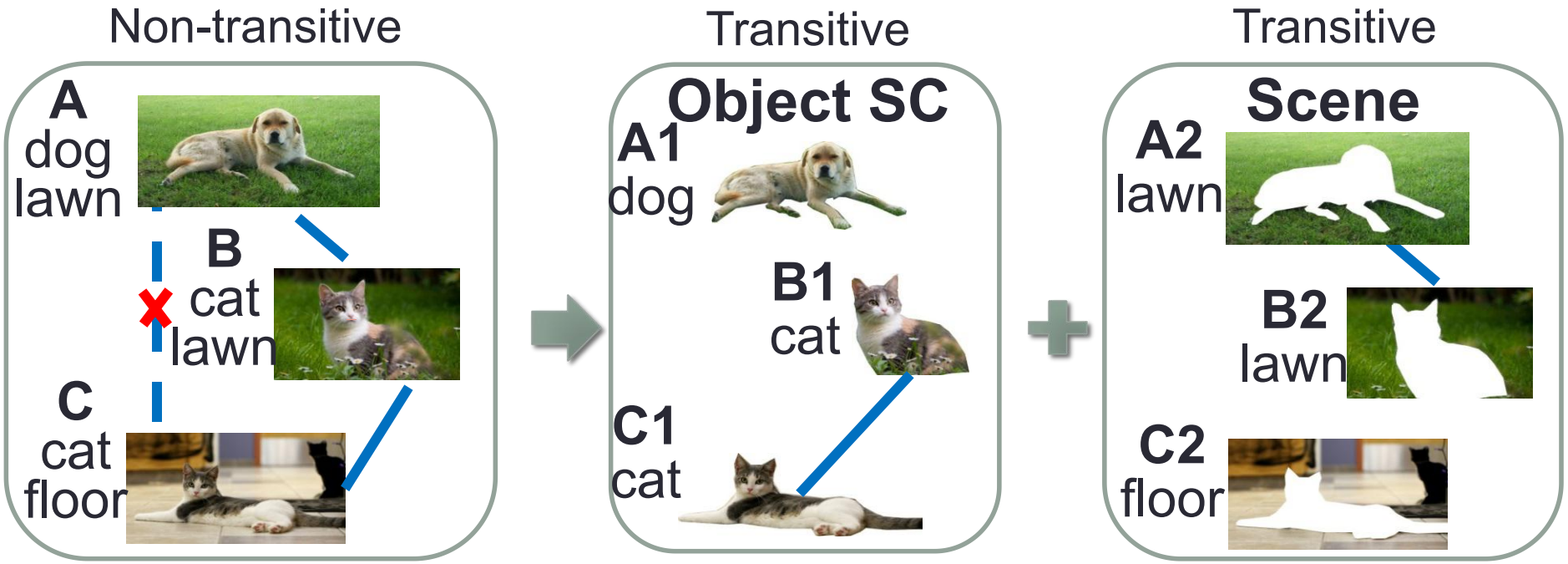
Twitter

Distance metric in embedding space is symmetric.
How to incorporate *Asymmetric Transitivity*?

Non-transitivity

The source of non-transitivity:

Each node has multiple similarity components

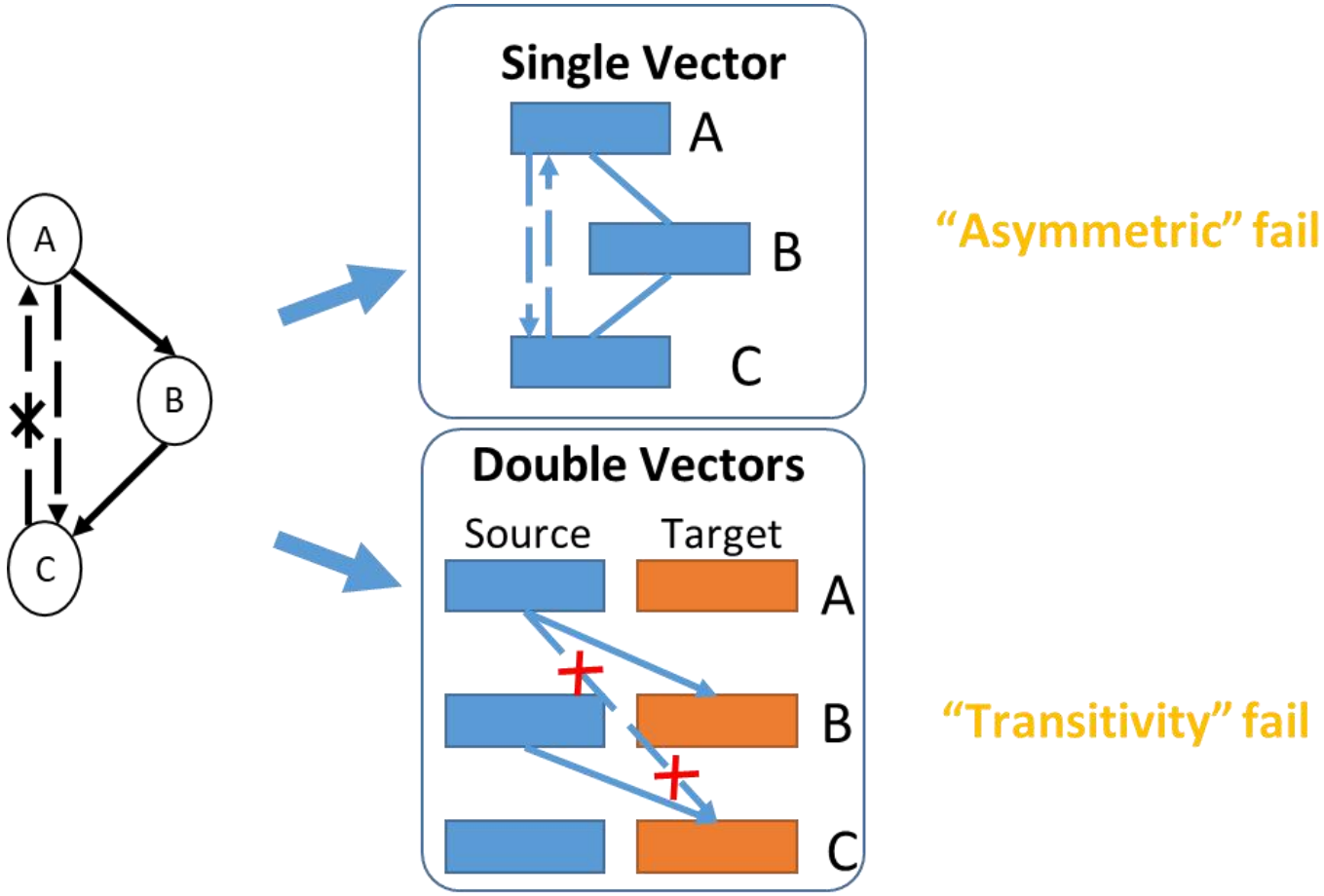


Non-transitive Embedding: represent non-transitive data with multiple latent similarity components

M. Ou, et al. Non-transitive Hashing with Latent Similarity Components. *KDD*, 2015.

Asymmetric Transitivity

All existing methods fail..

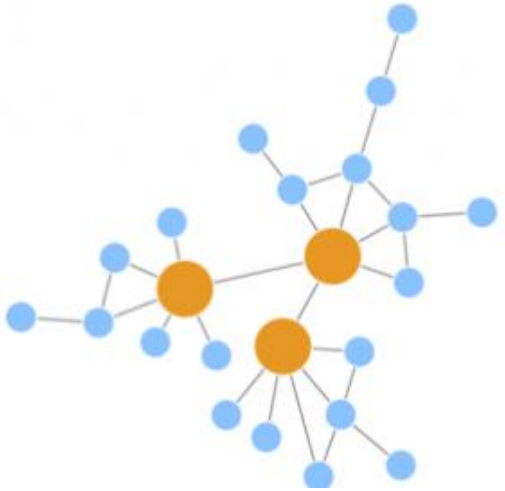


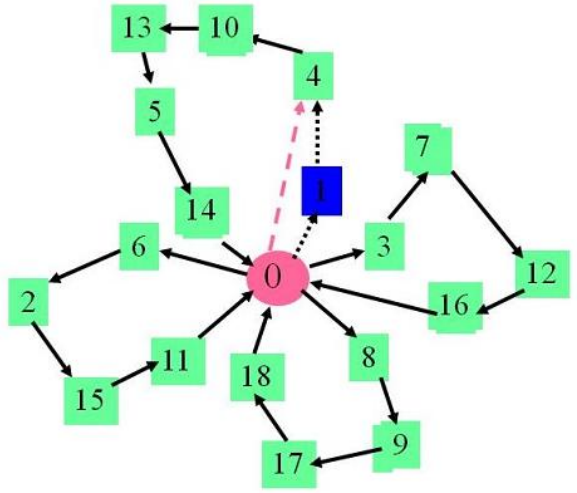
Mingdong Ou, **Peng Cui**, Jian Pei, Wenwu Zhu. Asymmetric Transitivity Preserving Graph Embedding. *KDD*, 2016.

Uncertainties in Networks

- The formation and evolution of real-world networks are full of uncertainties
  - E.g., for the nodes with low degree, they contain less information and thus their representations bear more uncertainties than others.

  - E.g., for the nodes across multiple communities, the possible contradiction between their neighboring nodes may also be large and thus cause the uncertainty.





DVNE for Structure and Uncertainty

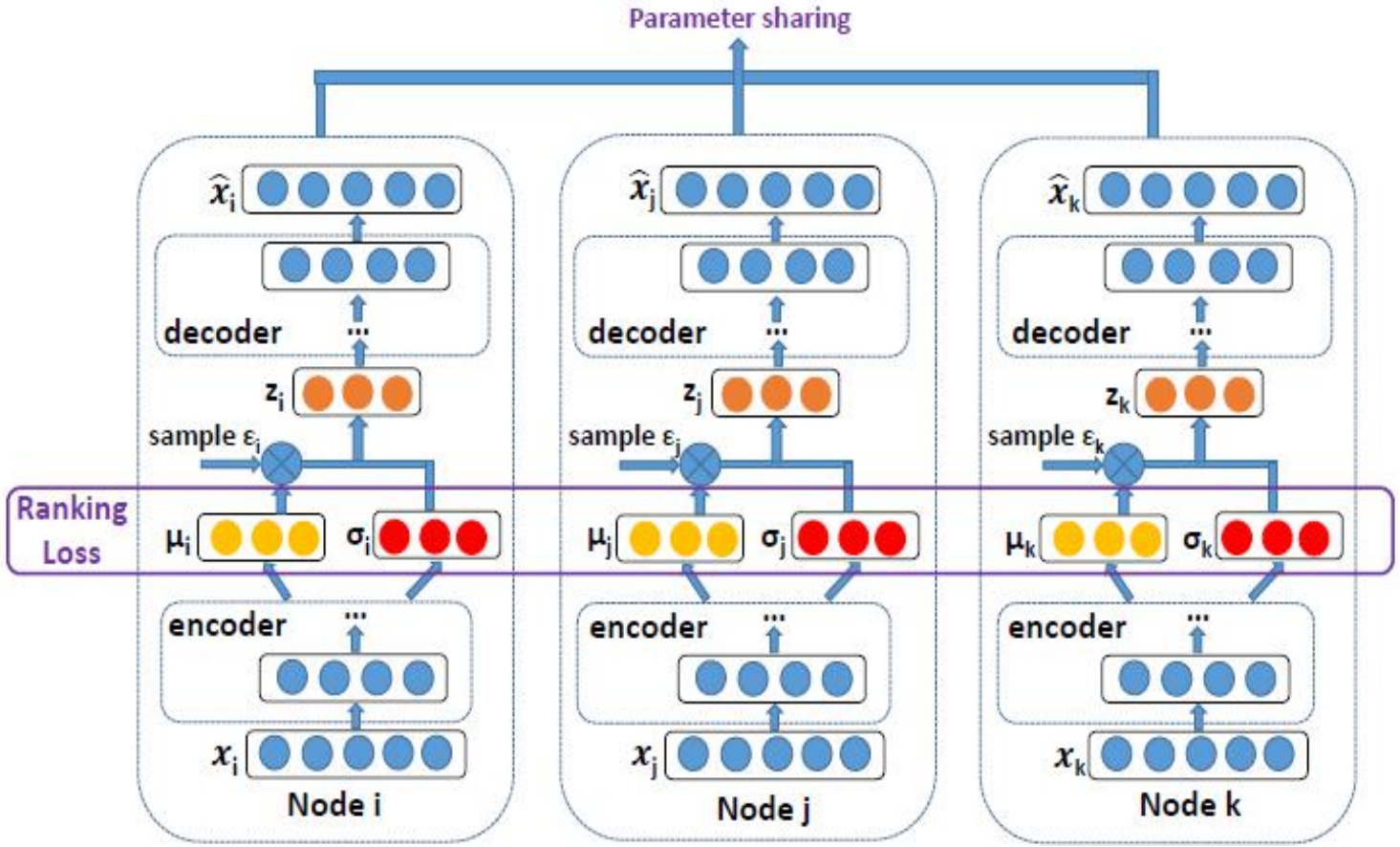


Figure 1: The framework of DVNE.

Dingyuan Zhu, et al. Deep Variational Network Embedding in Wasserstein Space. ***KDD***, 2018.

Section Summary

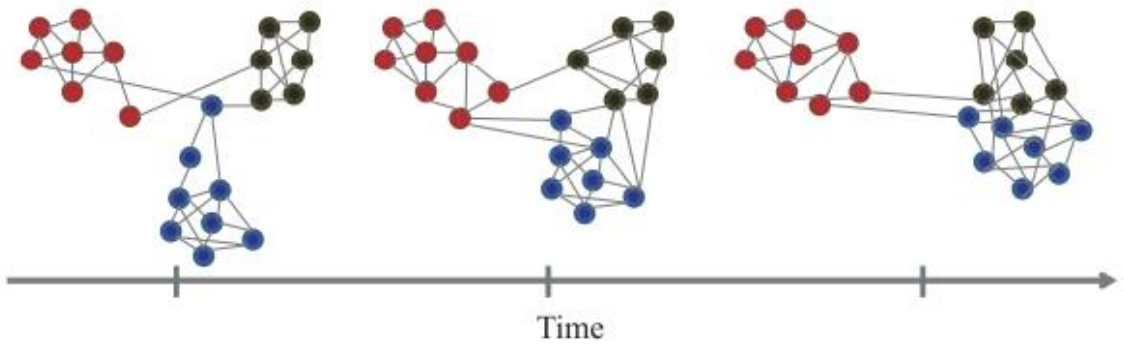
- Compared with network structures, **network properties** have large space to explore in network embedding.
- Transitivity is important for network inference.
- Uncertainty provides evidence in making network inference.
- Many other property issues:
  - The right embedding space: Euclidean space?
  - Power-law distribution
  - …

Outline

- Structure-preserved network embedding
- Property-preserved network embedding
- Dynamic network embedding

Dynamic Networks

- **Networks are dynamic in nature**
  - **New (old) nodes are added (deleted)**
    - New users, products, etc.
  - **The edges between nodes evolve over time**
    - Users add or delete friends in social networks, or neurons establish new connections in brain networks.
- **How to efficiently incorporate the dynamic changes when networks evolve?**

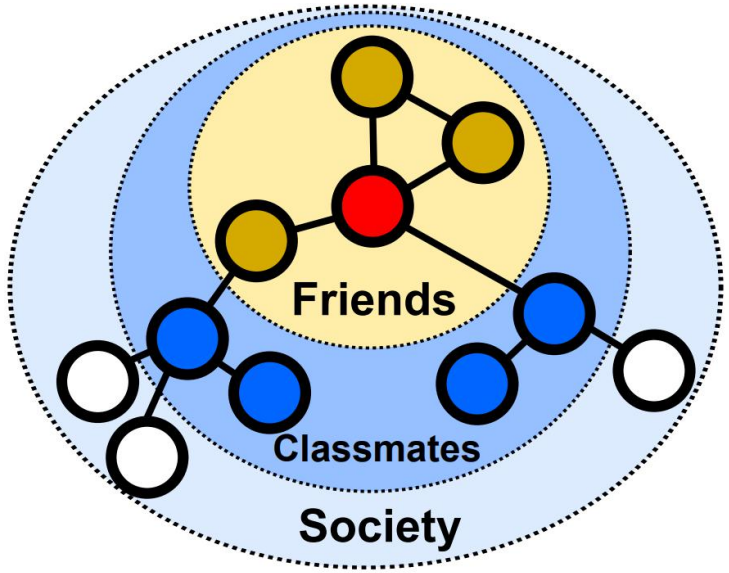


Key problems in dynamic network embedding

- I : Out-of-sample nodes
- II : Incremental edges
- III: Aggregated error
- IV: Scalable optimization

Challenge: High-order Proximity

- **High-order proximity**
  - Critical structural property of networks
  - Measure indirect relationship between nodes
  - Capture the structure of networks with different scales and sparsity



Network Embedding V.S. Traditional Graph Embedding

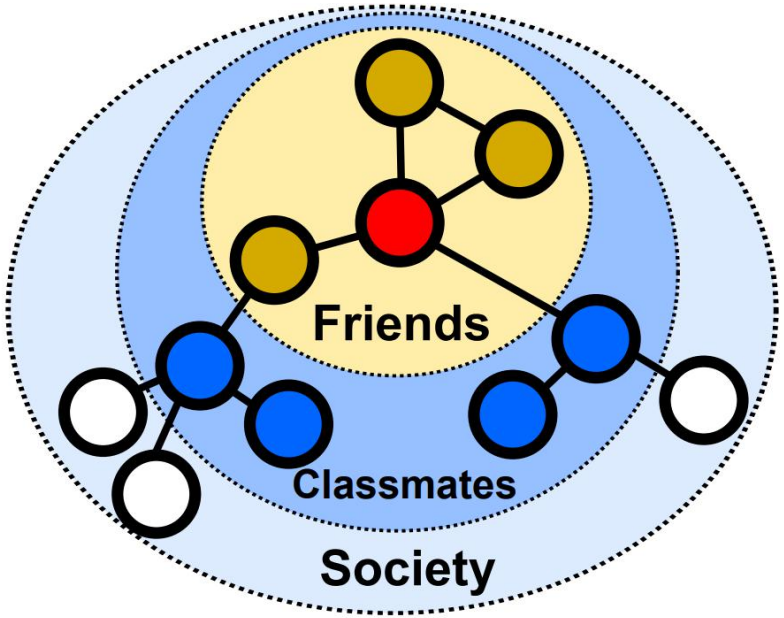
Challenge: High-order Proximity

I : Out-of-sample nodes

II : Incremental edges

III: Aggregated error

IV: Scalable optimization



Preserve High-order Proximities

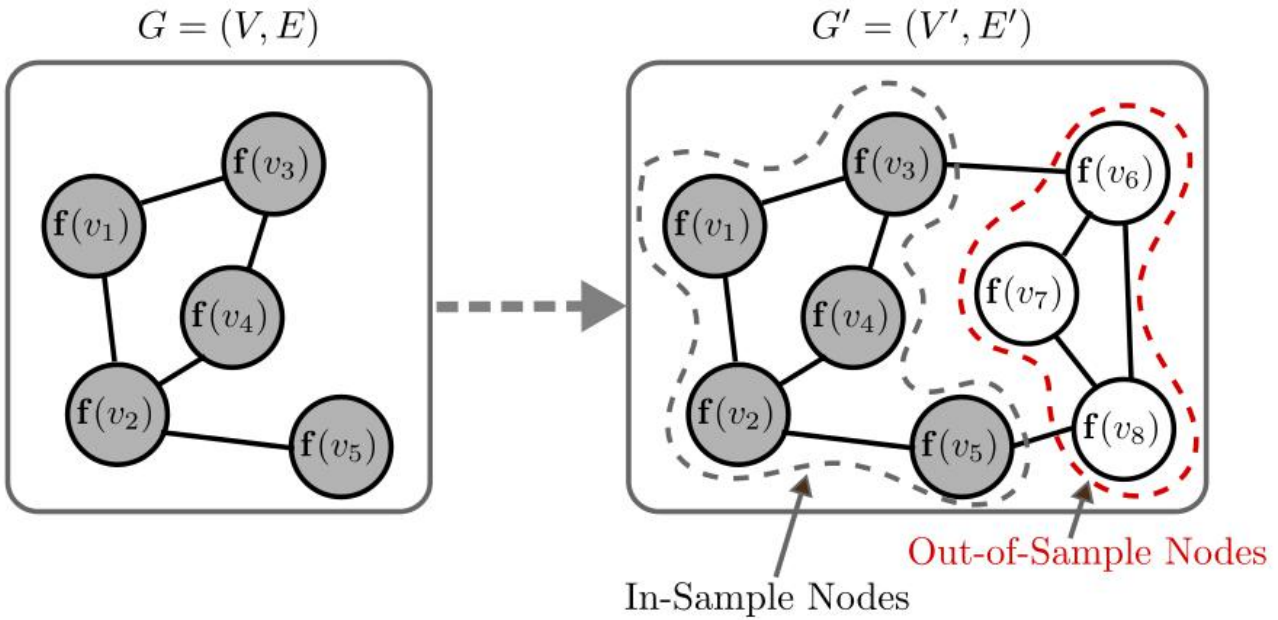
Local Change leads to Global Updating

Key problems in dynamic network embedding

- I : Out-of-sample nodes
- II : Incremental edges
- III: Aggregated error
- IV: Scalable optimization

Problem

- To infer embeddings for out-of-sample nodes.



- G=(V, E) evolves into G'=(V', E'), where V' = V ∪ V*.
- n old nodes: V = {v1,…,vn}, m new nodes: V* = {vn+1,…,vn+m}
- Network embedding: f: V→Rd
- We know f(v) for old nodes, want to infer f(v) for new nodes.

Challenges

- Preserve network structures
  - e.g. high-order proximity
  - need to incorporate prior knowledge on networks
- Share similar characteristics with in-sample embeddings
  - e.g. magnitude, mean, variance
  - requires a model with great expressive power to fit the data well
- Low computational cost

Jianxin Ma, et al. DepthLGP: Learning Embeddings of Out-of-Sample Nodes in Dynamic Networks. ***AAAI***, 2018.

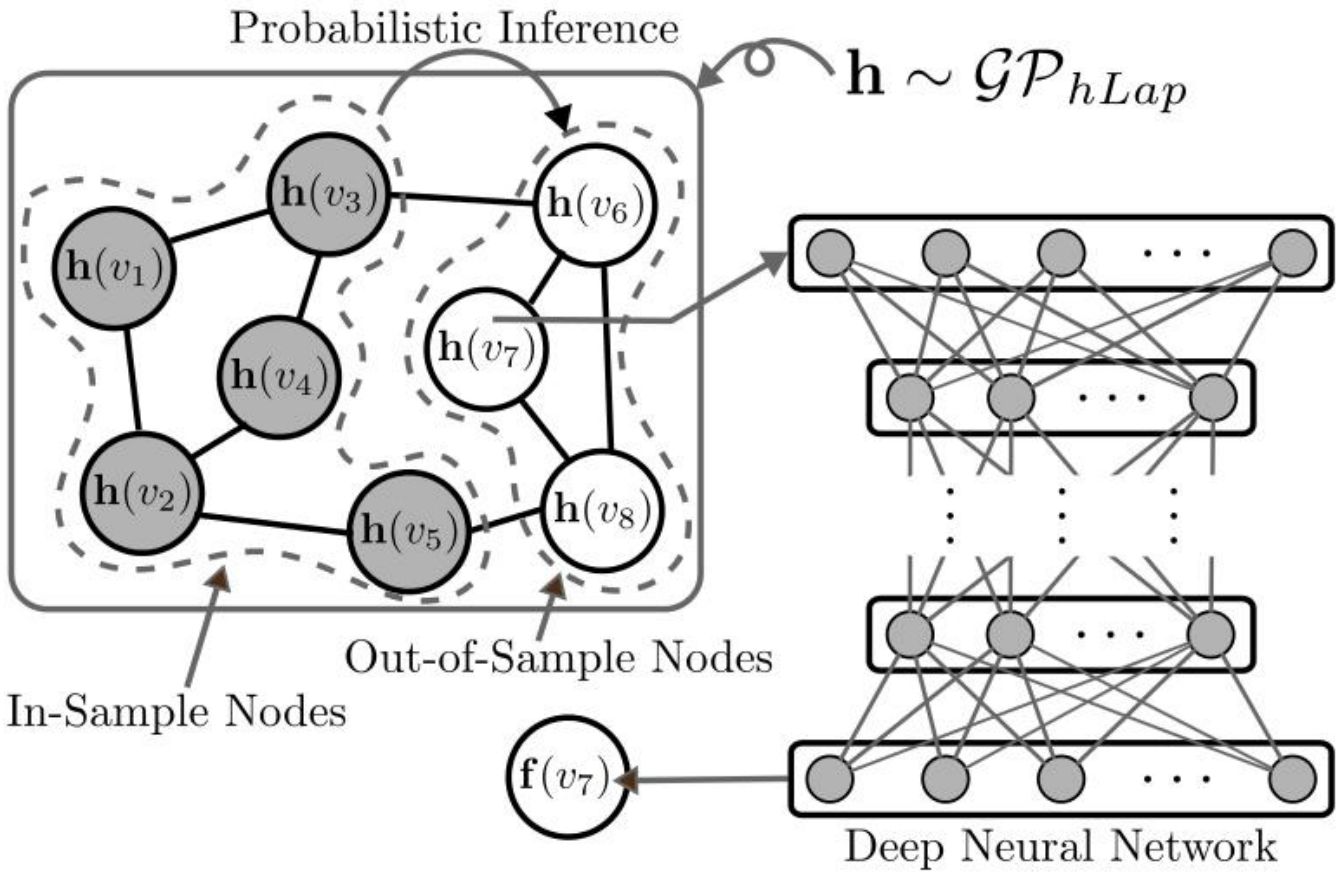
Specific vs. General

- Specific
  - A new NE algorithm capable of handling OOS nodes.
- General
  - A solution that helps an arbitrary NE algorithm handle OOS nodes.
- We propose a **general** solution.
  - But it can be easily integrated into an existing NE algorithm (e.g. DeepWalk) to derive a **specific** algorithm (see the paper).

Jianxin Ma, et al. DepthLGP: Learning Embeddings of Out-of-Sample Nodes in Dynamic Networks. ***AAAI***, 2018.

DepthLGP

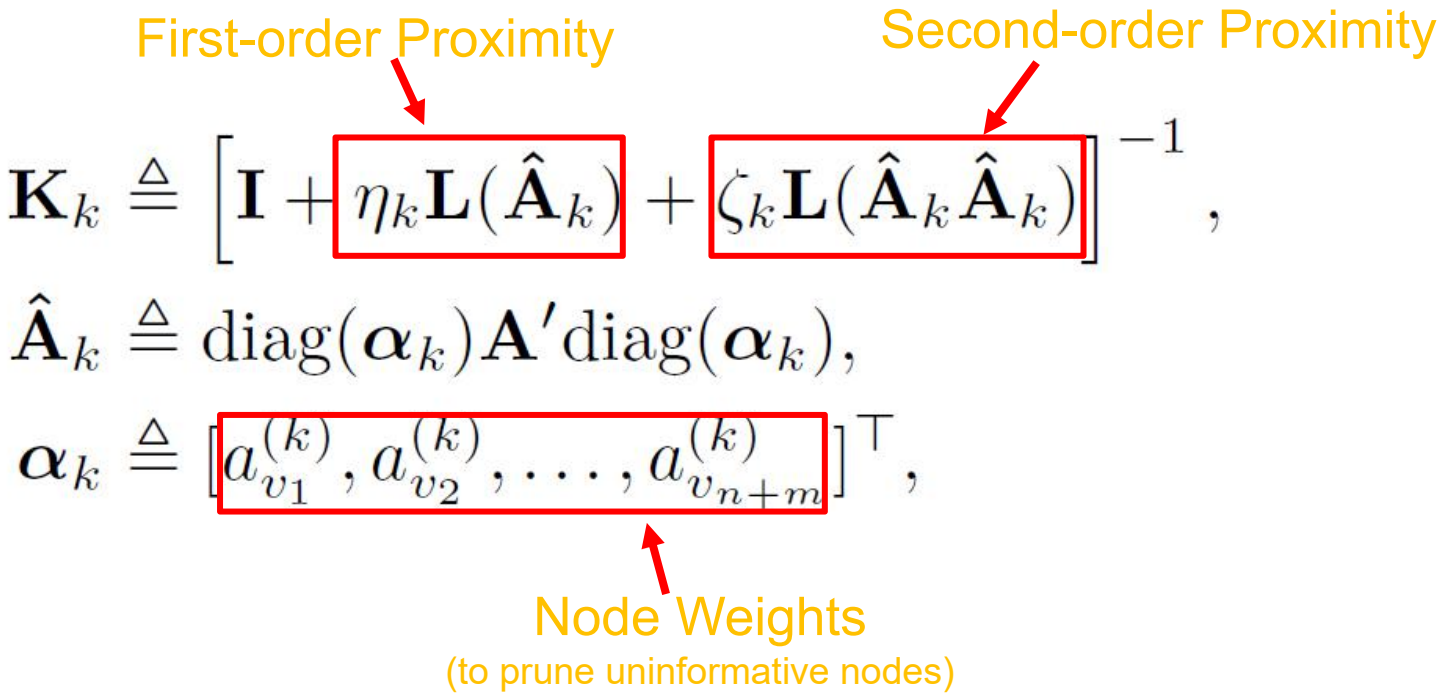
- Nonparametric probabilistic modeling + Deep Learning



Jianxin Ma, et al. DepthLGP: Learning Embeddings of Out-of-Sample Nodes in Dynamic Networks. ***AAAI***, 2018.

DepthLGP

- Design a kernel for the *k*th (*k=1,…,s*) dimension of $h(\cdot)$



[1]The matrix inversion can be bypassed without approximation.

[2]$a^{(k)}_v$ indicates how much attention we pay to a node. It is learned for an in-sample node, but fixed to one for an OOS node, as we are always interested in OOS nodes.

Task I: Classification

Metric	Embedding	Network	Baselines			This Work		Upper Bound
			LocalAvg	MRG	LabelProp	hLGP	DepthLGP	(rerunning)
Macro-F1(%)	LINE	DBLP	37.89	42.15	40.83	47.33	**48.25**	(49.07)
		PPI	10.52	10.02	12.42	13.42	**13.72**	(13.91)
		BlogCatalog	13.25	11.30	17.07	17.41	**18.03**	(18.90)
	GraRep	DBLP	50.61	55.79	55.02	57.43	**58.67**	(62.92)
		PPI	13.65	13.75	12.38	14.80	**14.84**	(15.33)
		BlogCatalog	14.76	14.80	14.71	15.94	**18.45**	(20.15)
	node2vec	DBLP	53.83	59.34	59.25	60.89	**62.63**	(64.87)
		PPI	15.05	13.43	13.78	15.85	**16.54**	(16.81)
		BlogCatalog	15.10	14.04	19.16	19.77	**20.32**	(20.82)
Micro-F1(%)	LINE	DBLP	49.58	50.49	50.88	54.01	**54.94**	(55.84)
		PPI	18.10	15.71	18.81	20.71	**21.42**	(21.43)
		BlogCatalog	27.40	23.21	30.79	31.36	**31.90**	(32.20)
	GraRep	DBLP	60.17	60.62	60.48	61.44	**62.29**	(65.44)
		PPI	20.23	20.35	20.23	20.79	**21.44**	(21.88)
		BlogCatalog	36.44	30.79	33.90	37.57	**38.14**	(38.37)
	node2vec	DBLP	60.54	62.29	62.52	62.83	**64.56**	(65.63)
		PPI	19.70	18.25	18.25	22.63	**23.11**	(23.41)
		BlogCatalog	34.83	25.82	36.94	37.96	**39.64**	(40.34)

Jianxin Ma, et al. DepthLGP: Learning Embeddings of Out-of-Sample Nodes in Dynamic Networks. ***AAAI***, 2018.

Key problems in dynamic network embedding

- I : Out-of-sample nodes
- II : Incremental edges
- III: Aggregated error
- IV: Scalable optimization

The Static Model

- We aim to preserve high-order proximity in the embedding matrix with the following objective function：

$$\min ||\mathbf{S} - \mathbf{U}\mathbf{U}'^{\top}||_F^2$$

  - where S denotes the high-order proximity matrix of the network
  - U and U’ is the results of matrix decomposition of S.

- For undirected networks, U and U’ are highly correlated.
  - Without loss of generality, we choose U as the embedding matrix.

Dingyuan Zhu, et al. High-order Proximity Preserved Embedding For Dynamic Networks. ***IEEE TKDE***, 2018.

GSVD

- We choose Katz Index as S because it is one of the most widely used measures of high-order proximity.

$$\mathbf{S}^{Katz} = {\mathbf{M}_a}^{-1}\mathbf{M}_b$$
$$\mathbf{M}_a = (\mathbf{I} - \beta\mathbf{A})$$
$$\mathbf{M}_b = \beta\mathbf{A}$$

  - where β is a decay parameter, **I** is the identity matrix and **A** is the adjacency matrix
- According to HOPE, the original objective function can be solved by the generalized SVD (GSVD) method

Generalized Eigen Perturbation

- We propose generalized eigen perturbation to fulfill the task.
  - The goal of generalized eigen perturbation is to update $X^{(t)}$ to $X^{(t+1)}$

- Specifically, given the change of adjacency matrix $\triangle A$ between two consecutive time steps, the change of Ma and Mb can be represented as:

$$\Delta \mathbf{M}_a = -\beta \Delta \mathbf{A},\ and\ \Delta \mathbf{M}_b = \beta \Delta \mathbf{A}$$

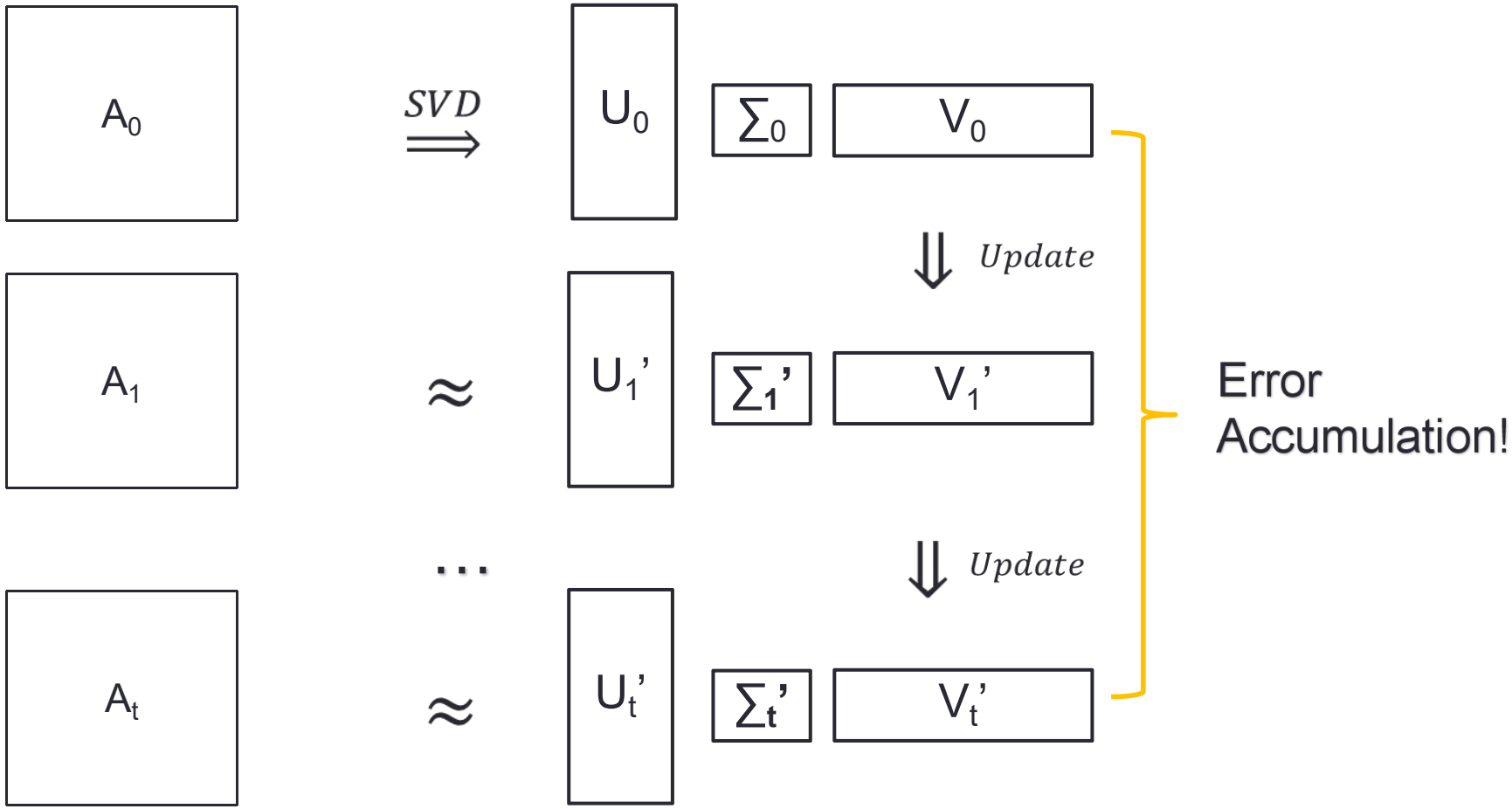
Dingyuan Zhu, et al. High-order Proximity Preserved Embedding For Dynamic Networks. ***IEEE TKDE***, 2018.

Key problems in dynamic network embedding

- I : Out-of-sample nodes
- II : Incremental edges
- III: Aggregated error
- IV: Scalable optimization

Problem: Error Accumulation

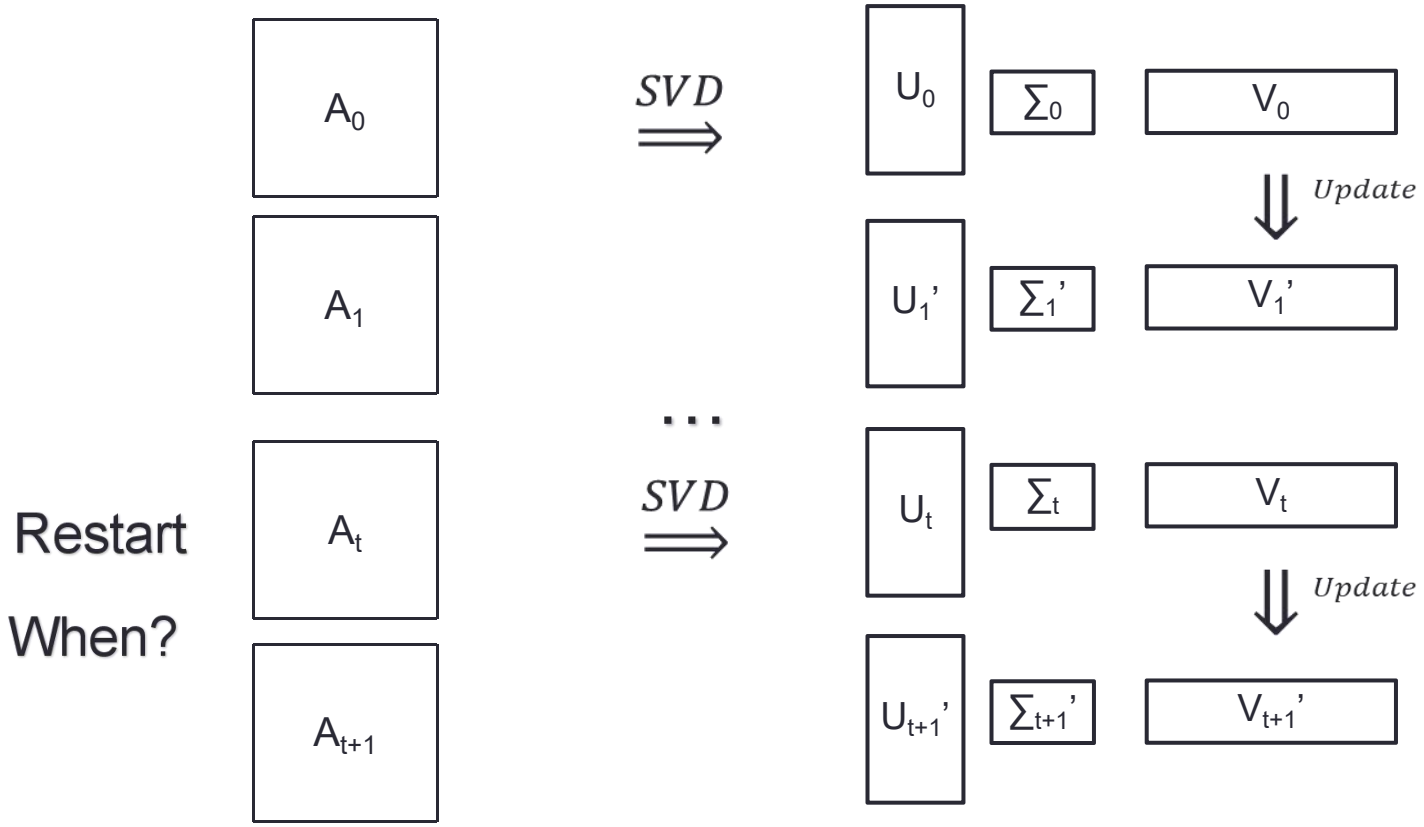
- Eigen perturbation is at the cost of inducing approximation



- Problem: error accumulation is inevitable

Solution: SVD Restarts

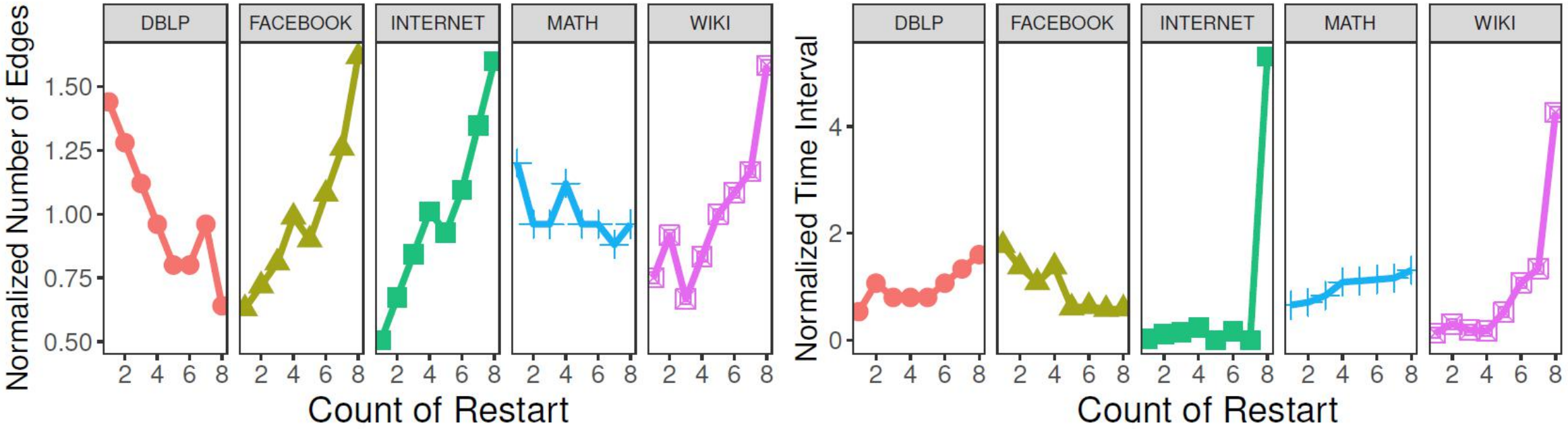
- Solution: restart SVD occasionally



- What are the appropriate time points?
  - Too early restarts: waste of computation resources
  - Too late restarts: serious error accumulation

Naïve Solution

- Naïve solution: fixed time interval or fixed number of changes
- Difficulty: error accumulation is not uniform



Ziwei Zhang, et al. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. ***AAAI***, 2018.

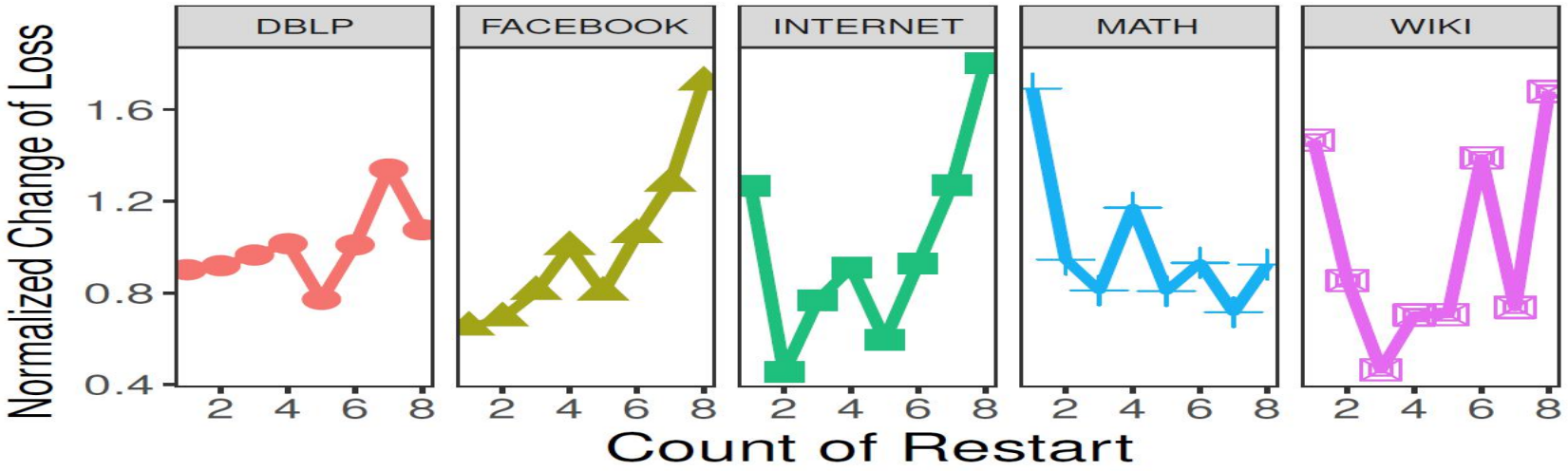
Existing Method

- Existing method: monitor loss (Chen and Candan, KDD 2014)
- Loss in SVD:

$$\mathcal{J} = \|S - U\Sigma V^T\|_F^2$$

S: target matrix, $[U, \Sigma, V]$: results of SVD

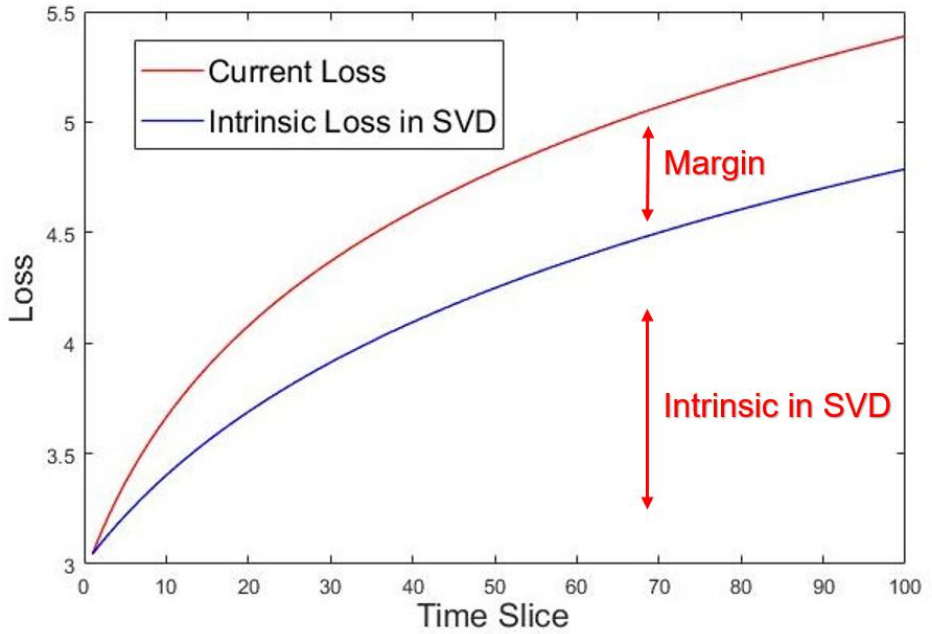
- Problem: loss includes approximation error and intrinsic loss in SVD



Ziwei Zhang, et al. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. ***AAAI***, 2018.

Framework: Monitor Margin

- Observation: the margin between the current loss and intrinsic loss in SVD is the actual accumulated error
  - Current loss: $\mathcal{J} = \|S - U\Sigma V^T\|_F^2$
  - Intrinsic loss: $\mathcal{L}(S,k) = \min_{U^*,\Sigma^*,V^*} \left\|S - U^*\Sigma^*V^{*T}\right\|_F^2, k: dimensionality$



Ziwei Zhang, et al. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. ***AAAI***, 2018.

Solution: Lazy Restarts

- Lazy restarts: restart only when the margin exceeds the threshold
- Problem: intrinsic loss is hard to compute
  - Direct calculation has the same time complexity as SVD
- Relaxation: an upper bound on margin
  - A lower bound on intrinsic loss $\mathcal{L}(S, k)$

$$\mathcal{L}(\mathbf{S}_t, k) \geq B(t) \Rightarrow \frac{\mathcal{J}(t) - \mathcal{L}(\mathbf{S}_t, k)}{\mathcal{L}(\mathbf{S}_t, k)} \leq \frac{\mathcal{J}(t) - B(t)}{B(t)}.$$

$\mathcal{J}(t)$: current loss; $\mathcal{L}(S_t, k)$: intrinsic loss; $B(t)$: bound of intrinsic loss

Ziwei Zhang, et al. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. ***AAAI***, 2018.

A Lower Bound of SVD Intrinsic Loss

- Idea: use matrix perturbation

Theorem 1 (A Lower Bound of SVD Intrinsic Loss). *If* $\mathbf{S}$ *and* $\Delta\mathbf{S}$ *are symmetric matrices, then:*

$$\mathcal{L}(\mathbf{S}+\Delta\mathbf{S},k) \geq \mathcal{L}(\mathbf{S},k) + \Delta tr^2(\mathbf{S}+\Delta\mathbf{S},\mathbf{S}) - \sum_{l=1}^{k}\lambda_l, \quad (9)$$

where $\lambda_1 \geq \lambda_2 ... \geq \lambda_k$ *are the top-*k *eigenvalues of* $\nabla_{S^2} = \mathbf{S}\cdot\Delta\mathbf{S} + \Delta\mathbf{S}\cdot\mathbf{S} + \Delta\mathbf{S}\cdot\Delta\mathbf{S}$, *and*

$$\Delta tr^2(\mathbf{S}+\Delta\mathbf{S},\mathbf{S}) = tr\left((\mathbf{S}+\Delta\mathbf{S})\cdot(\mathbf{S}+\Delta\mathbf{S})\right) - tr(\mathbf{S}\cdot\mathbf{S}).$$

- Intuition: treat changes as a perturbation to the original network

Ziwei Zhang, et al. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. ***AAAI***, 2018.

Time Complexity Analysis

Theorem 2. *The time complexity of calculating $B(t)$ in Eqn (13) is $O(M_S + M_L k + N_L k^2)$, where M_S is the number of the non-zero elements in $\Delta\mathbf{S}$, and N_L, M_L are the number of the non-zero rows and elements in ∇_{S^2} respectively.*

- If every node has a equal probability of adding new edges, we have: $M_L \approx 2d_{avg}M_S$, where d_{avg} is the average degree of the network .
- For Barabasi Albert model (Barabási and Albert 1999), a typical example of preferential attachment networks, we have: $M_L \approx \frac{12}{\pi^2}\left[log(d_{max}) + \gamma\right] M_S$, where d_{max} is the maximum degree of the network and $\gamma \approx 0.58$ is a constant.

- Conclusion: the complexity is only linear to the local dynamic changes

Ziwei Zhang, et al. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. ***AAAI***, 2018.

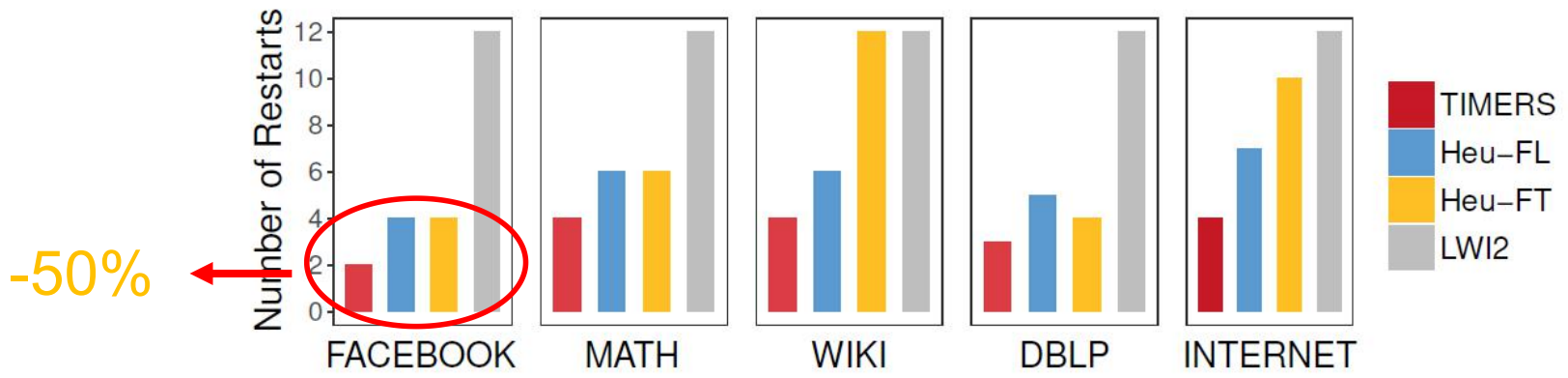
Experimental Results: Approximation Error

- Fixing number of restarts

Dataset	$avg(r)$				$max(r)$			
	TIMERS	LWI2	Heu-FL	Heu-FT	TIMERS	LWI2	Heu-FL	Heu-FT
FACEBOOK	**0.005**	0.020	0.009	0.011	**0.014**	0.038	0.025	0.023
MATH	**0.037**	0.057	0.044	0.051	**0.085**	0.226	0.117	0.179
WIKI	**0.053**	0.086	0.071	0.281	**0.139**	0.332	0.240	0.825
DBLP	**0.042**	0.110	0.053	0.064	**0.121**	0.386	0.198	0.238
INTERNET	**0.152**	0.218	0.196	0.961	**0.385**	0.806	0.647	1.897

27%~42% Improvement

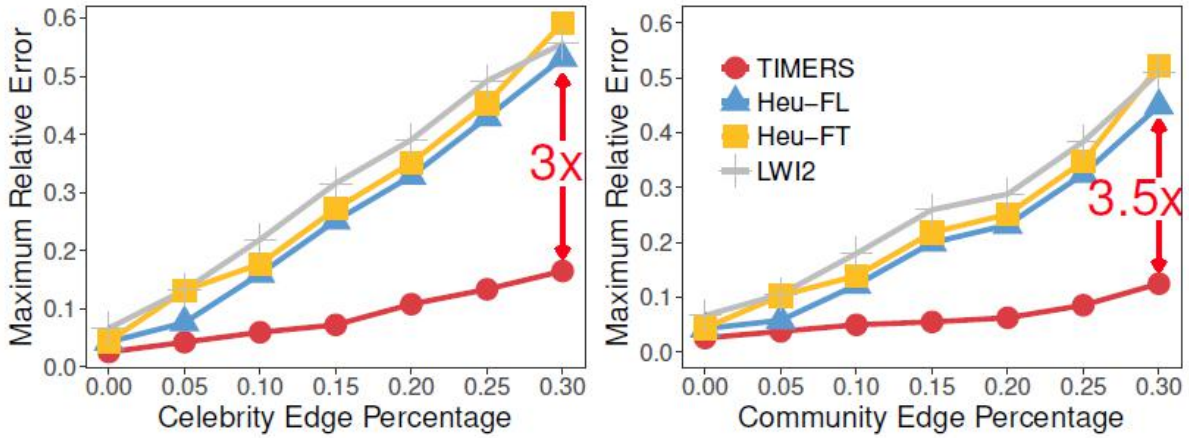
- Fixing maximum error



Ziwei Zhang, et al. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. ***AAAI***, 2018.

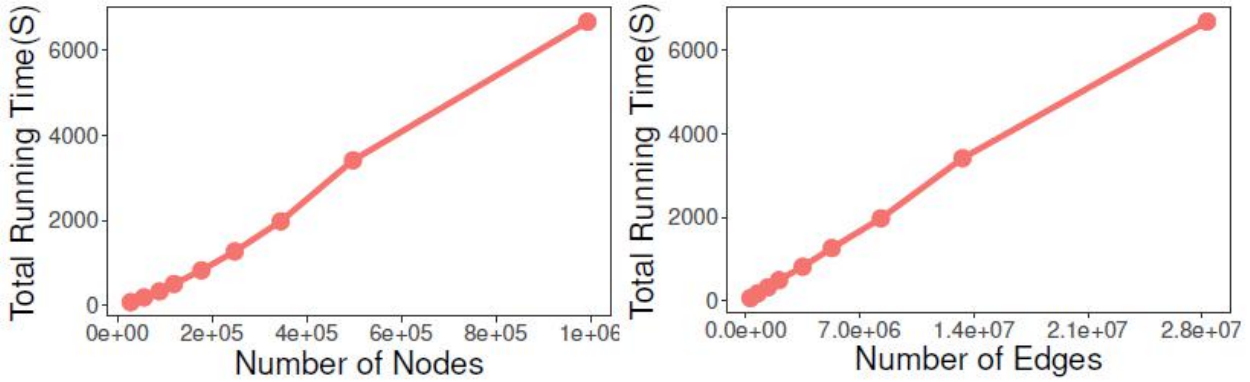
Experimental Results: Analysis

- Syntactic networks: simulate drastic changes in the network structure



  - Robust to sudden changes

- Linear scalability

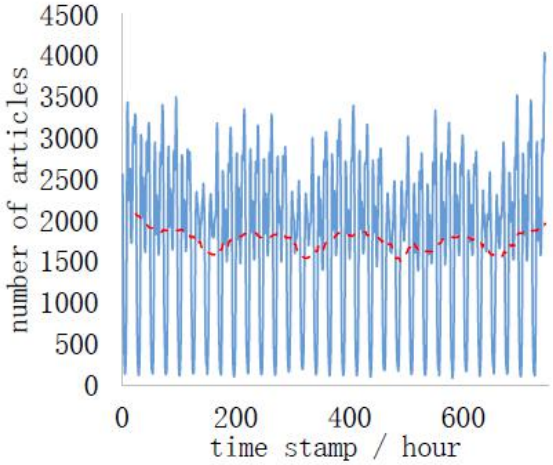


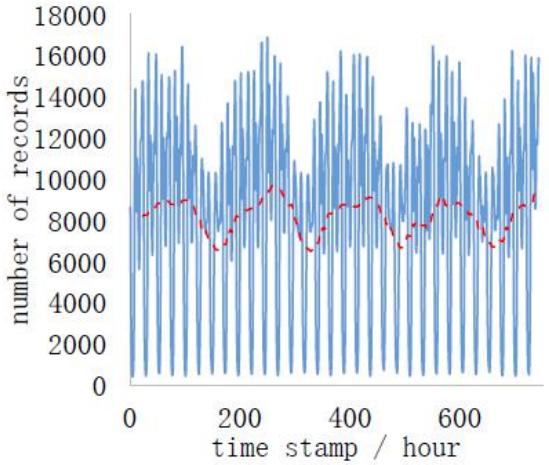
Key problems in dynamic network embedding

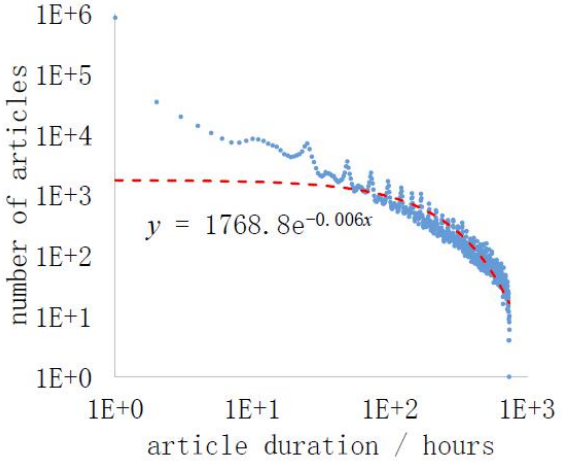
- I : Out-of-sample nodes
- II : Incremental edges
- III: Aggregated error
- IV: Scalable optimization

Highly-dynamic & Recency-sensitive Data

- News recommendation applications: a bipartite graph
- WeChat news recommendation network is highly dynamic
  - 81 articles and 1400 reading records per second
- The network is also recency-sensitive
  - >73% articles died less than 6 hours while no one read again
  - Obvious exponential decay for article duration length.







Xumin Chen, et al. Scalable Optimization for Embedding Highly-Dynamic and Recency-Sensitive Data. ***KDD***, 2018.(Applied)

Limited resources

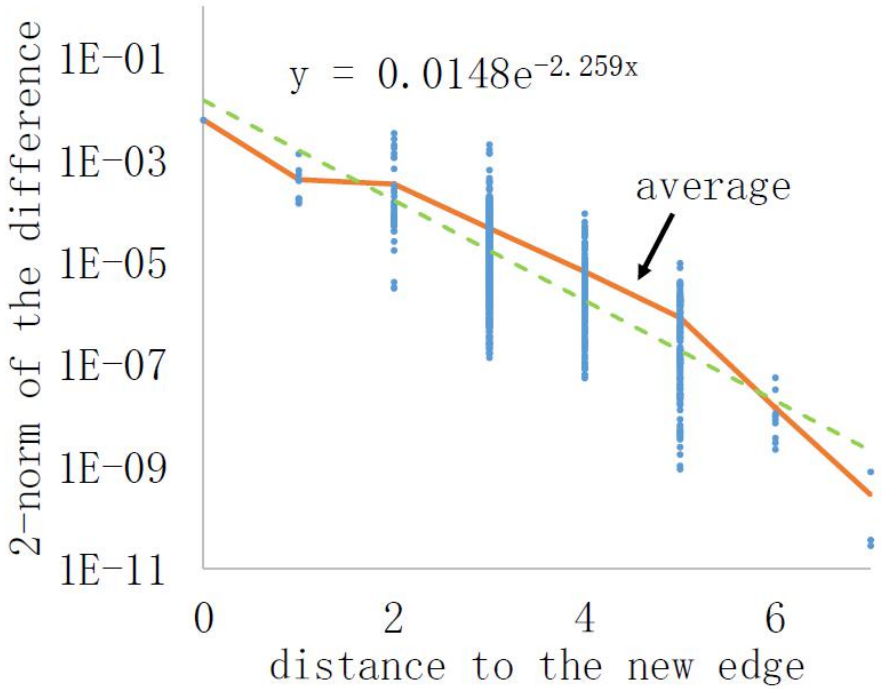
- We cannot guarantee convergence in-between every two timestamps.
- Just do it.

- How to do better?
- Non-uniform resource allocation.
- New edges and nodes worth more resources.

Xumin Chen, et al. Scalable Optimization for Embedding Highly-Dynamic and Recency-Sensitive Data. ***KDD***, 2018.(Applied)

Diffused SGD: Step-wise Weight Diffusion Mechanism

- The Change of a node embedding vector depends on its distance to the changed edge.
- Diffuse across **training steps**
- For step r, if edge (i, j) is chosen by stochastic method



For edge (i, j), we have

$$p_{i,j}(r) \leftarrow \tau_e\left(i, p_{i,j}(r-1)\right);$$

for $(i, k) \in \mathbb{E} \wedge k \neq j$, we use

$$p_{i,k}(r) \leftarrow p_{i,k}(r-1) + \tau_n\left(i, p_{i,j}(r-1)\right);$$

and for other edges $(l, k) \in \mathbb{E} \wedge l \neq i$,

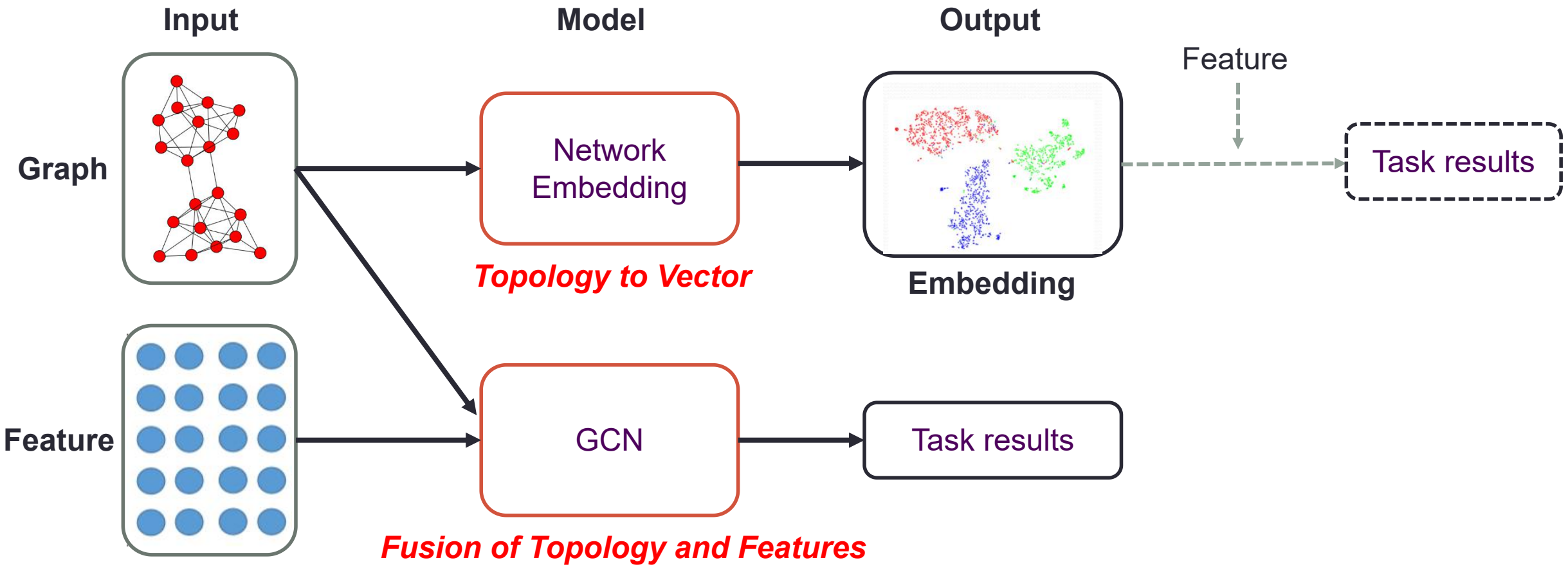
$$p_{l,k}(r) \leftarrow p_{l,k}(r-1);$$

Xumin Chen, et al. Scalable Optimization for Embedding Highly-Dynamic and Recency-Sensitive Data. ***KDD***, 2018.(Applied)

Section Summary

- I : Out-of-sample nodes
  - DepthLGP = Non-parametric GP + DNN
- II : Incremental edges
  - DHPE: Generalized Eigen Perturbation
- III: Aggregated error
  - TIMERS: A theoretically guaranteed SVD restart strategy
- IV: Scalable optimization
  - D-SGD: A iteration-wise weighted SGD for highly dynamic data

From Network Embedding to GCN



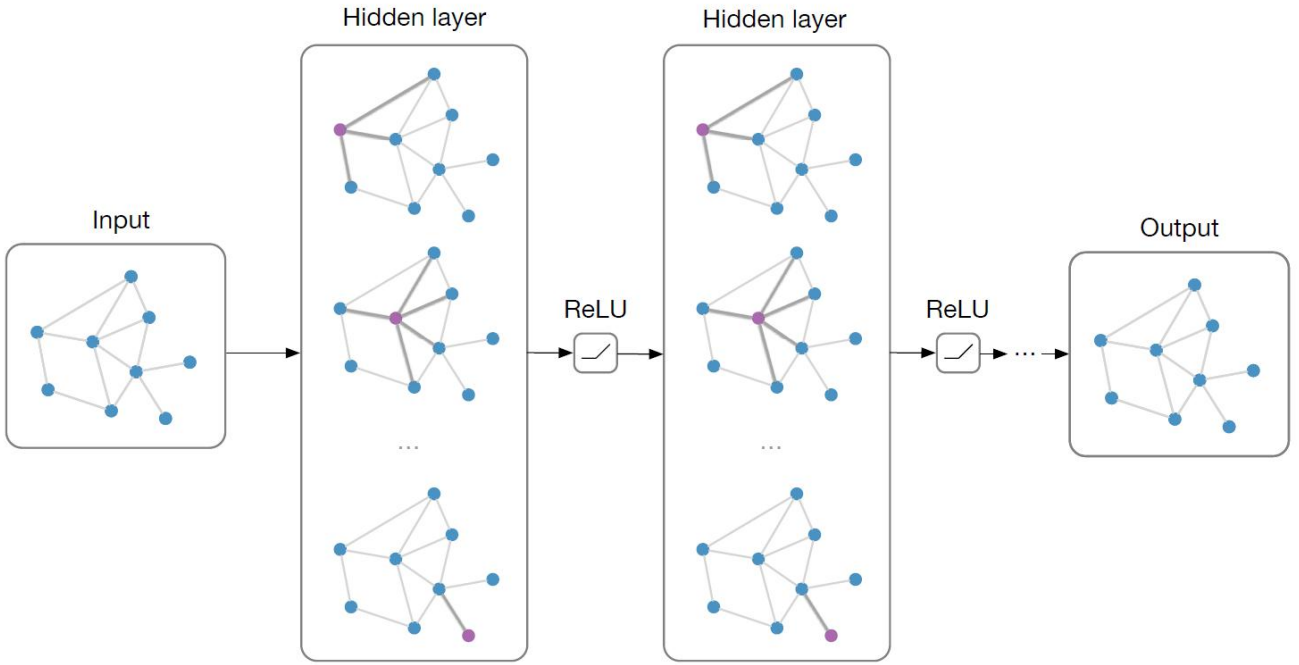
Unsupervised v.s. (Semi-)Supervised

Graph Convolutional Networks (GCN)

- Main idea: pass messages between pairs of nodes & agglomerate

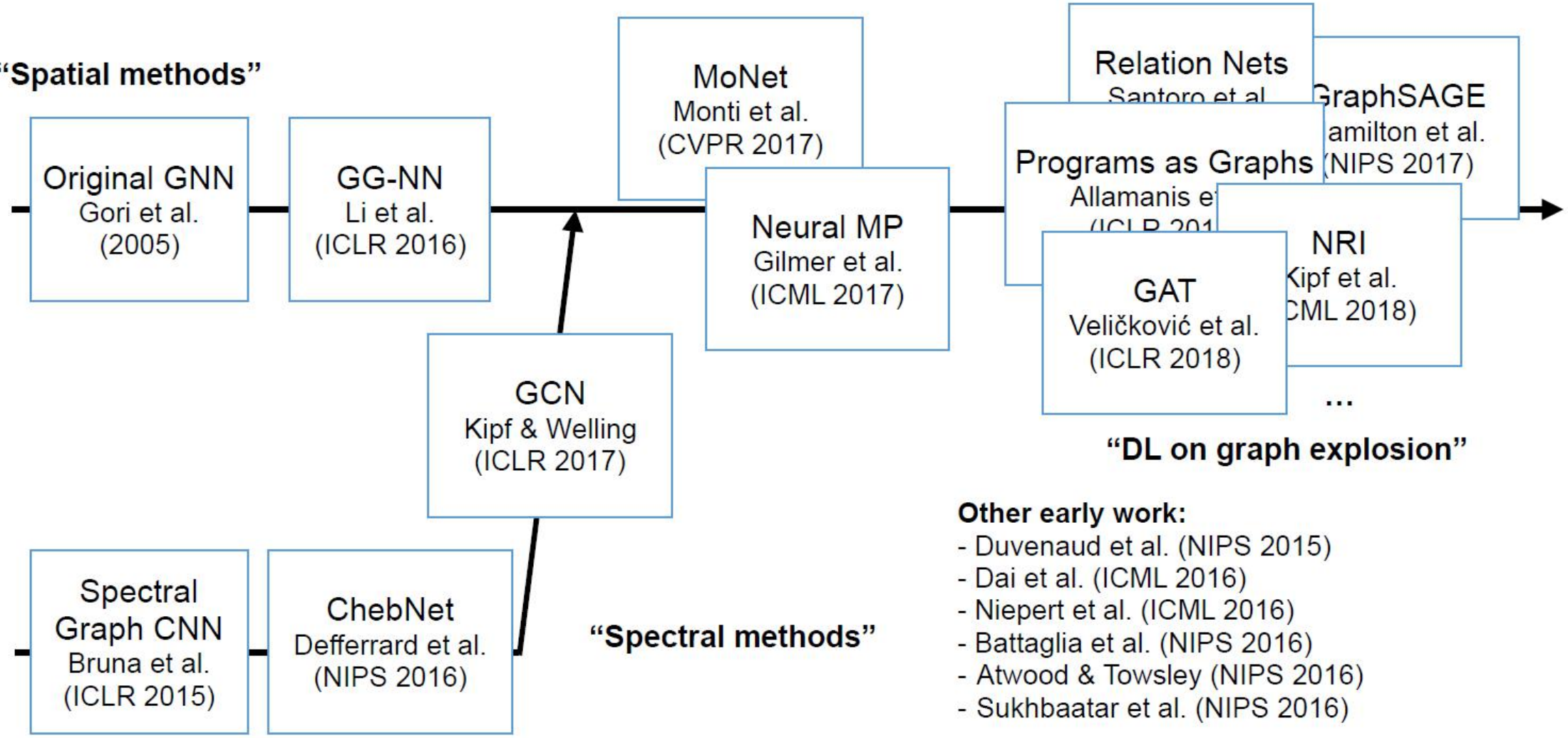
$$\mathbf{H}^{l+1} = \rho\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{l}\mathbf{\Theta}^{l}\right)$$

- Stacking multiple layers like standard CNNs:
  - State-of-the-art results on node classification



T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *ICLR, 2017.*

A brief history of GNNs



Other early work:

- Duvenaud et al. (NIPS 2015)
- Dai et al. (ICML 2016)
- Niepert et al. (ICML 2016)
- Battaglia et al. (NIPS 2016)
- Atwood & Towsley (NIPS 2016)
- Sukhbaatar et al. (NIPS 2016)

(slide inspired by Alexander Gaunt's talk on GNNs)

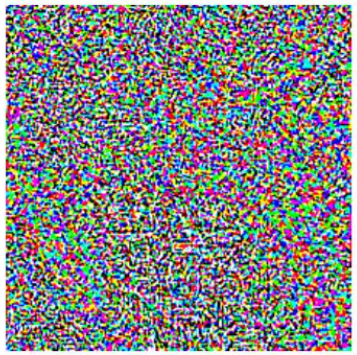
Technical challenges in real applications

Research → Application

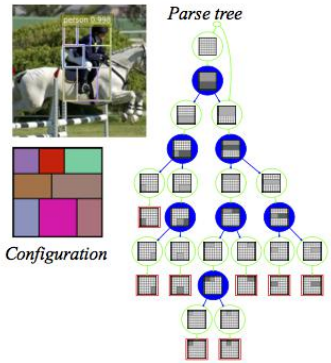
Robustness | Interpretability | Applicability

Hot directions in computer vision:

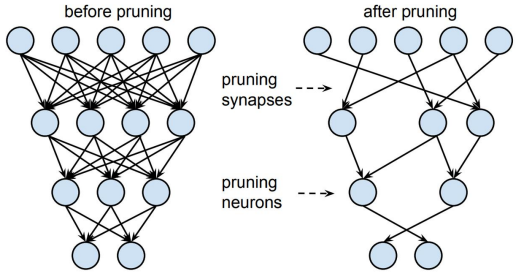
Adversarial



Explainable

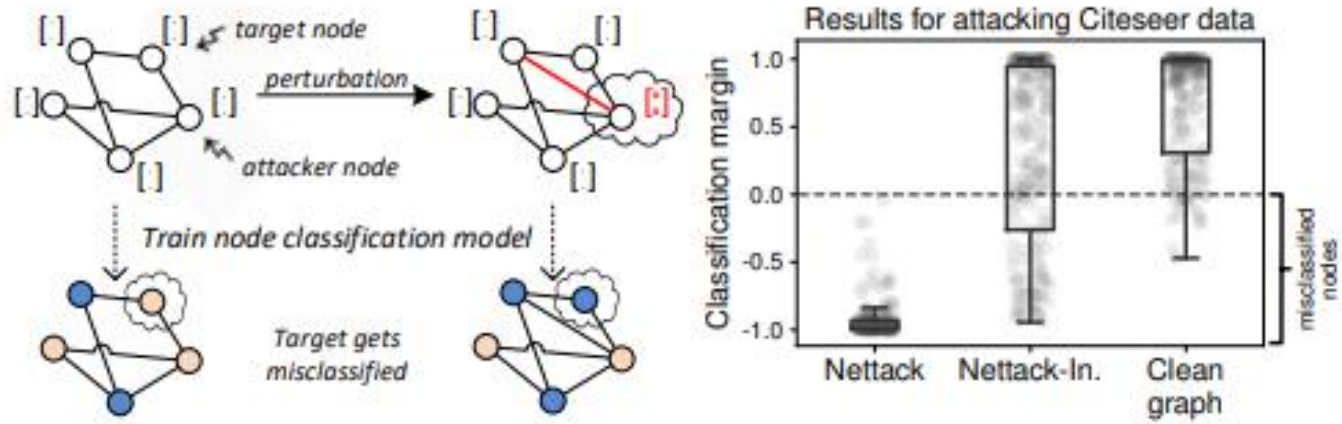


Scalable



Robustness in network embedding

- **Adversarial attacks**
  - small perturbations in graph structures and node attributes
  - great challenges for applying GCNs to node classification

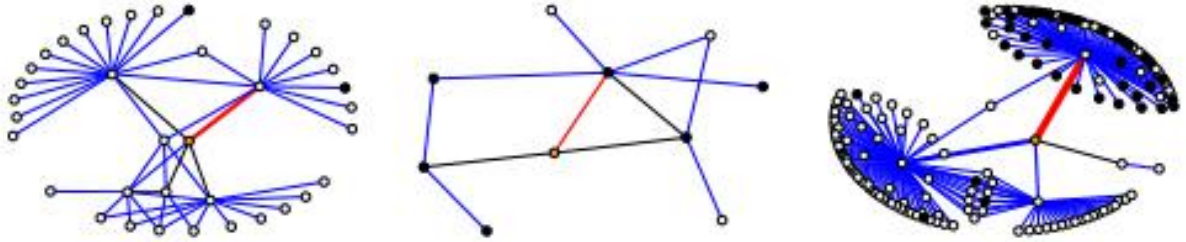


Adversarial Attacks on GCNs

- Categories
  - Targeted VS Non-targeted
    - Targeted: the attacker focus on misclassifying some target nodes
    - Non-targeted: the attacker aims to reduce the overall model performance
  - Direct vs Influence
    - Direct: the attacker can directly manipulate the edges or features of the target nodes
    - Influence: the attacker can only manipulate other nodes except the targets
- How to enhance the robustness of GCNs against adversarial attacks?

Robust Graph Convolutional Networks

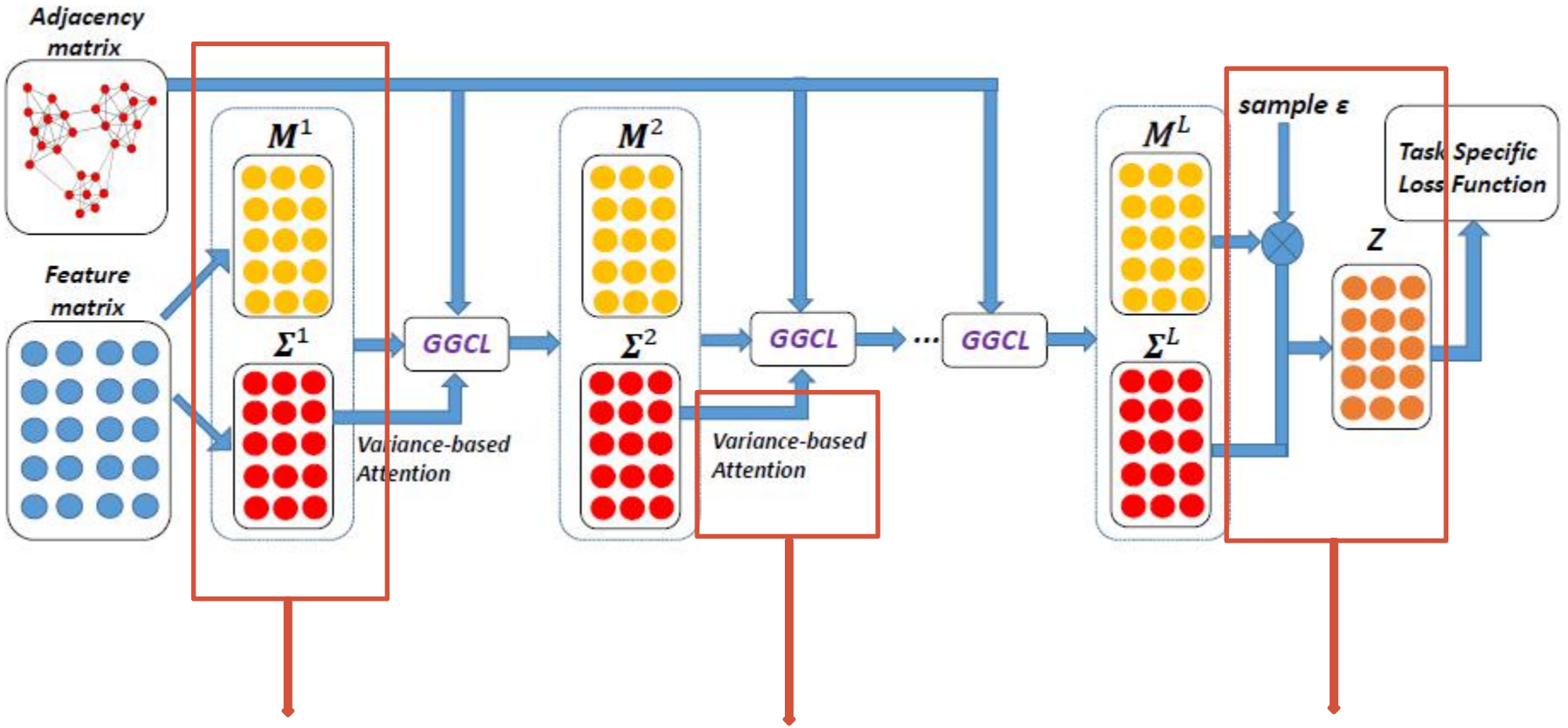
- Adversarial attacks in node classification
  - Connect nodes from different communities to confuse the classifier



- Distribution V.S. plain vectors
  - Plain vectors cannot adapt to such changes
  - Variances can help to absorb the effects of adversarial changes
  - Gaussian distributions -> Hidden representations of nodes

Dingyuan Zhu, Ziwei Zhang, Peng Cui, Wenwu Zhu. Robust Graph Convolutional Networks Against Adversarial Attacks. ***KDD***, 2019.

The Framework of RGCN



Gaussian Based hidden representations:

Variance terms absorb the effects of adversarial attacks

Attention mechanism:

Remedy the propagation of adversarial attacks

Sampling process:

Explicitly considers mathematical relevance between means and variances

Dingyuan Zhu, Ziwei Zhang, Peng Cui, Wenwu Zhu. Robust Graph Convolutional Networks Against Adversarial Attacks. ***KDD***, 2019.

Experimental Results

☐ Node Classification on Clean Datasets

	Cora	Citeseer	Pubmed
GCN	81.5	70.9	79.0
GAT	83.0	72.5	79.0
RGCN	**83.1**	71.3	**79.2**

☐ Against Non-targeted Adversarial Attacks

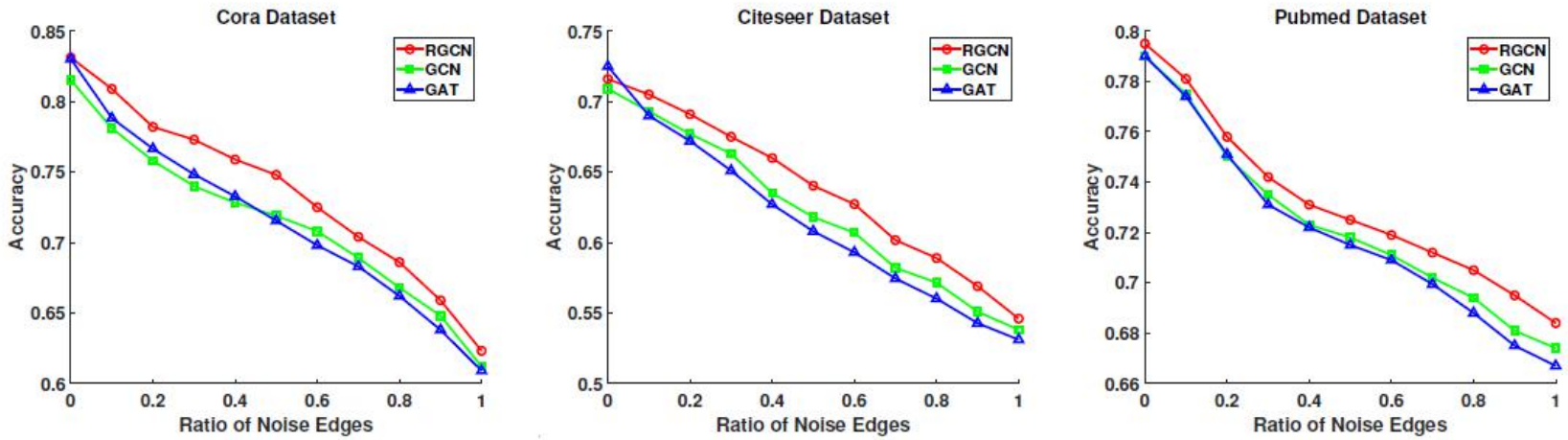
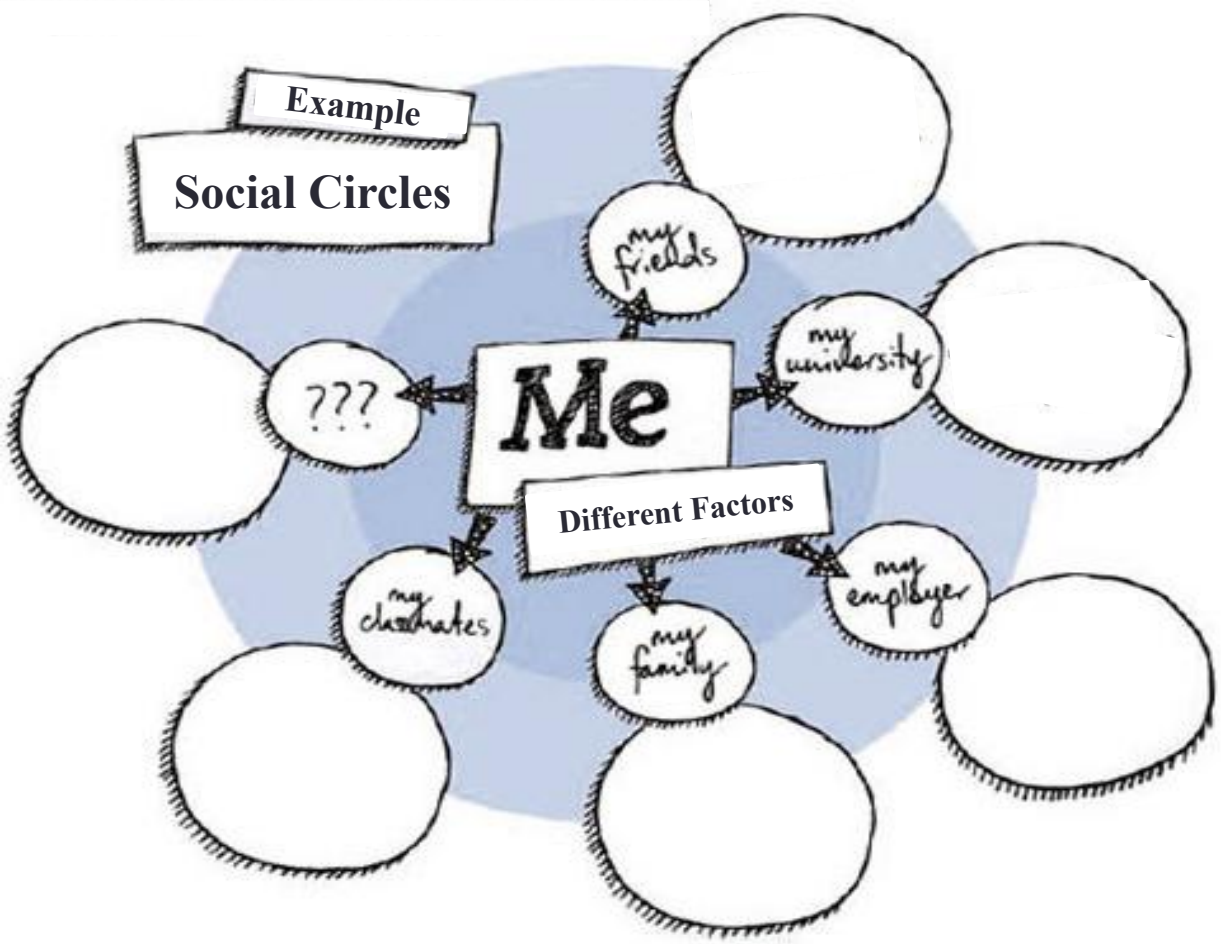


Figure 2: Results of different methods when adopting Random Attack as the attack method.

Dingyuan Zhu, Ziwei Zhang, Peng Cui, Wenwu Zhu. Robust Graph Convolutional Networks Against Adversarial Attacks. ***KDD***, 2019.

Interpretability of network embedding

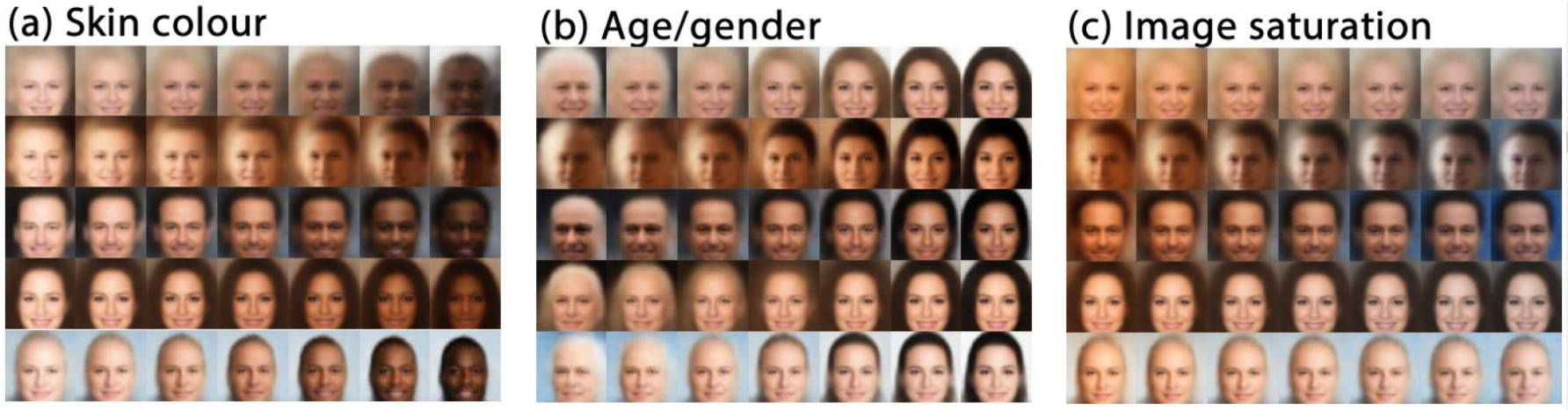
- **A real-world graph is typically formed due to *many* latent factors.**



- Existing GNNs/GCNs:
  - A holistic approach, that takes in the *whole* neighborhood to produce a *single* node representation.
- We suggest:
  - To disentangle the latent factors.
    (By segmenting the heterogeneous parts, and learning multiple factor-specific representations for a node.)
  - Robustness (e.g., not overreact to an irrelevant factor) & Interpretability.

Disentangled Representation Learning

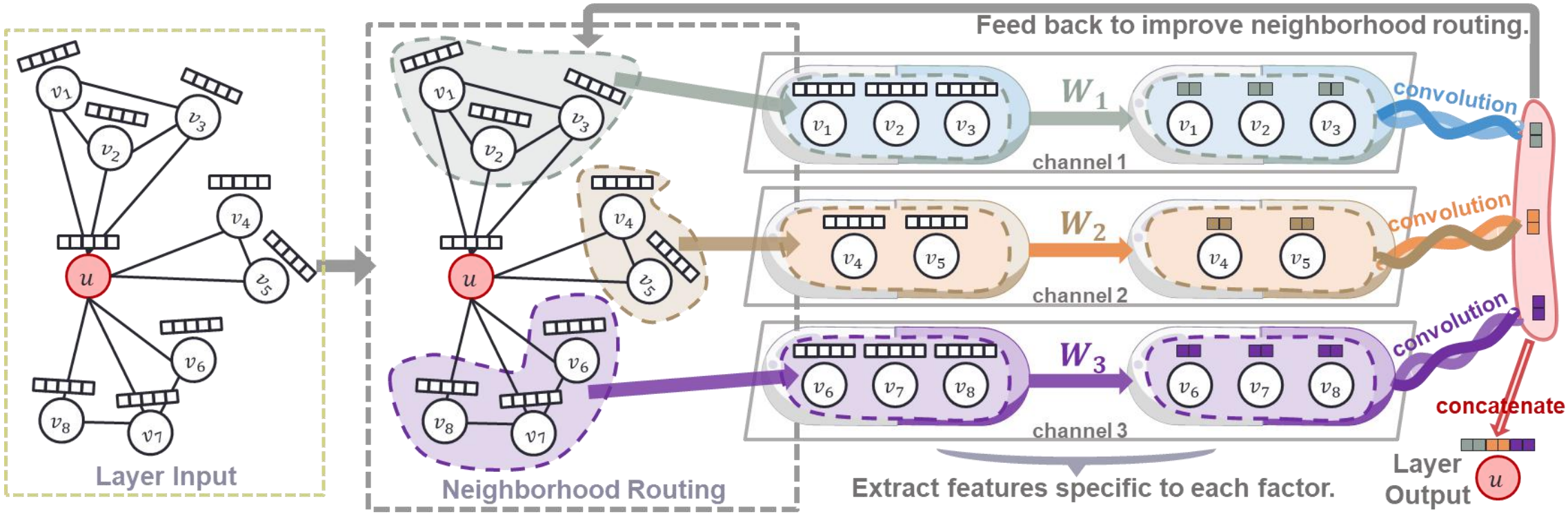
- That is, we aim to learn disentangled node representation,
  - A representation that contains independent components, that describes different aspects (caused by different latent factors) of the observation.
- The topic is well studied in the field of computer vision.
  - But largely unexplored in the literature of GNNs.



Example: Three dimensions that are related skin color, age/gender, and saturation, respectively.

Method Overview

- We present DisenGCN, the *disentangled* graph convolutional network.
  - DisenConv, a disentangled multichannel convolutional layer (figure below).
  - Each channel convolutes features related with a single latent factor.



Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, Wenwu Zhu. Disentangled Graph Convolutional Networks. ***ICML***, 2019.

Neighborhood Routing: Hypothesis I

- A neighbor is *patched to channel k* (for further in-channel graph convolution), if the edge between the neighbor and the center node is *caused by factor k*.
- But the actual causes are unknown. *Neighborhood routing* is therefore proposed to infer the latent causes, based on two hypothesis.

- The first is analogous to the second-order proximity.

> **Hypothesis 1.** Factor k is likely to be the reason why node u connects with a certain subset of its neighbors, if the subset is large and the neighbors in the subset are similar w.r.t. aspect k, i.e., they form a cluster in the k^{th} subspace.

- It inspires us to search for the biggest cluster in each of the K subspaces.

Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, Wenwu Zhu. Disentangled Graph Convolutional Networks. ***ICML***, 2019.

Neighborhood Routing: Hypothesis II

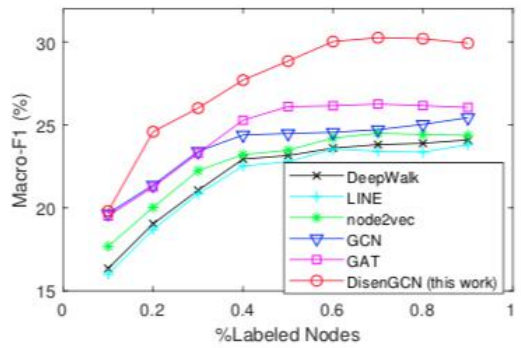
- The second hypothesis is analogous to the first-order proximity.

> **Hypothesis 2.** Factor k is likely to be the reason why node u and neighbor v are connected, if the two are similar in terms of aspect k.

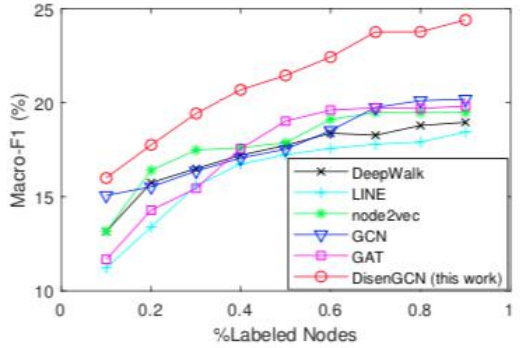
- Hypothesis 2 is not robust if either $\boldsymbol{x}_u$ or $\boldsymbol{x}_v$ misses features about aspect k, and therefore must be combined with Hypothesis 1. But it can provide a fast guess.

Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, Wenwu Zhu. Disentangled Graph Convolutional Networks. ***ICML***, 2019.

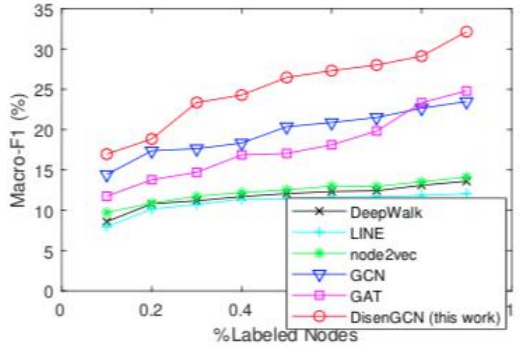
Results: Multi-label Classification



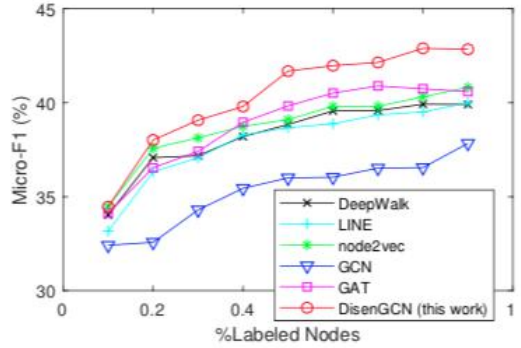
(a) Macro-F1(%), BlogCatalog.



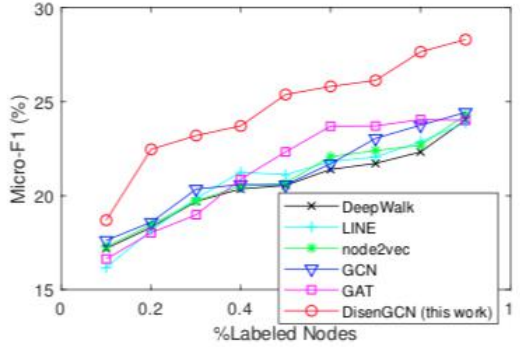
(c) Macro-F1(%), PPI.



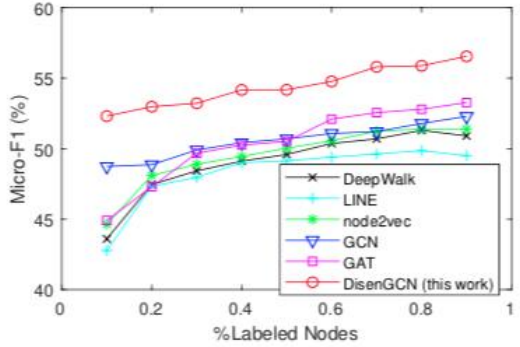
(e) Macro-F1(%), POS.



(b) Micro-F1(%), BlogCatalog.



(d) Micro-F1(%), PPI.



(f) Micro-F1(%), POS.

Figure 2. Macro-F1 and Micro-F1 scores on the multi-label classification tasks. Our approach consistently outperforms the best performing baselines by a large margin, reaching 10% to 20% relative improvement in most cases.

Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, Wenwu Zhu. Disentangled Graph Convolutional Networks. ***ICML***, 2019.

Results: On Synthetic Graphs

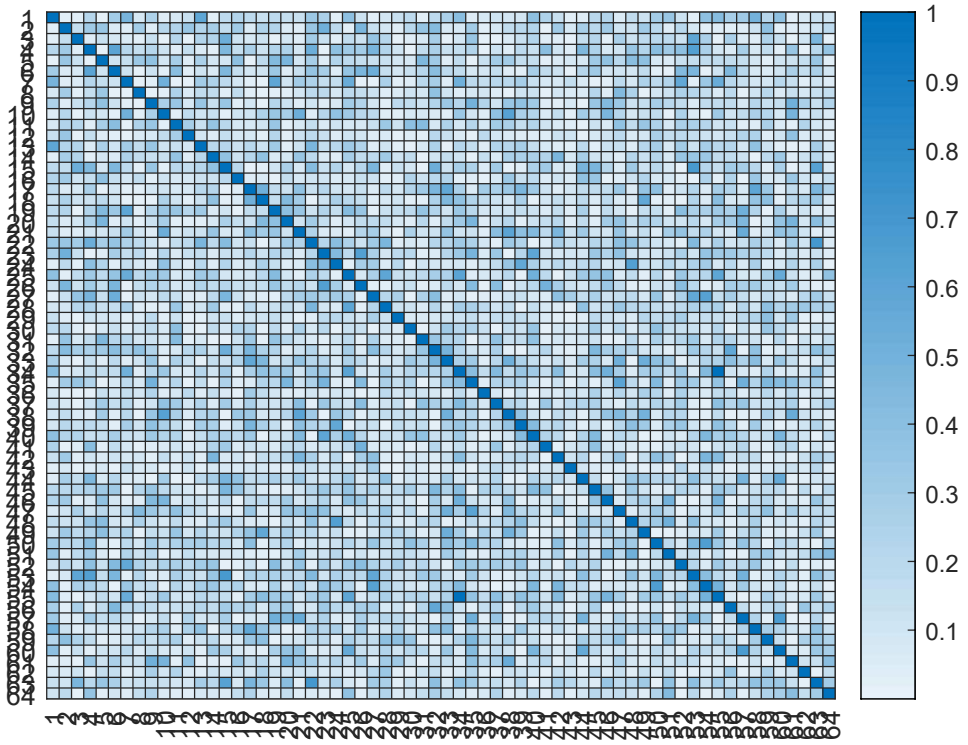
Table 3. Micro-F1 scores on synthetic graphs generated with different numbers of latent factors.

Method	Number of latent factors						
	4	6	8	10	12	14	16
GCN	78.78 ± 1.52	65.73 ± 1.94	46.55 ± 1.55	37.37 ± 1.52	24.49 ± 1.03	18.14 ± 1.50	16.43 ± 0.92
GAT	83.77 ± 2.32	60.89 ± 3.75	45.88 ± 3.79	36.72 ± 3.58	24.77 ± 3.47	20.89 ± 3.57	19.53 ± 3.97
DisenGCN (this work)	**93.84** ± 1.12	**74.68** ± 1.92	**54.57** ± 1.79	**43.96** ± 1.45	**28.17** ± 1.22	**23.57** ± 1.28	**21.99** ± 1.34
Relative improvement	+12.02%	+13.62%	+17.23%	+17.63%	+13.73%	+12.83%	+12.6%

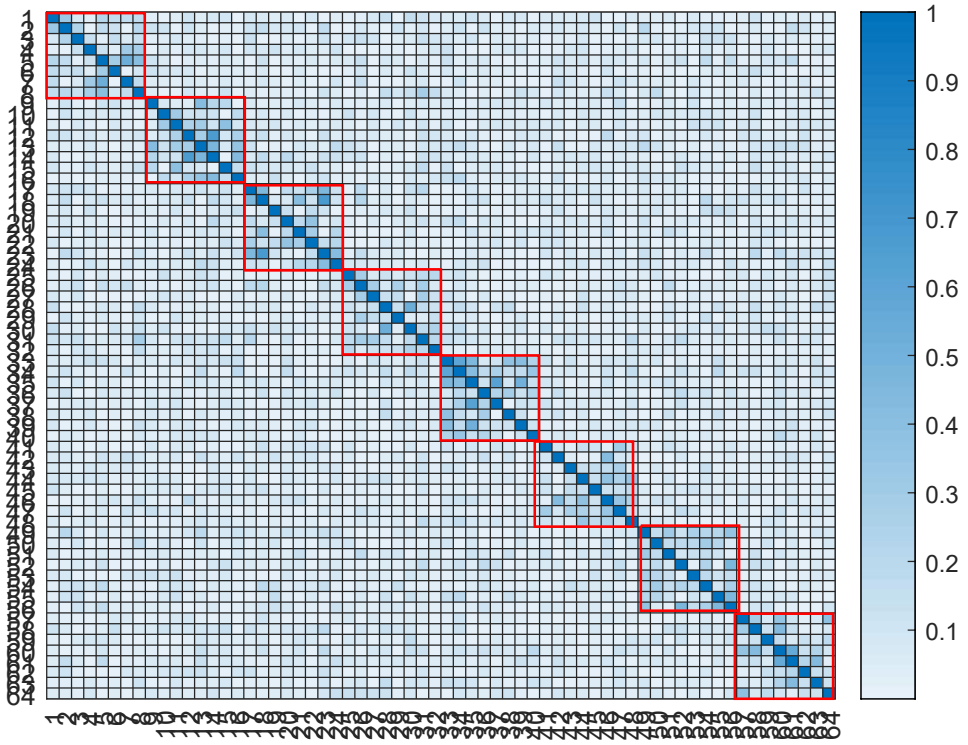
- Improvement is larger when #factors is relatively large (around 8).
- However, all methods are bad when #factors is extremely large.

Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, Wenwu Zhu. Disentangled Graph Convolutional Networks. ***ICML***, 2019.

Results: Correlations between the Neurons



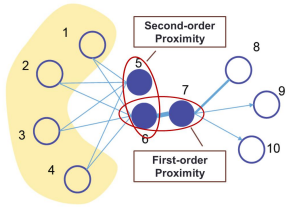
(a) GCN.



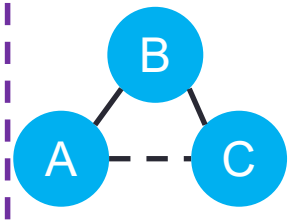
(b) DisenGCN (this work).

Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, Wenwu Zhu. Disentangled Graph Convolutional Networks. ***ICML***, 2019.

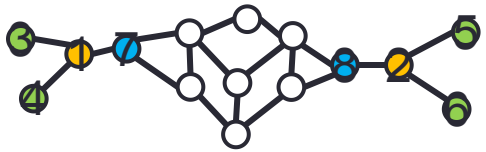
Applicability of network embedding and GCN



High-order



Transitivity



Global position

Various network properties

- ☐ Link Prediction
- ☐ Community Detection
- ☐ Node Classification
- ☐ Network Distance
- ☐ Node Importance
- ☐ …

Various applications

- Leading to a large number of hyperparameters
- Must be carefully tuned

AutoML

AutoML

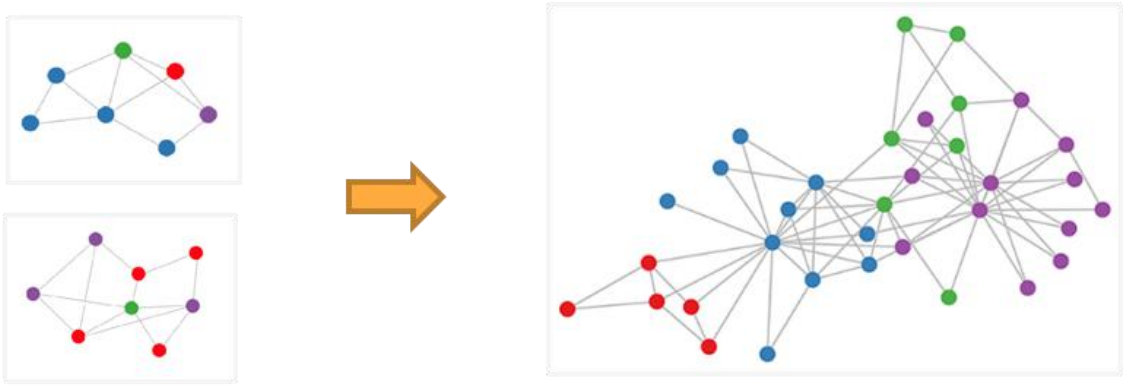
- Ease the adoption of machine learning and reduce the reliance on human experts
  - e.g., hyperparameter optimization
- Largely unexplored on network data

- Large scale issue:
  - Complexity of Network Embedding is usually at least $O(E)$
    - E is the number of edges (can be 10 billion)
  - Total complexity: $O(ET)$, T is the times searching for optimal hyperparameters

How to incorporate AutoML into massive network embedding efficiently?

AutoML for network embedding

- A straightforward way: configuration selection on sampled sub-networks



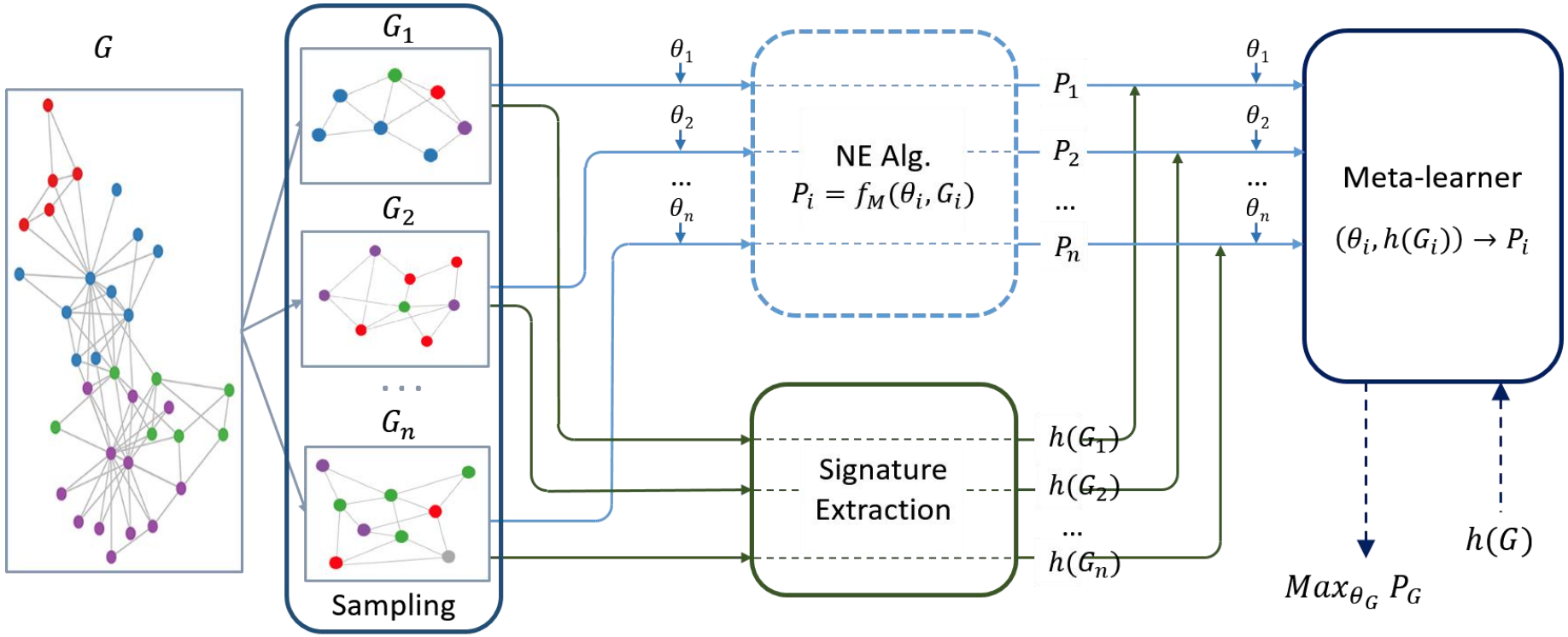
Sampled sub-network
Optimal configuration θ

Origin massive network

- Transferability
  - $\theta \neq$ optimal configuration on origin network
- Heterogeneity
  - several highly heterogeneous components => carefully designed sampling

Ke Tu, Jianxin Ma, Peng Cui, Jian Pei, Wenwu Zhu. AutoNE: Hyperparameter Optimization for Massive Network Embedding. ***KDD***, 2019.

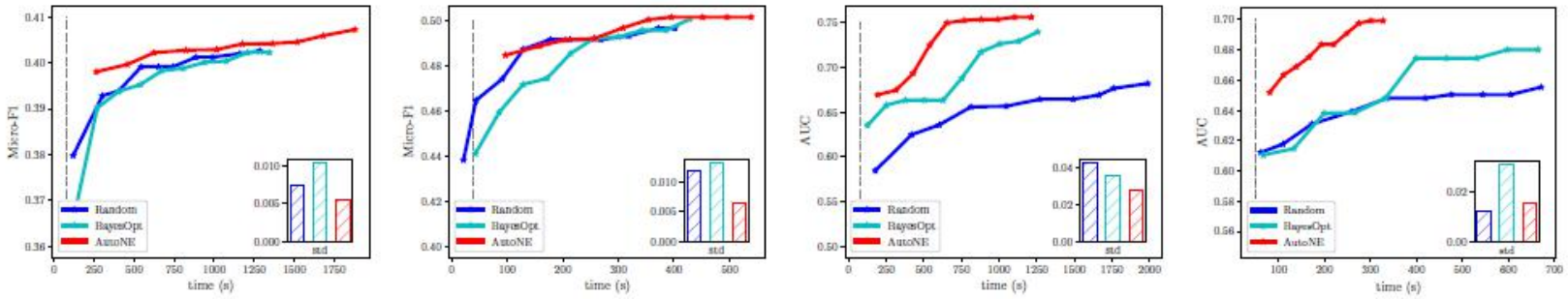
AutoNE



Transfer the knowledge about optimal hyperparameters from the sub-networks to the original massive network

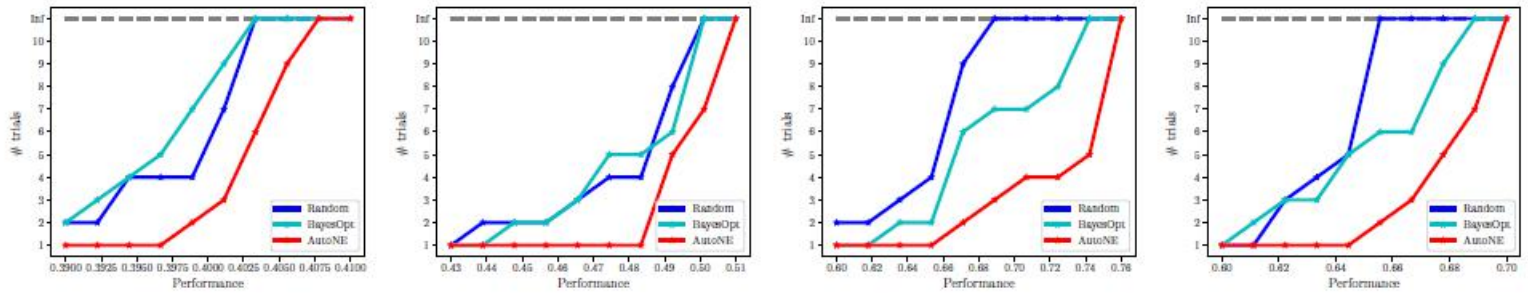
Ke Tu, Jianxin Ma, Peng Cui, Jian Pei, Wenwu Zhu. AutoNE: Hyperparameter Optimization for Massive Network Embedding. ***KDD***, 2019.

Experiment --- Sampling-Based NE



(a) Classification on BlogCatalog (b) Classification on Wikipedia (c) Link prediction on BlogCatalog (d) Link prediction on Wikipedia

The performance achieved within various time thresholds.



(a) Classification on BlogCatalog (b) Classification on Wikipedia (c) Link prediction on BlogCatalog (d) Link prediction on Wikipedia

The number of trials to reach a certain performance threshold

Ke Tu, Jianxin Ma, Peng Cui, Jian Pei, Wenwu Zhu. AutoNE: Hyperparameter Optimization for Massive Network Embedding. ***KDD***, 2019.

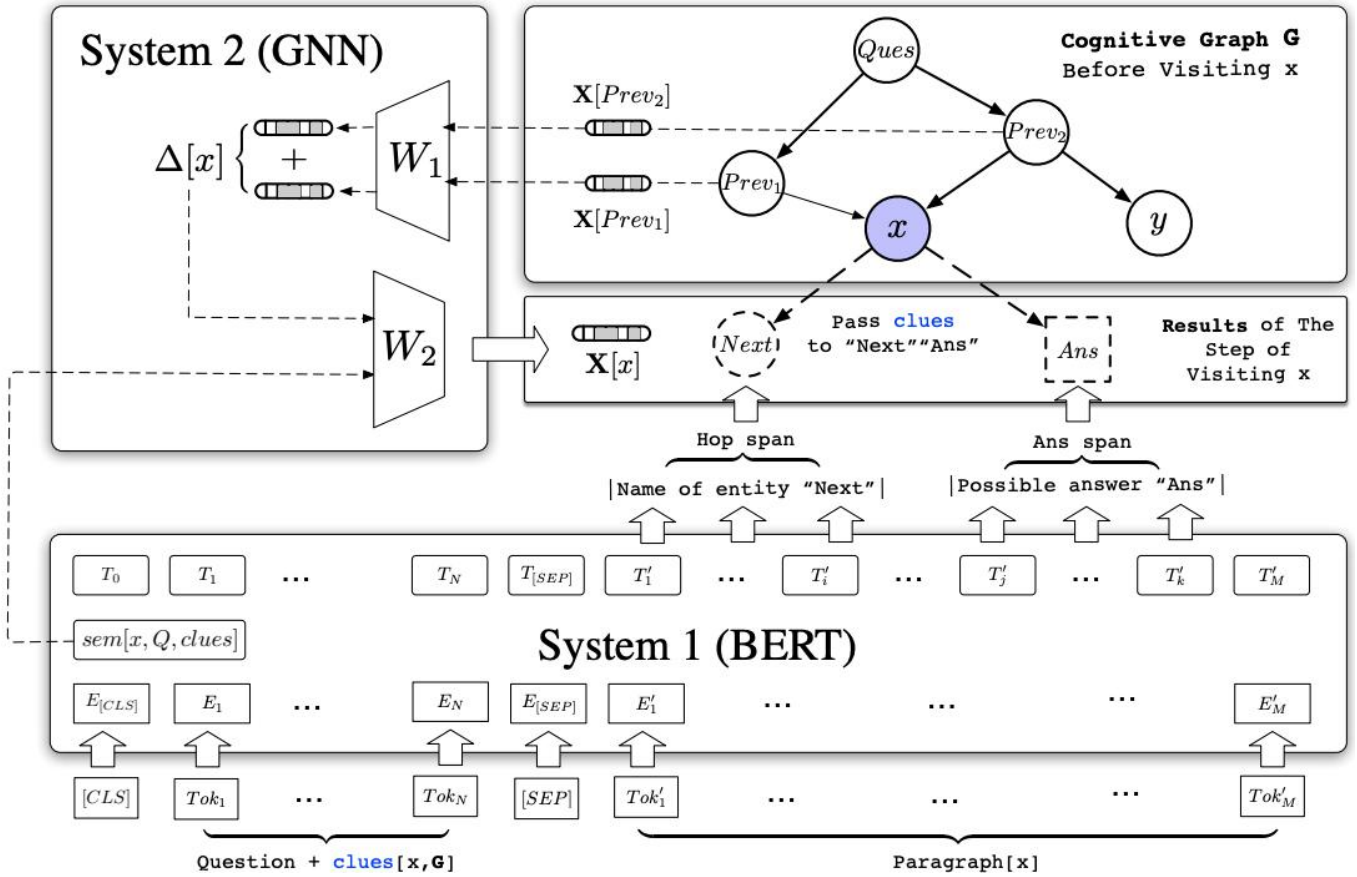
Network Embedding v.s. GCN

Graph convolutional network v.s. Network embedding

- In some sense, they are different.
- **Graphs** exist in ***mathematics***. (Data Structure)
  - Mathematical structures used to model pairwise relations between objects
- **Networks** exist in the ***real world***. (Data)
  - Social networks, logistic networks, biology networks, transaction networks, etc.
- A network can be represented by a graph.
- A dataset that is not a network can also be represented by a graph.

GCN for Natural Language Processing

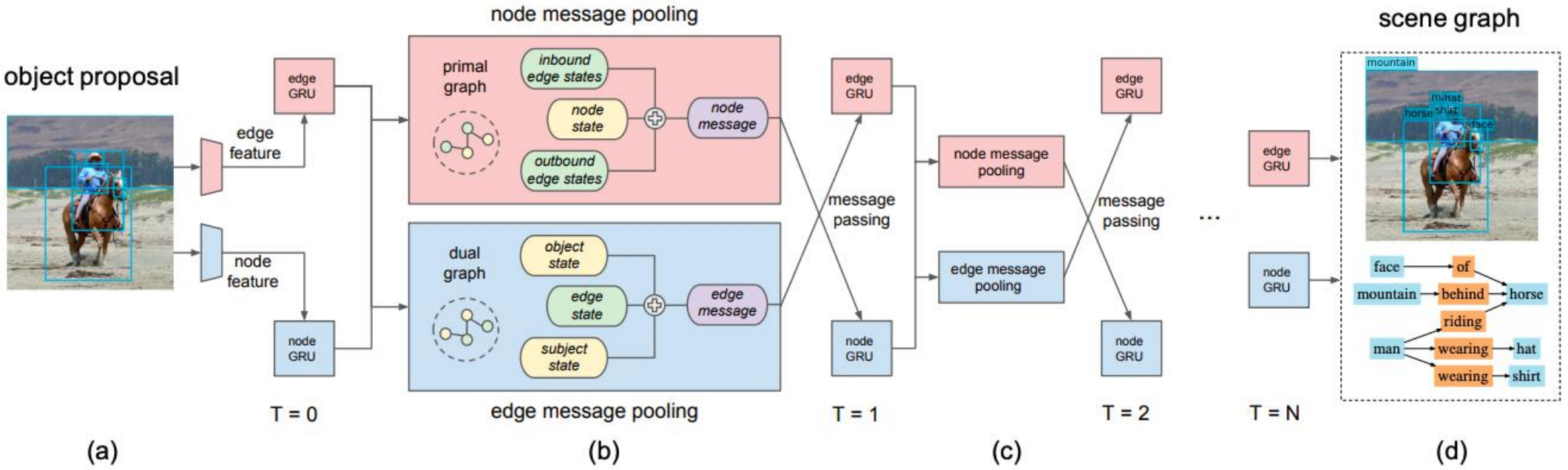
- Many papers on BERT + GNN.
- BERT is for retrieval.
  - It creates an initial graph of relevant entities and the initial evidence.
- GNN is for reasoning.
  - It collects evidence (i.e., old messages on the entities) and arrive at new conclusions (i.e., new messages on the entities), by passing the messages around and aggregating them.



Cognitive Graph for Multi-Hop Reading Comprehension at Scale. Ding et al., ACL 2019.
Dynamically Fused Graph Network for Multi-hop Reasoning. Xiao et al., ACL 2019.

GCN for Computer Vision

- A popular trend in CV is to construct a graph during the learning process.
  - To process multiple objects or parts in a scene, and to infer their relationships.
- Example: Scene graphs.



Scene Graph Generation by Iterative Message Passing. Xu et al., CVPR 2017.
Image Generation from Scene Graphs. Johnson et al., CVPR 2018.

GCN for Symbolic Reasoning

- We can view the process of symbolic reasoning as a directed acyclic graph.
- Many recent efforts use GNNs to perform symbolic reasoning.

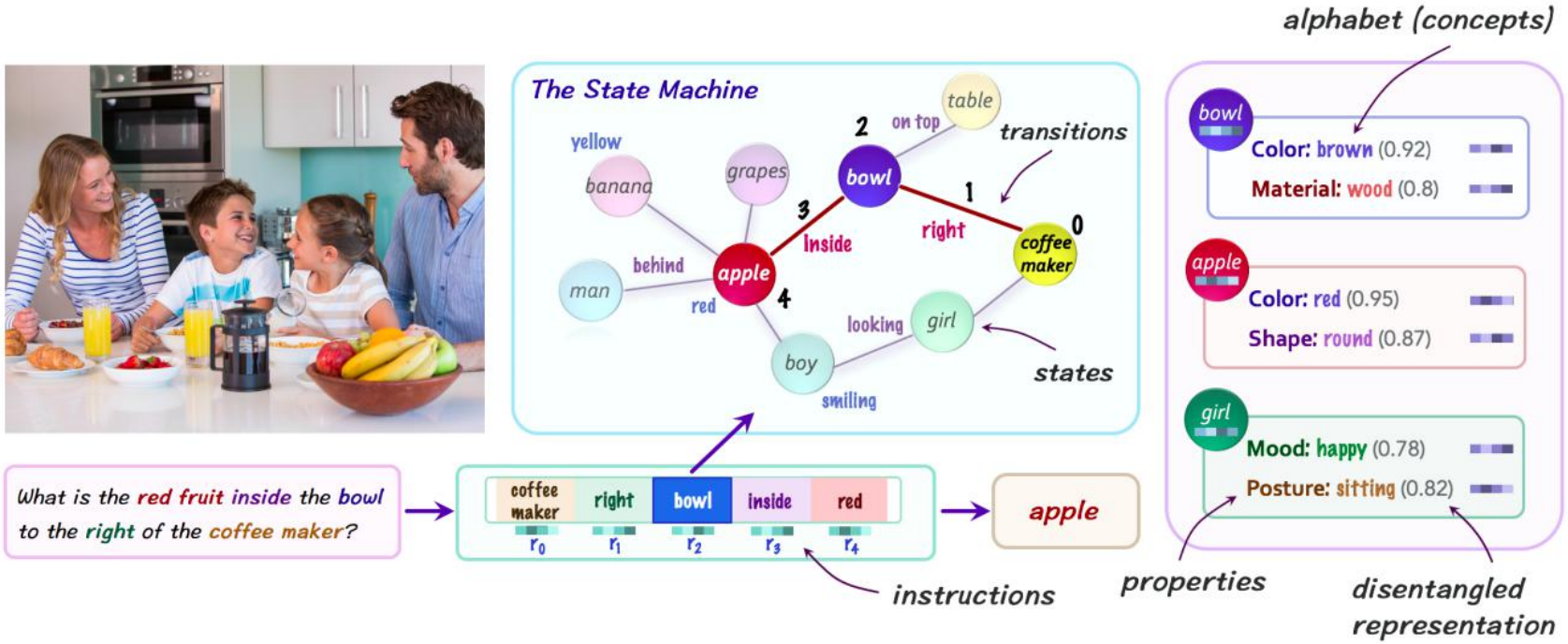
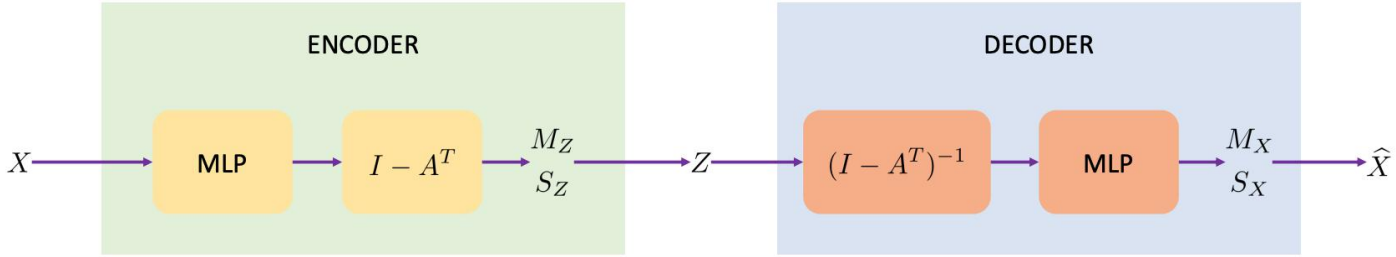


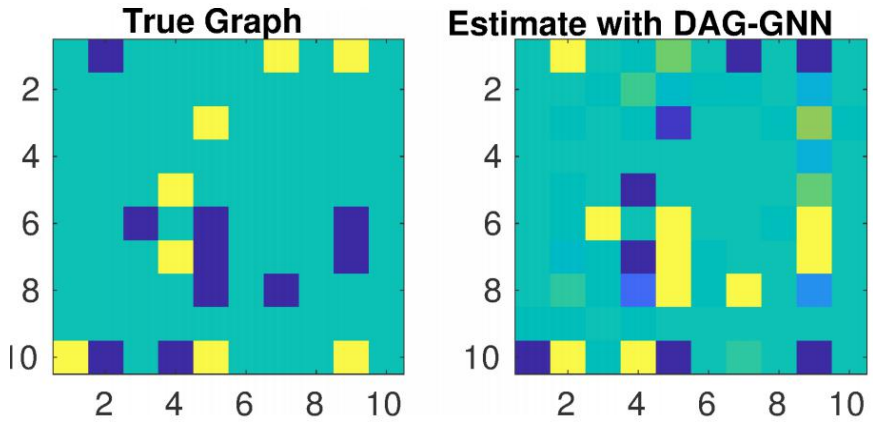
Figure 1: The Neural State Machine is a graph network that simulates the computation of an automaton.

Learning by Abstraction: The Neural State Machine. Hudson & Manning, 2019.
Can Graph Neural Networks Help Logic Reasoning? Zhang et al., 2019.
Symbolic Graph Reasoning Meets Convolutions. Liang et al., NeurIPS 2018.

GCN for Structural Equation Modeling

- Structural equation modeling, a form of causal modeling, tries to describe the relationships between the variables as a directed acyclic graph (DAG).
- GNN can be used to represent a nonlinear structural equation and help find the DAG, after treating the adjacency matrix as parameters.





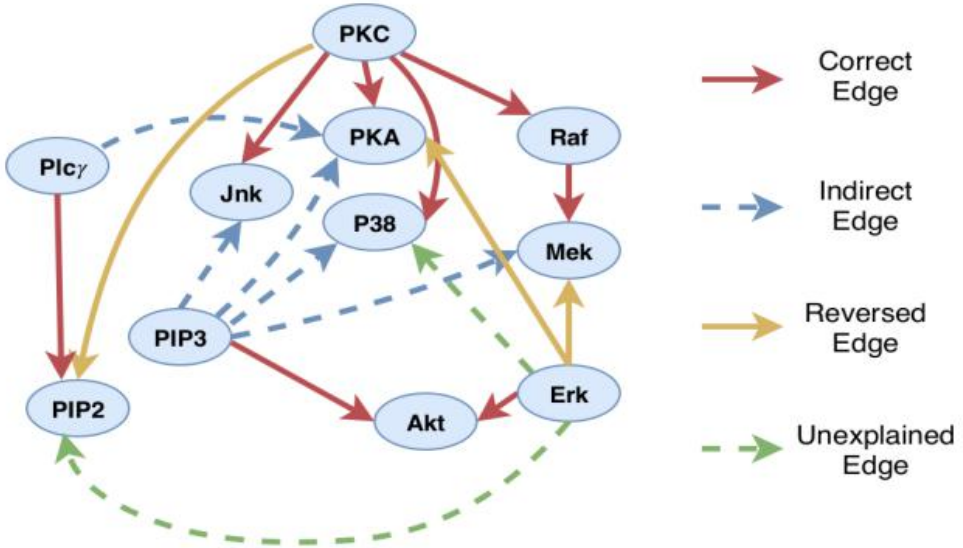
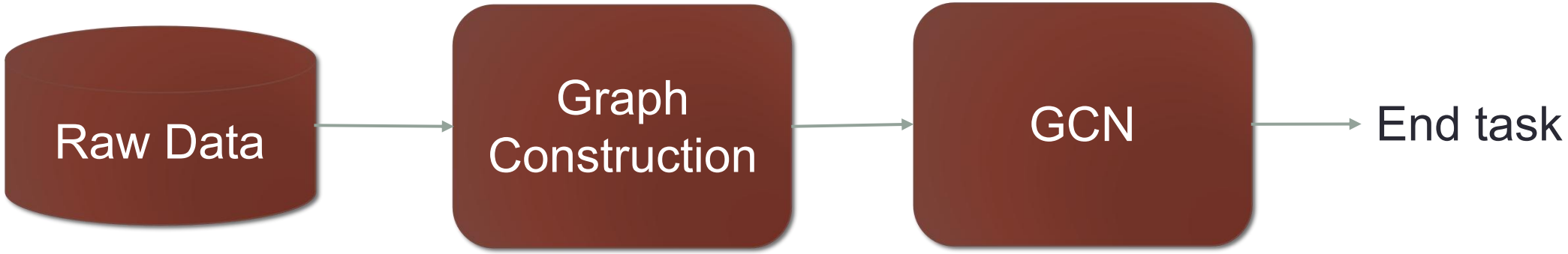


Figure 8. Estimate protein signaling network.

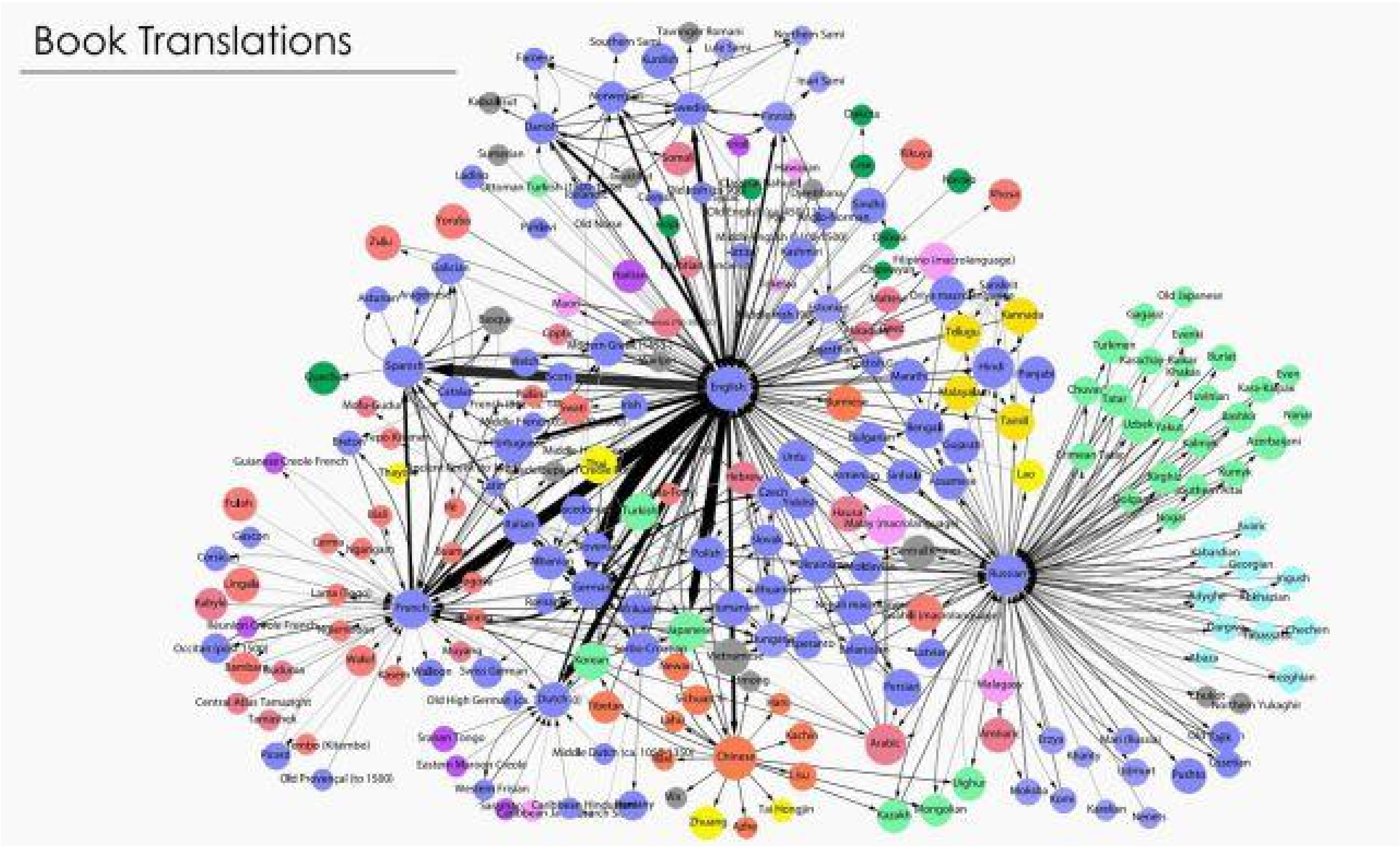
DAG-GNN: DAG Structure Learning with Graph Neural Networks. Yu et al., ICML 2019.

Pipeline for (most) GCN works



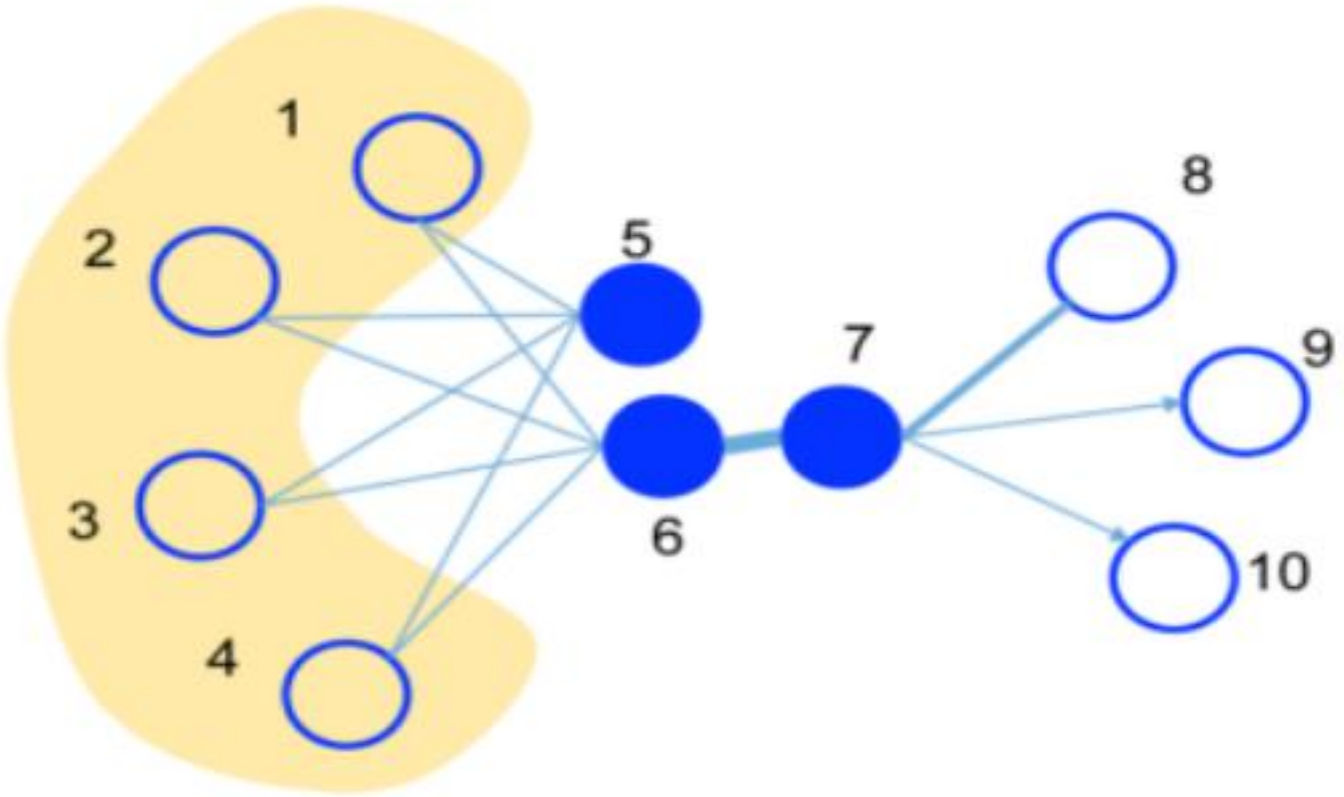
Network embedding: topology to vector

- **Co-occurrence (neighborhood)**



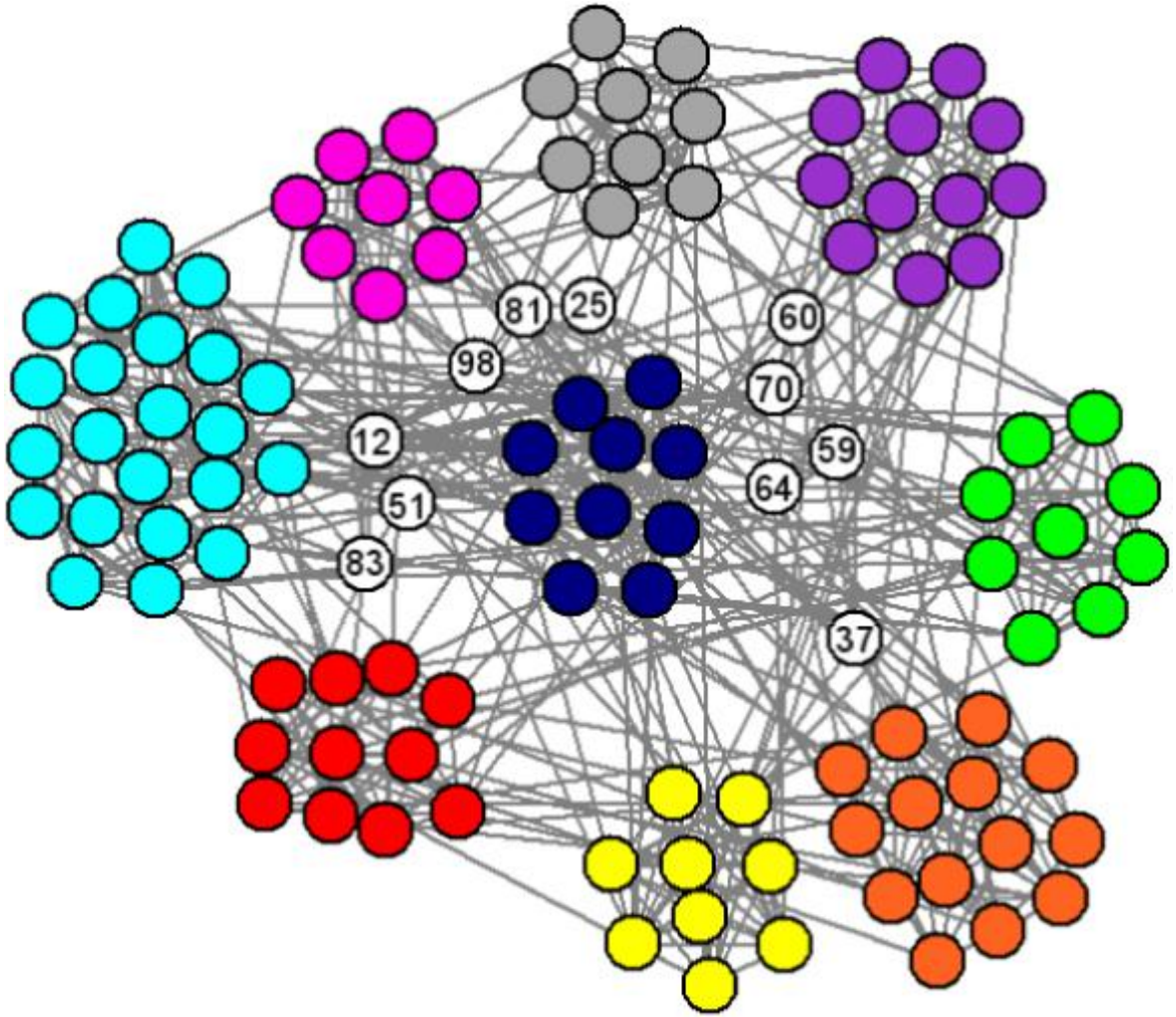
Network embedding: topology to vector

- **High-order proximities**



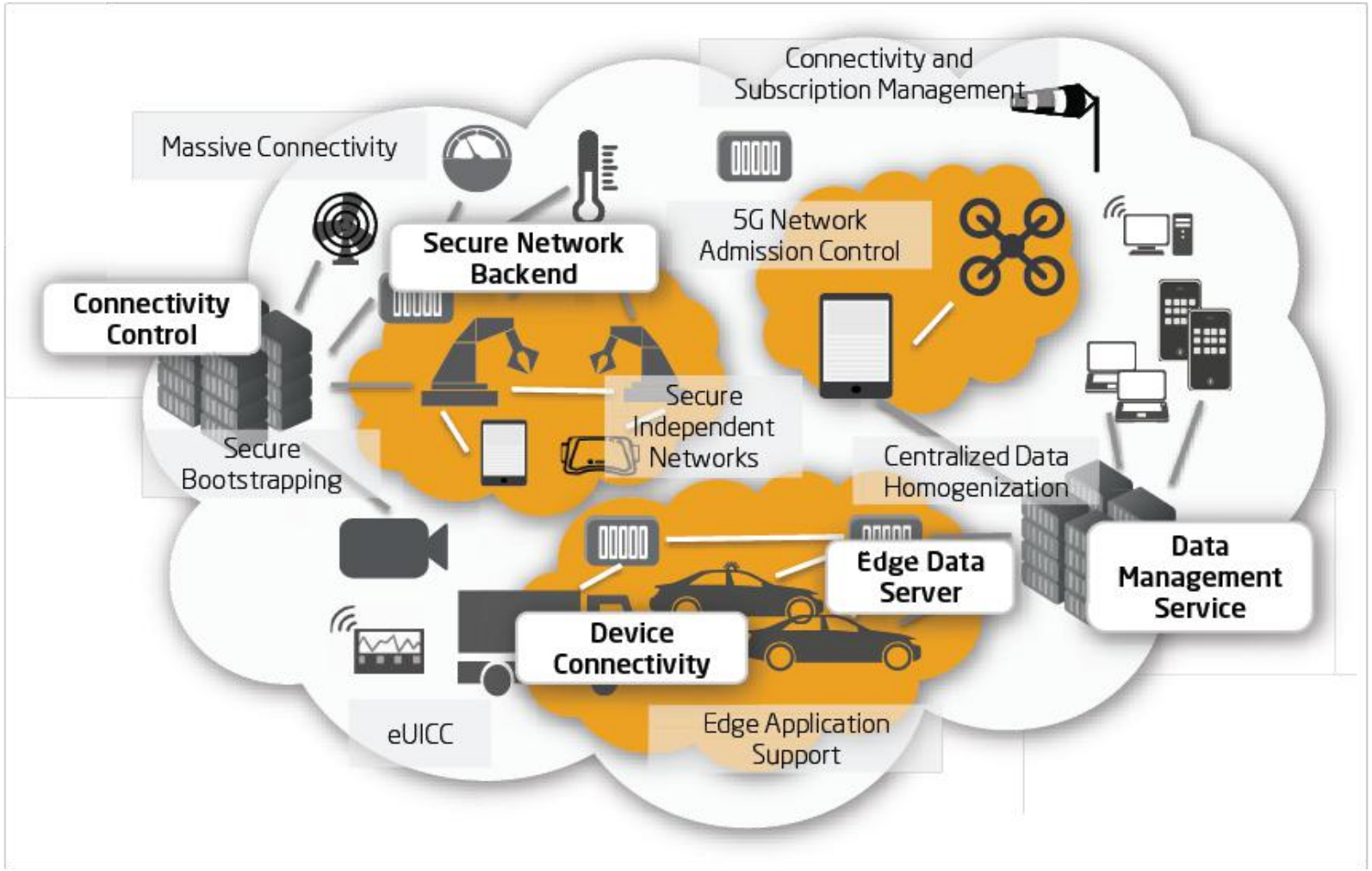
Network embedding: topology to vector

- **Communities**

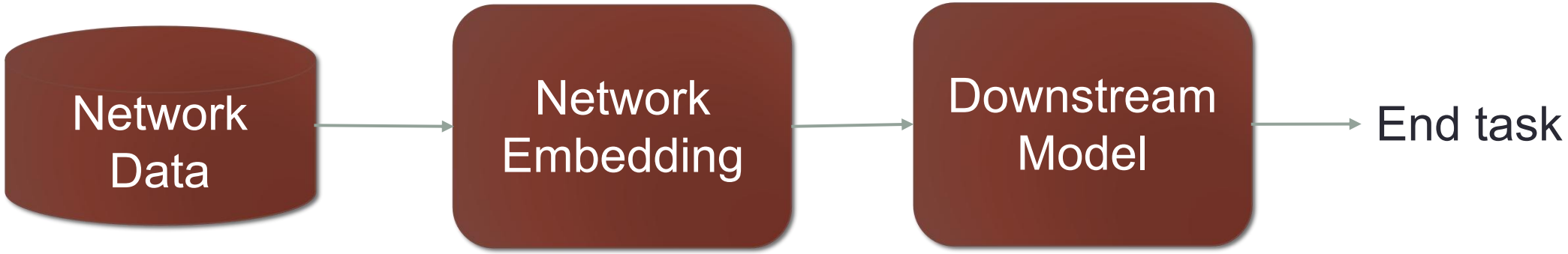


Network embedding: topology to vector

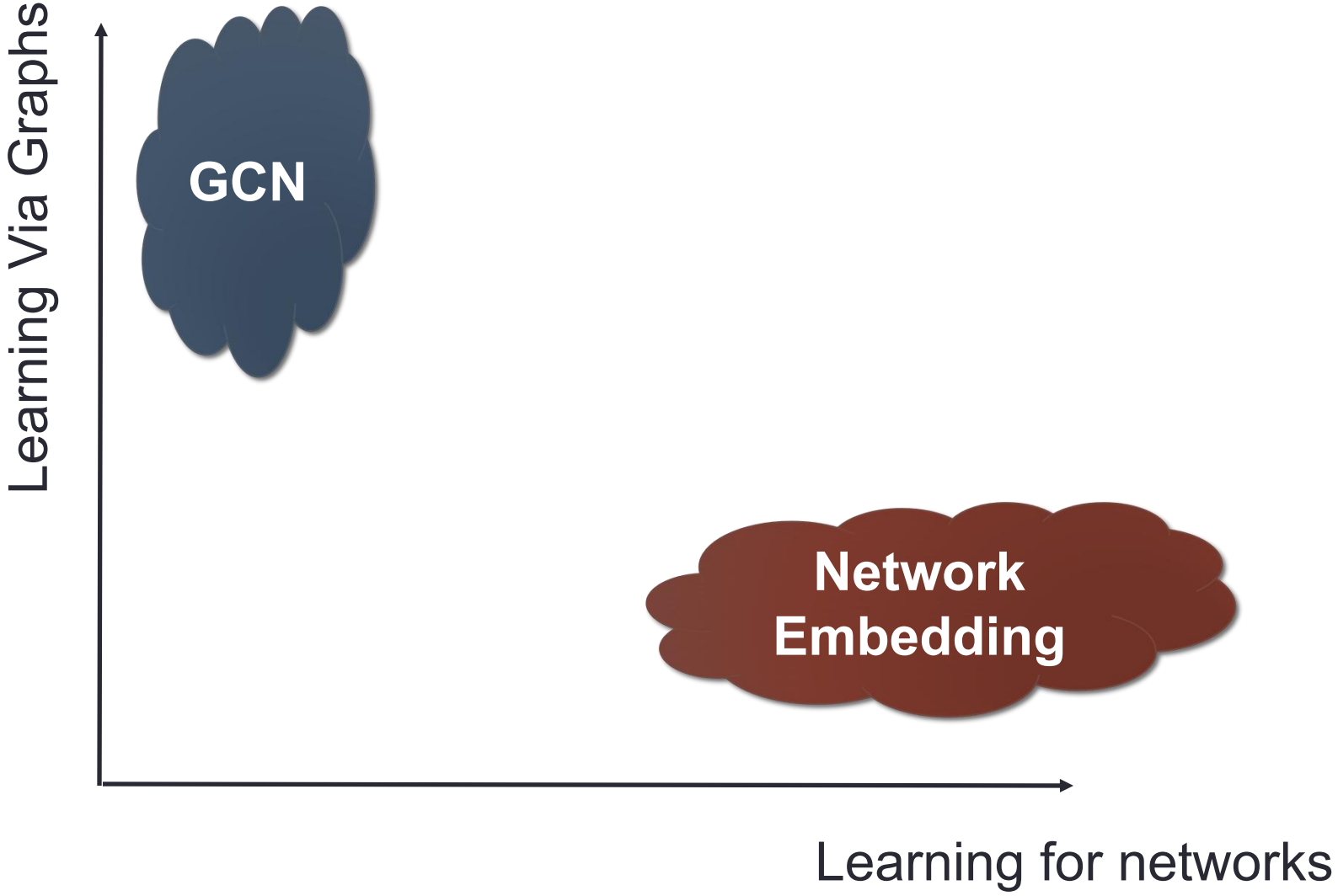
- **Heterogeneous networks**



Pipeline for (most) Network Embedding works



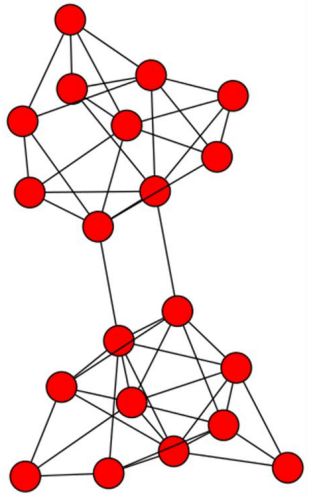
Learning for Networks vs. Learning via Graphs



The intrinsic problems NE is solving

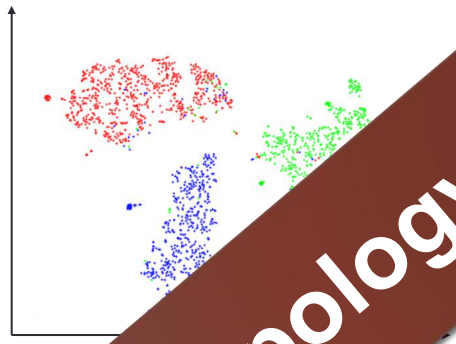
Reducing representation dimensionality while preserving necessary topological **structures** and **properties**.

- Nodes & Links
- Node Neighborhood
- Pair-wise Proximity
- Community
- Hyper Edges
- Global Structure



generate

embed



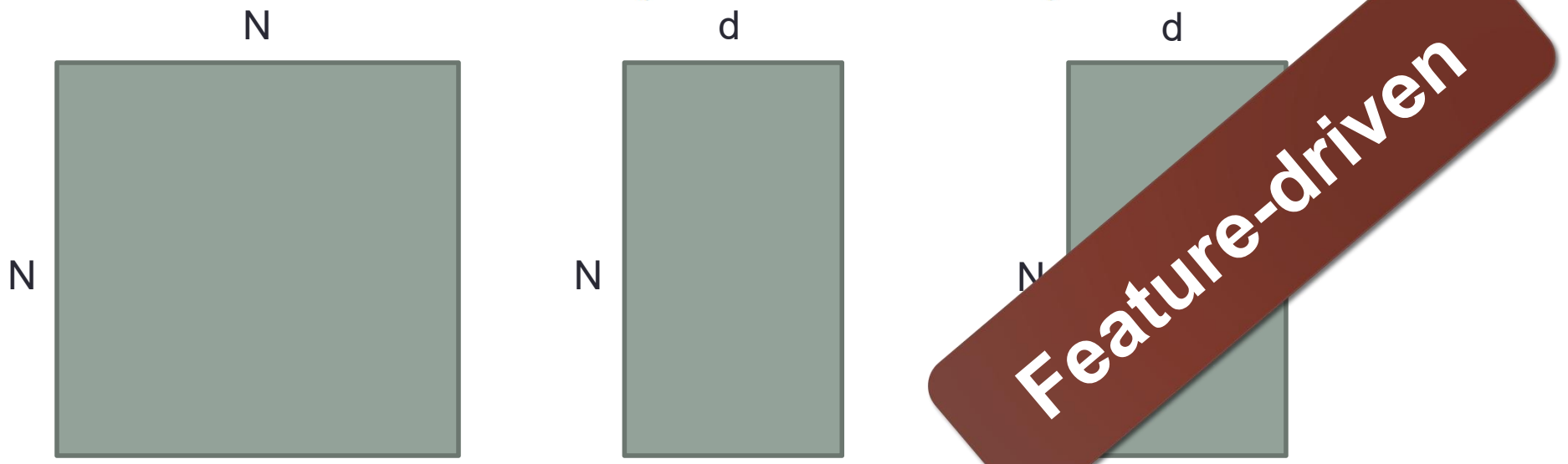
Topology-driven

- Non-transitivity
- [illegible]ansitivity
- [illegible]ncertainty
- Dynamic
- Heterogeneity
- Interpretability

The intrinsic problem GCN is solving

Fusing topology and features in the way of smoothing features with the assistance of topology.

$$\mathbf{H}^{l+1} = \rho\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{l}\mathbf{\Theta}^{l}\right)$$



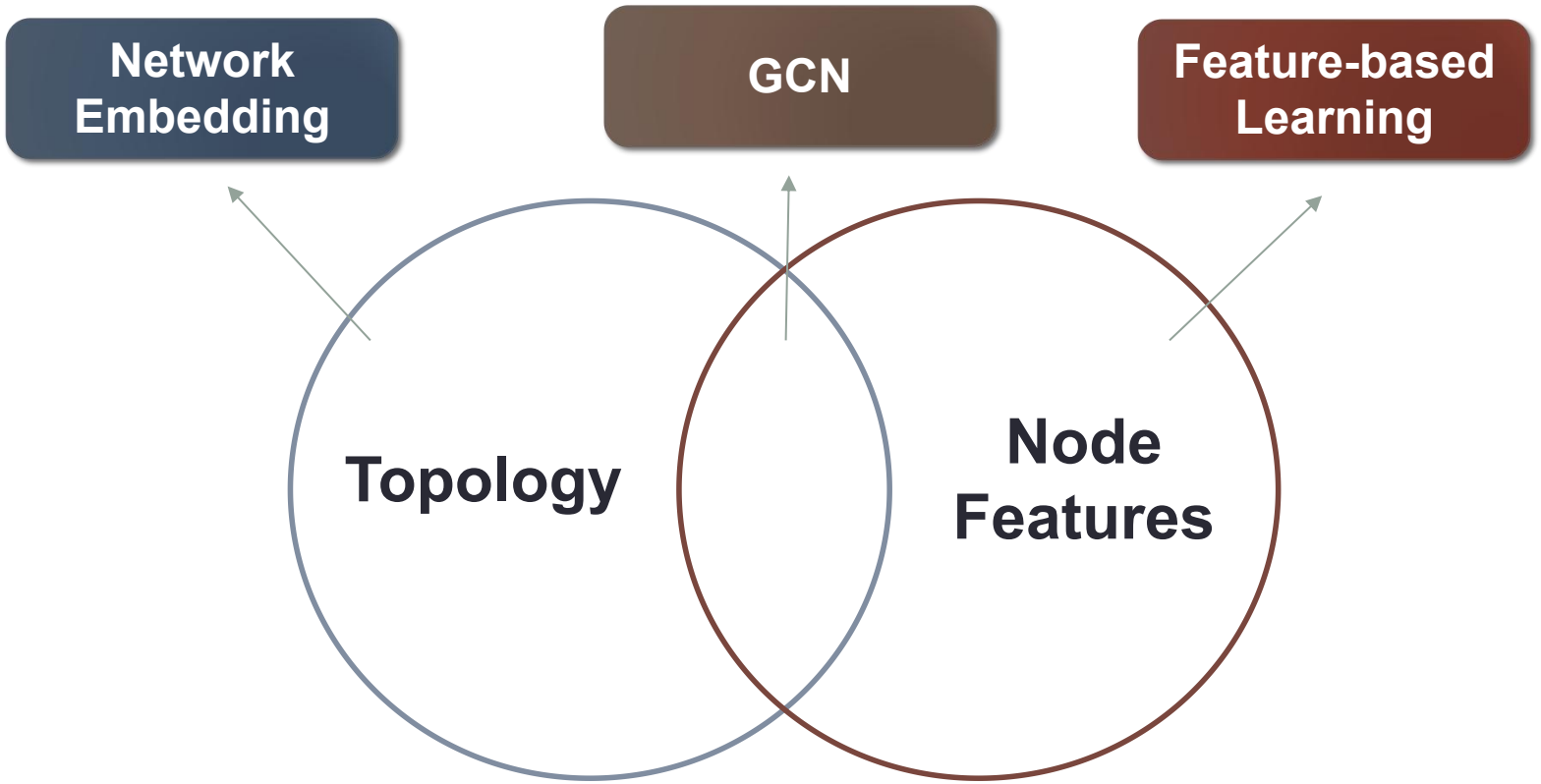
What if the problem is topology-driven?

- Since GCN is filtering features, it is inevitably feature-driven
  - Structure only provides auxiliary information (e.g. for filtering/smoothing)
- When feature plays the key role, GNN performs good …
- How about the contrary?
- Synthesis data: stochastic block model + random features

Method	Results
Random	10.0
GCN	18.3 ± 1.1
DeepWalk	99.0 ± 0.1

Network Embedding v.s. GCN

There is no better one, but there is more proper one.



Rethinking: Is GCN truly a Deep Learning method?

- Recall GNN formulation:

$$H^{(k+1)} = \sigma\left(SH^{(k)}W^{(k)}\right), S = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$$

- How about removing the non-linear component:

$$H^{(k+1)} = SH^{(k)}W^{(k)}$$

  - Stacking multiple layers and add softmax classification:

$$\begin{aligned}\hat{Y} &= softmax\left(H^{(K)}\right) \\ &= softmax\left(SS\ldots SH^{(0)}W^{(0)}W^{(1)}\ldots W^{(K-1)}\right) \\ &= softmax\left(S^K H^{(0)}W\right)\end{aligned}$$

High-order proximity

Wu, Felix, et al. Simplifying graph convolutional networks. *ICML, 2019.*

Rethinking: Is GCN truly a Deep Learning method?

- This simplified GNN (SGC) shows remarkable results:

Node classification

	Cora	Citeseer	Pubmed
GCN	81.4 ± 0.4	70.9 ± 0.5	79.0 ± 0.4
GAT	83.3 ± 0.7	72.6 ± 0.6	78.5 ± 0.3
FastGCN	79.8 ± 0.3	68.8 ± 0.6	77.4 ± 0.3
GIN	77.6 ± 1.1	66.1 ± 0.9	77.0 ± 1.2
LNet	$80.2 \pm 3.0^{\dagger}$	67.3 ± 0.5	$78.3 \pm 0.6^{\dagger}$
AdaLNet	$81.9 \pm 1.9^{\dagger}$	$70.6 \pm 0.8^{\dagger}$	$77.8 \pm 0.7^{\dagger}$
DGI	82.5 ± 0.7	71.6 ± 0.7	78.4 ± 0.7
SGC	81.0 ± 0.0	71.9 ± 0.1	78.9 ± 0.0

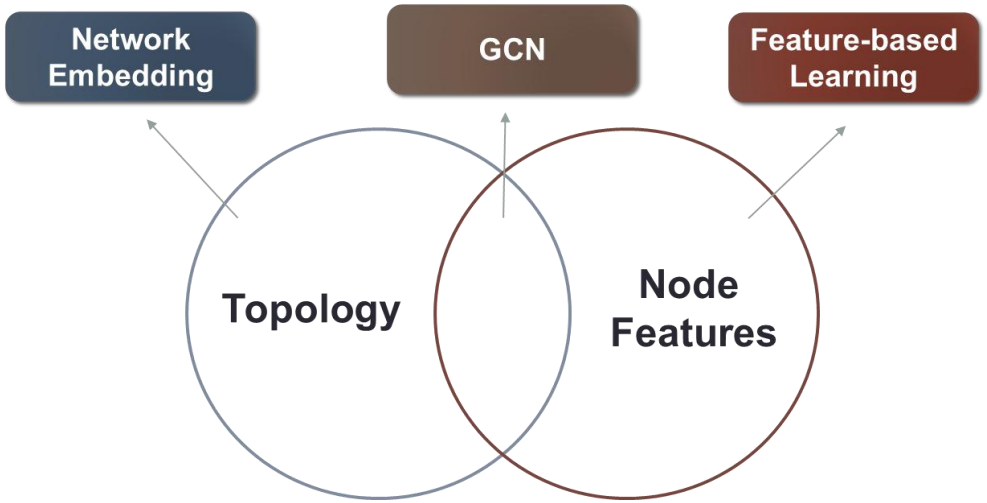
Text Classification

Dataset	Model	Test Acc. ↑	Time (seconds) ↓
20NG	GCN	87.9 ± 0.2	1205.1 ± 144.5
	SGC	88.5 ± 0.1	19.06 ± 0.15
R8	GCN	97.0 ± 0.2	129.6 ± 9.9
	SGC	97.2 ± 0.1	1.90 ± 0.03
R52	GCN	93.8 ± 0.2	245.0 ± 13.0
	SGC	94.0 ± 0.2	3.01 ± 0.01
Ohsumed	GCN	68.2 ± 0.4	252.4 ± 14.7
	SGC	68.5 ± 0.3	3.02 ± 0.02
MR	GCN	76.3 ± 0.3	16.1 ± 0.4
	SGC	75.9 ± 0.3	4.00 ± 0.04

Wu, Felix, et al. Simplifying graph convolutional networks. *ICML, 2019.*

Summaries and Conclusions

- Unsupervised v.s. (Semi-)Supervised
- Learning for Networks v.s. Learning via Graphs
- Topology-driven v.s. Feature-driven
- Both GCN and NE need to treat the counterpart as the baselines



media and network lab

Thanks!

Peng Cui
cuip@tsinghua.edu.cn
http://pengcui.thumedialab.com