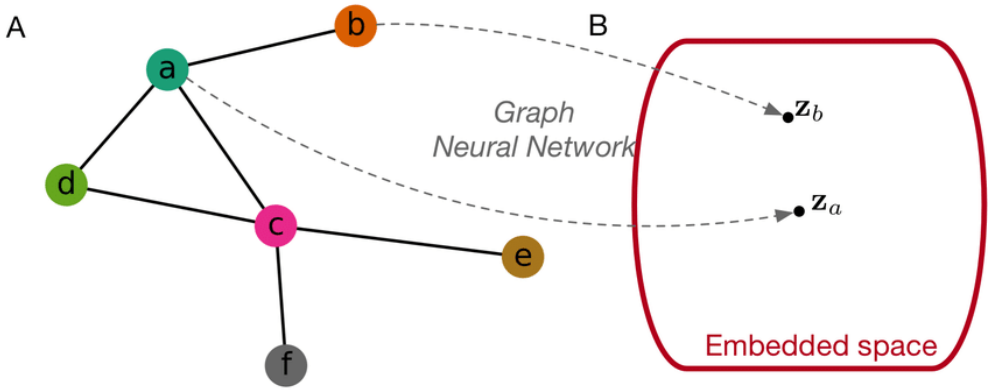
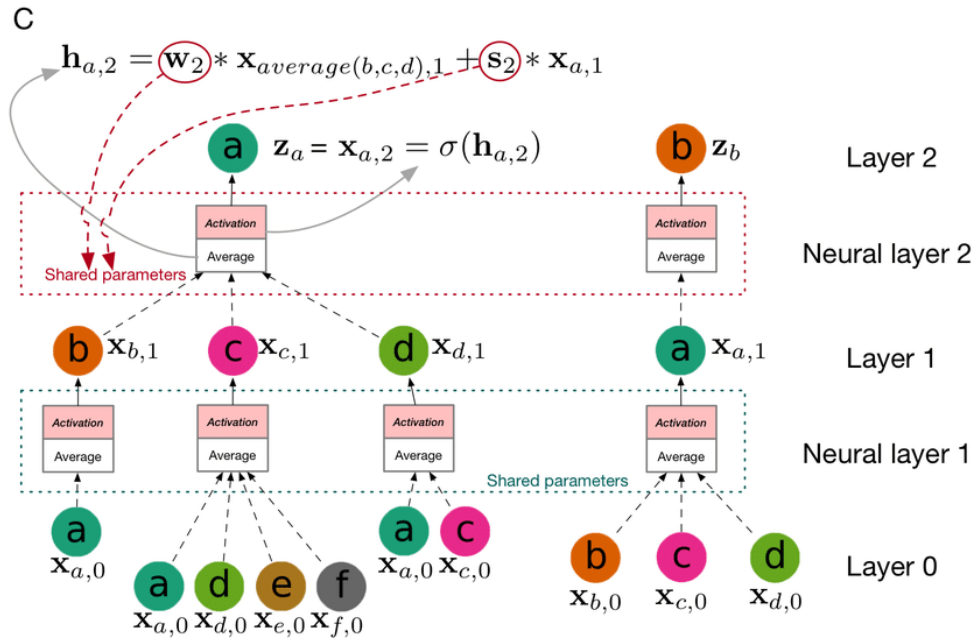




Graph Neural Networks

张江

北京师范大学系统科学学院教授

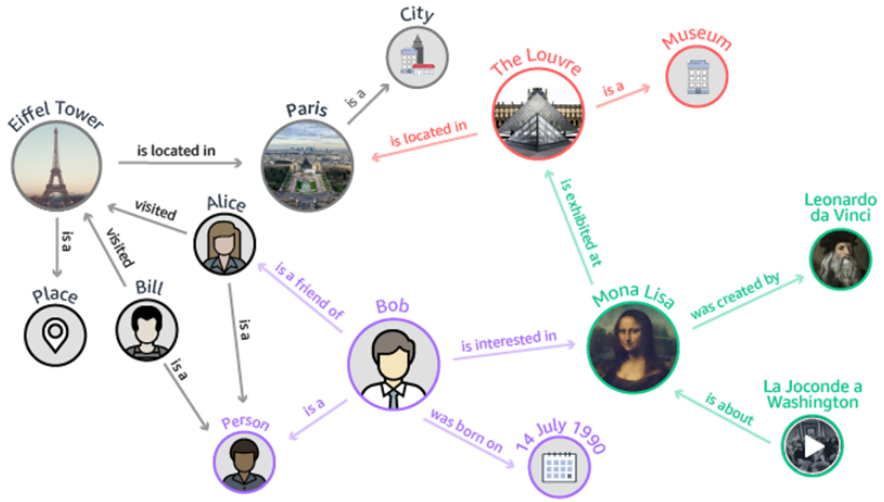
集智俱乐部、集智学园创始人

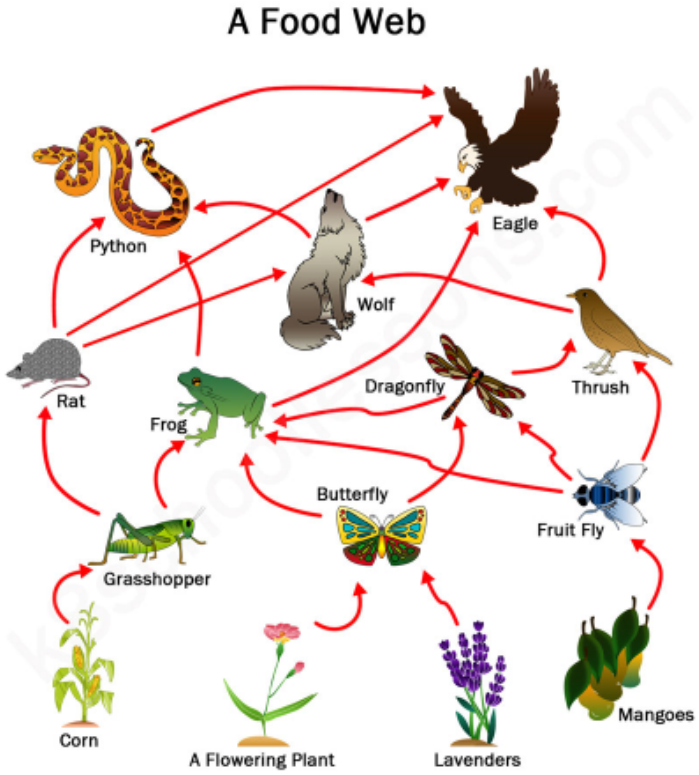
集智研究中心理事长

Outline

- Background
- Network Embedding
- Graph Neural Networks
  - Graph Convolution
  - Graph Attention Network
  - Other extension
- Variational Graph Autoencoder
- Graph Generation

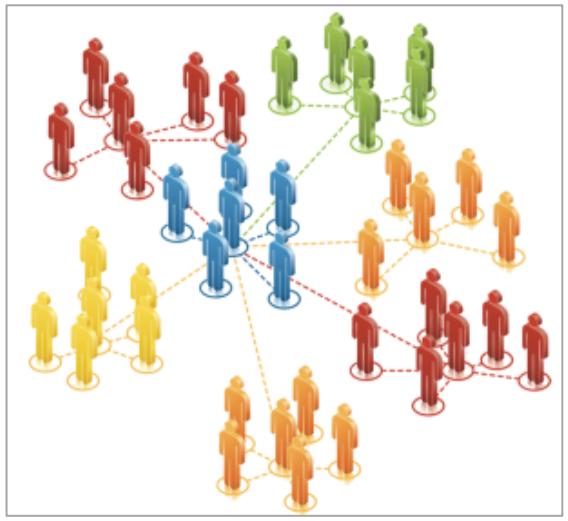
Networks Are Everywhere



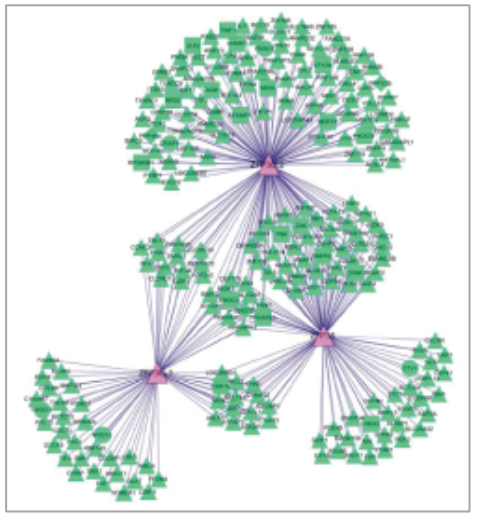


Images v.s. Graphs

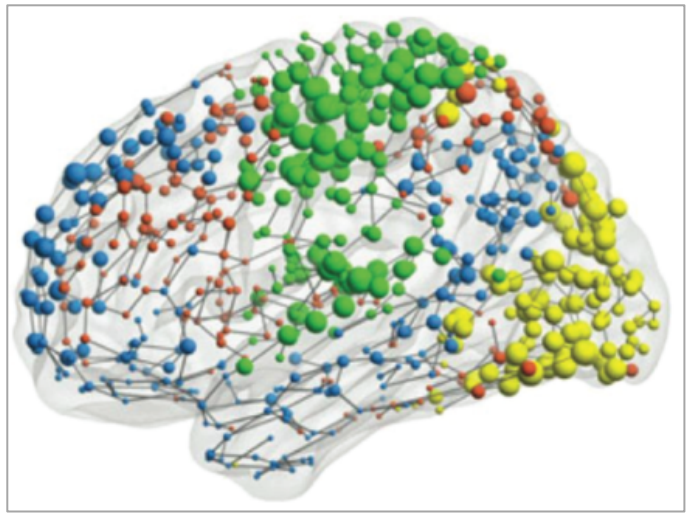
Image = 2D Grid



Social Network



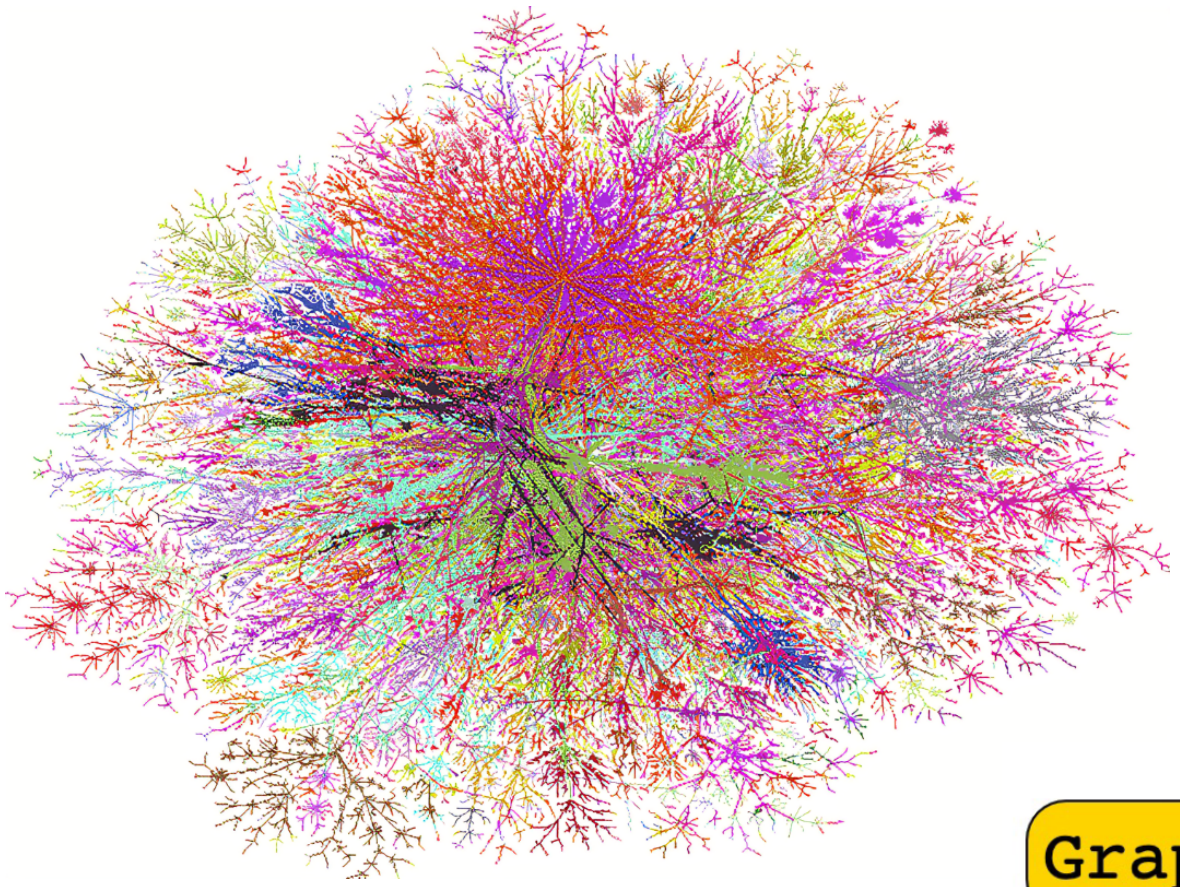
Regulatory Network



Functional Network

Michael Bronstein et al., Geometric Deep Learning, CVPR Tutorial 21 July 2017, Honolulu

Sequences v.s. Graphs



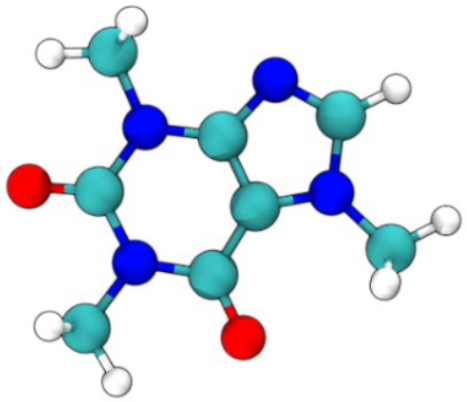
ATGACCAACAAGTGTCTCCTCCAAATTGCTCTCCTGTTGTGCTTCTCCACTACAG
CTCTTTCCATGAGCTACAACTTGCTTGGATTCCTACAAAGAAGCAGCAATTTTCA
GTGTCAGAAGCTCCTGTGGCAATTGAATGGGAGGCTTGAATACTGCCTCAAGCAC
AGGATGAACTTTGACATCCCTGAGGAGATTAAGCAGCTGCAGCAGTTCCAGAAGG
AGGACGCCGCATTGACCATCTATGAGATGCTCCAGAACATCTTTGCTATTTTCAG
ACAAGATTCATCTAGCACTGGCTGGAATGAGACTATTGTTGAGAACCTCCTGGCT
AATGTCTATCATCAGATAAACCATCTGAAGACAGTCCTGGAAGAAAAACTGGAGA
AAGAAGATTTCACCAGGGGAAAACTCATGAGCAGTCTGCACCTGAAAAGATATTA
TGGGAGGATTCTGCATTACCTGAAGGCCAAGGAGTACAGTCACTGTGCCTGGACC
ATAGTCAGAGTGGAAATCCTAAGGAACTTTTACTTCATTAACAGACTTACAGGTT
ACCTCCGAAAC, and

ATGAGCTACAACTTGCTTGGATTCCTACAAAGAAGCAGCAATTTTCAGTGTCAGA
AGCTCCTGTGGCAATTGAATGGGAGGCTTGAATACTGCCTCAAGCACAGGATGAA
CTTTGACATCCCTGAGGAGATTAAGCAGCTGCAGCAGTTCCAGAAGGAGGACGCC
GCATTGACCATCTATGAGATGCTCCAGAACATCTTTGCTATTTTCAGACAAGATT
CATCTAGCACTGGCTGGAATGAGACTATTGTTGAGAACCTCCTGGCTAATGTCTA
TCATCAGATAAACCATCTGAAGACAGTCCTGGAAGAAAAACTGGAGAAAGAAGAT
TTCACCAGGGGAAAACTCATGAGCAGTCTGCACCTGAAAAGATATTATGGGAGGA
TTCTGCATTACCTGAAGGCCAAGGAGTACAGTCACTGTGCCTGGACCATAGTCAG
AGTGGAAATCCTAAGGAACTTTTACTTCATTAACAGACTTACAGGTTACCTCCGA
AAC.

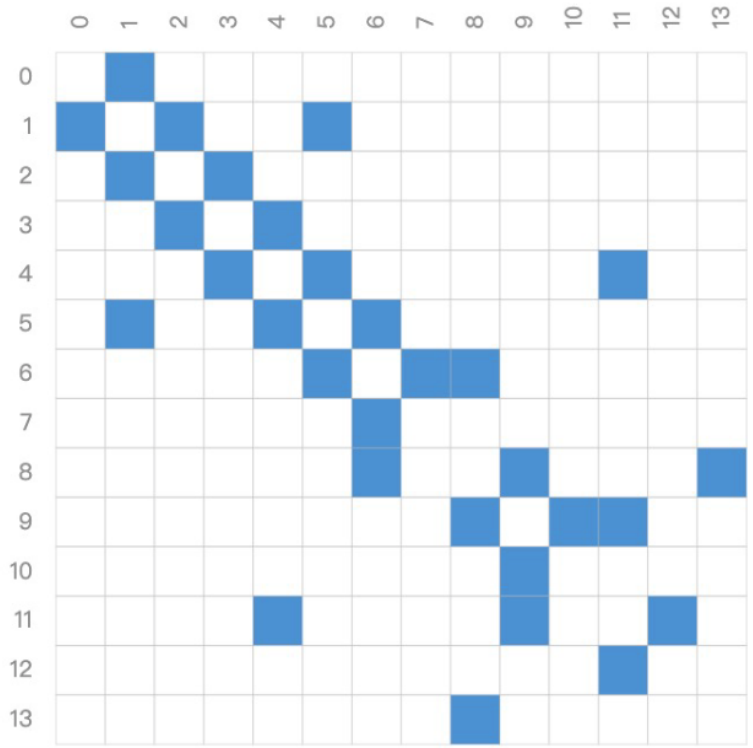
Outline

- Background
- Network Embedding
- Graph Neural Networks
  - Graph Convolution
  - Graph Attention Network
  - Other extension
- Variational Graph Autoencoder
- Graph Generation

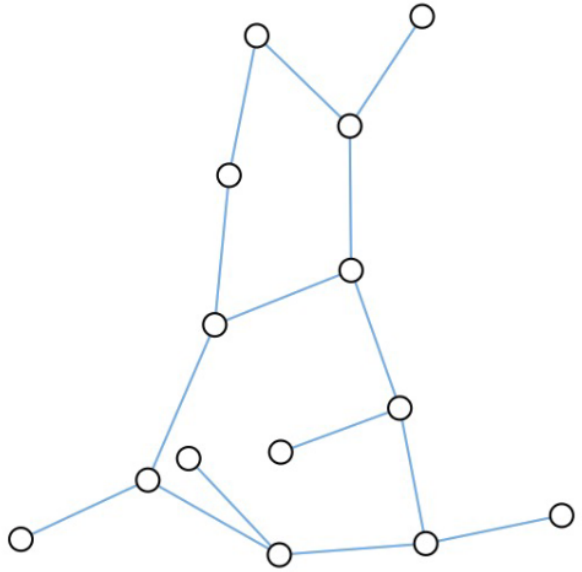
How to represent a graph?



分子结构式



邻接矩阵



图

How to represent a graph?

DeepWalk: Online Learning of Social Representations

Bryan Perozzi
Stony Brook University
Department of Computer Science

Rami Al-Rfou
Stony Brook University
Department of Computer Science

Steven Skiena
Stony Brook University
Department of Computer Science

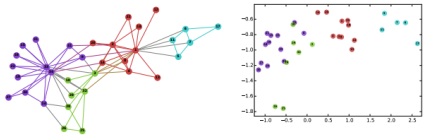
{bperozzi, ralrfou, skiena}@cs.stonybrook.edu

ABSTRACT

We present DeepWalk, a novel approach for learning latent representations of vertices in a network. These latent representations encode social relations in a continuous vector space, which is easily exploited by statistical models. DeepWalk generalizes recent advancements in language modeling and unsupervised feature learning (or *deep learning*) from sequences of words to graphs.

DeepWalk uses local information obtained from truncated random walks to *learn* latent representations by treating walks as the equivalent of sentences. We demonstrate DeepWalk's latent representations on several multi-label network classification tasks for social networks such as BlogCatalog, Flickr, and YouTube. Our results show that DeepWalk outperforms challenging baselines which are allowed a global view of the network, especially in the presence of missing information. DeepWalk's representations can provide F_1 scores up to 10% higher than competing methods when labeled data is sparse. In some experiments, DeepWalk's representations are able to outperform all baseline methods while using 60% less training data.

DeepWalk is also scalable. It is an online learning algorithm which builds useful incremental results, and is trivially parallelizable. These qualities make it suitable for a broad class of real world applications such as network classification, and anomaly detection.

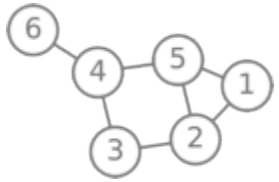


(a) Input: Karate Graph (b) Output: Representation

Figure 1: Our proposed method *learns* a latent space representation of social interactions in $\mathbb{R}^d$. The learned representation encodes community structure so it can be easily exploited by standard classification methods. Here, our method is used on Zachary's Karate network [44] to generate a latent representation in $\mathbb{R}^2$. Note the correspondence between community structure in the input graph and the embedding. Vertex colors represent a modularity-based clustering of the input graph.

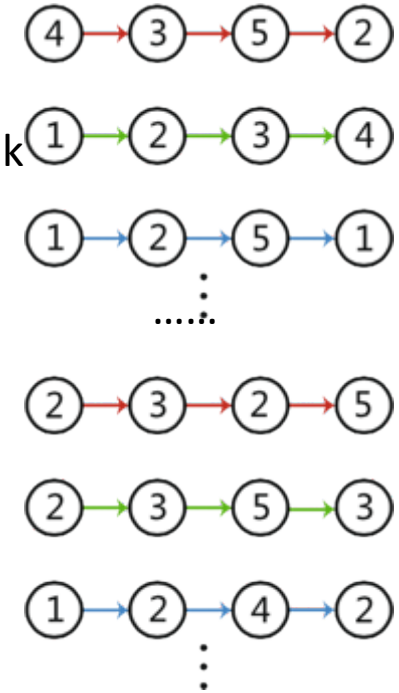
ommendation [11], anomaly detection [5], and missing link prediction [22]) must be able to deal with this sparsity in order to survive.

In this paper we introduce *deep learning* (unsupervised feature learning) [2] techniques, which have proven successful in natural language processing, into network analysis for



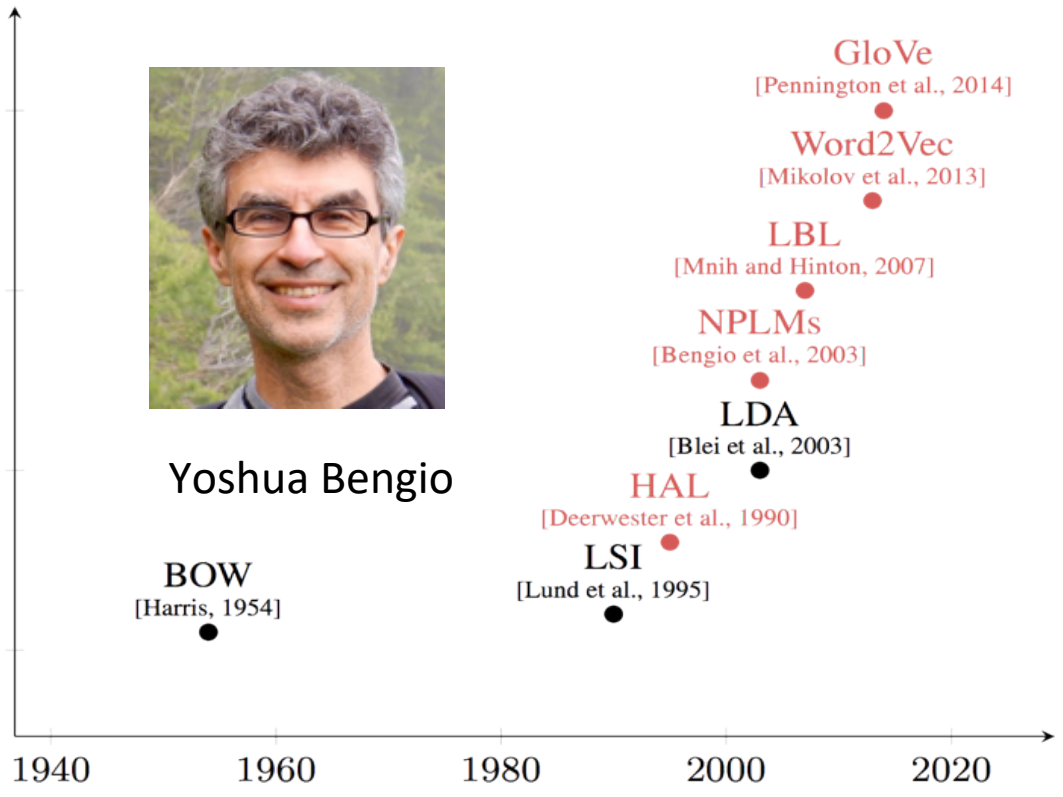
Random Walk

Node Sequences

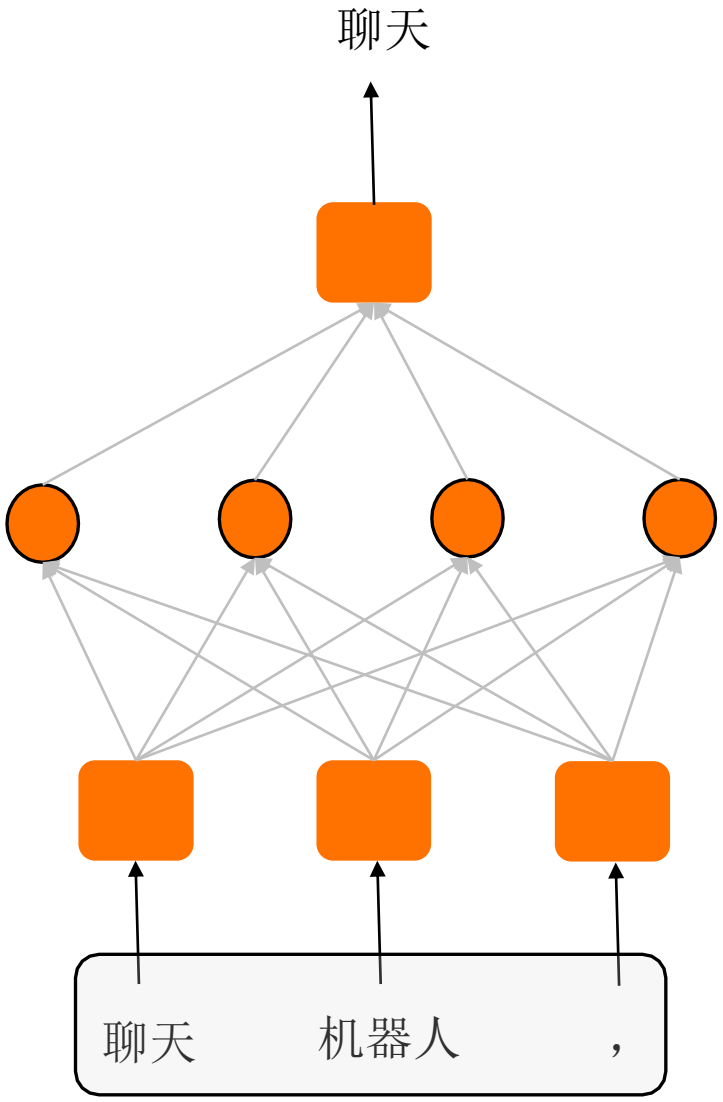


https://arxiv.org/pdf/1403.6652.pdf

Neural Language Model



Yoshua Bengio

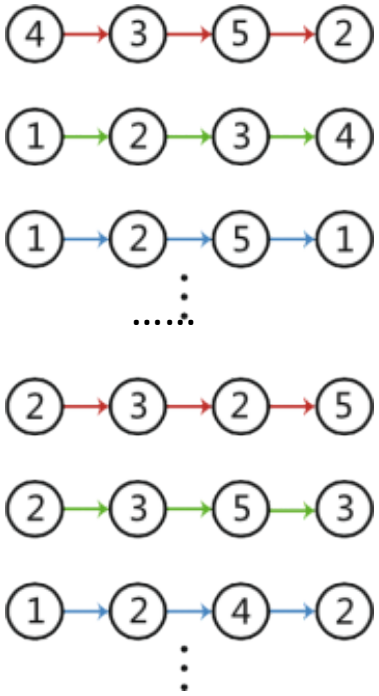


F.Sun et al.: Learning Word Representations by Jointly Modeling Syntagmatic and Paradigmatic Relations (slides), http://www.bigdatalab.ac.cn/~lanyanyan/slides/2015/ACL2015-sun.pdf

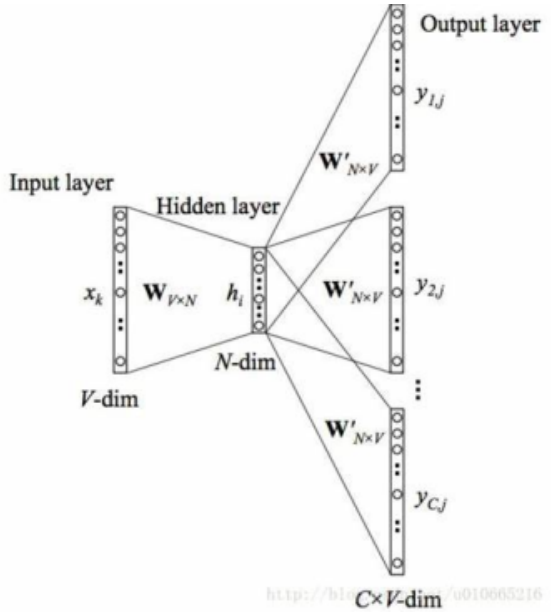
Deep Walk

Random Walk

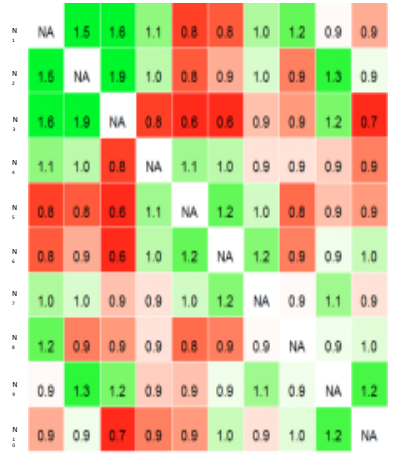
Node Sequences



Word2Vec



Node Vectors



https://arxiv.org/pdf/1403.6652.pdf

Deep Walk

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 $\Phi \in \mathbb{R}^{|V| \times d}$

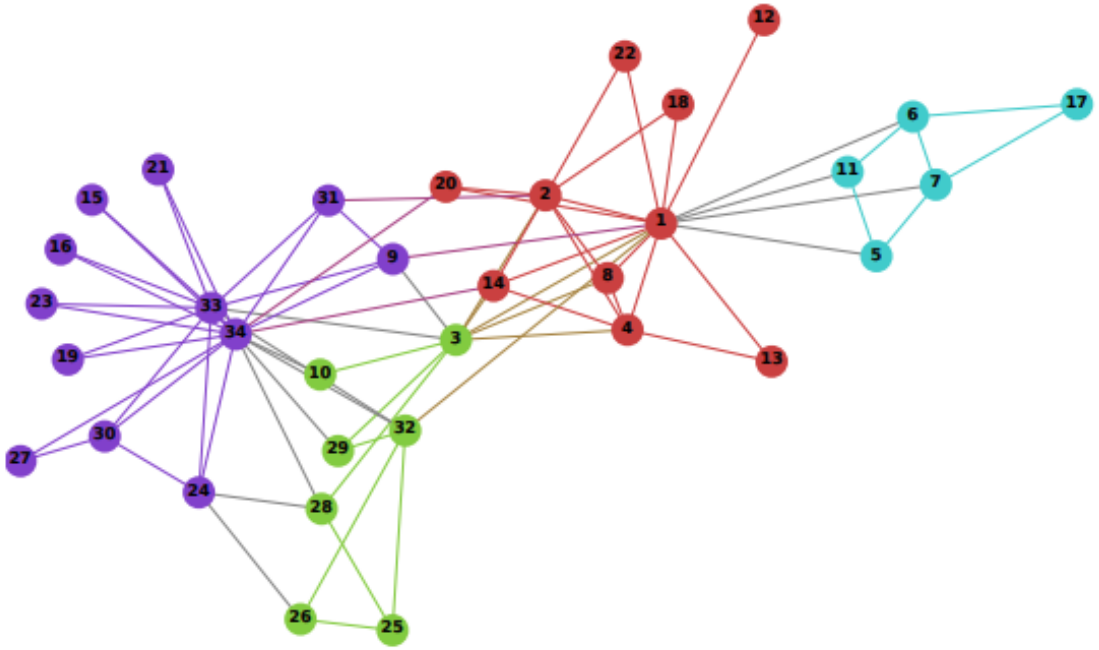
Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
  $\mathcal{O} = \text{Shuffle}(V)$
  for each $v_i \in \mathcal{O}$ **do**
    $\mathcal{W}_{v_i} = RandomWalk(G, v_i, \text{t})$
    SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
  end for
end for

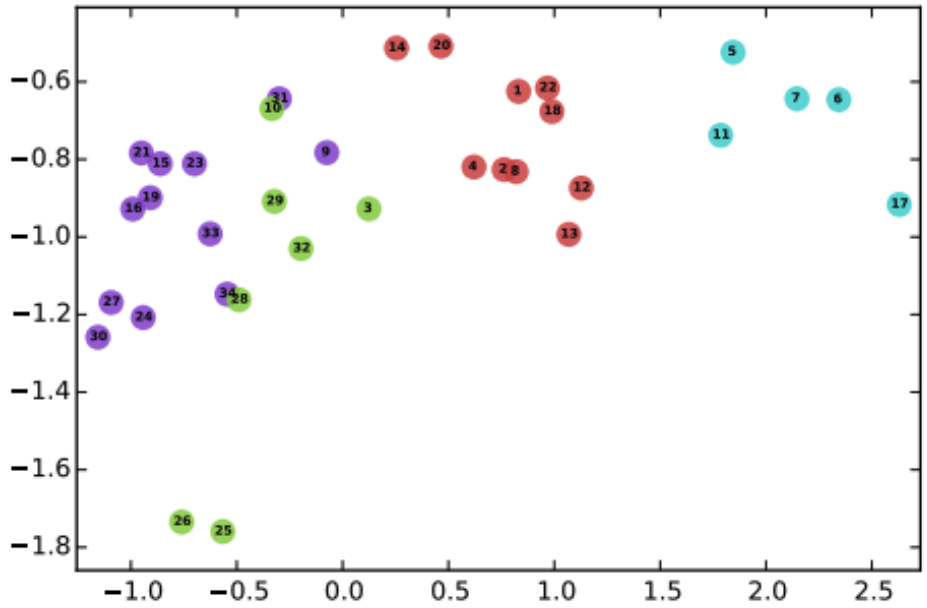
Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

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

https://arxiv.org/pdf/1403.6652.pdf

Deep Walk



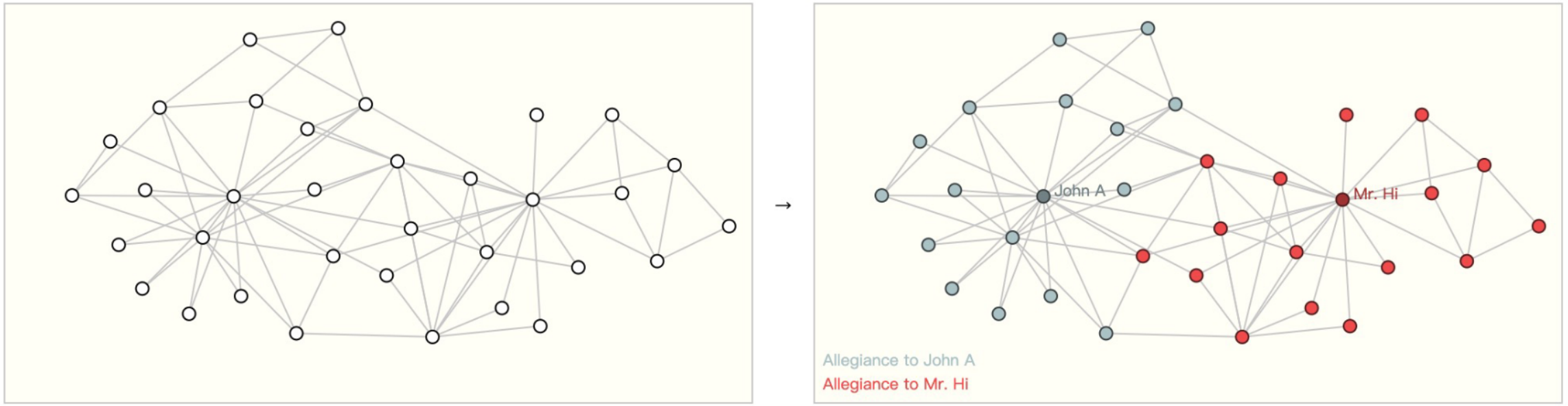


https://arxiv.org/pdf/1403.6652.pdf

Tasks

- **Node level**：Node classification, Node properties prediction, Community detection, Visualization
- **Link level**：Link classification, Link Prediction, Link property inference
- **Graph level**：Graph classification, Graph generation
- ……

Node Classification/Labelling

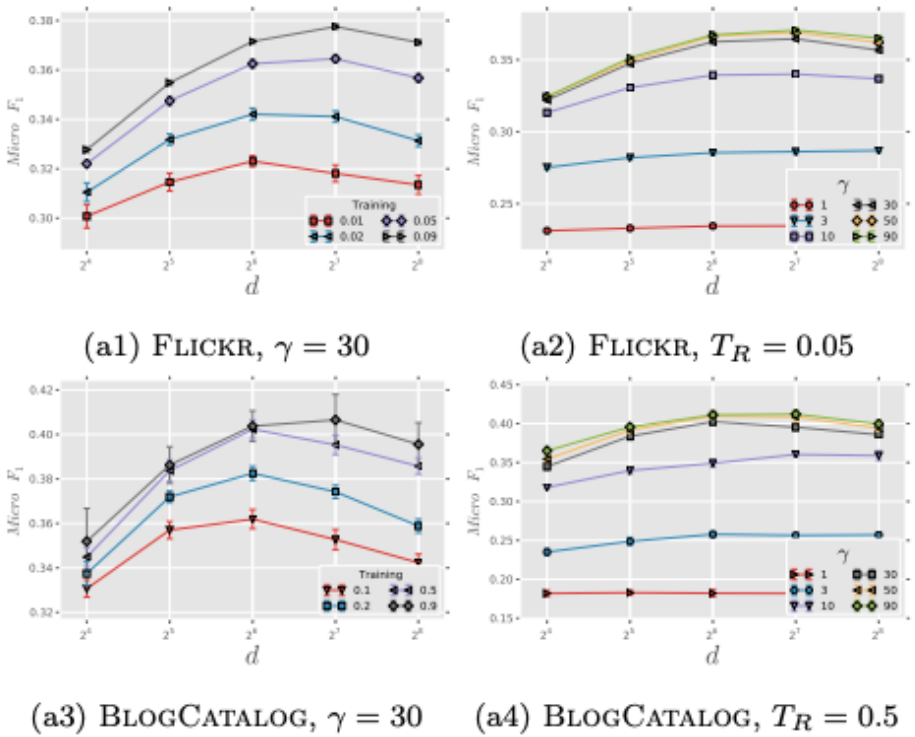


Input: graph with unlabled nodes

Output: graph node labels

有监督、无监督、半监督

Deep Walk



(a1) FLICKR, $\gamma = 30$ (a2) FLICKR, $T_R = 0.05$

(a3) BLOGCATALOG, $\gamma = 30$ (a4) BLOGCATALOG, $T_R = 0.5$

(a) Stability over dimensions, d

	% 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

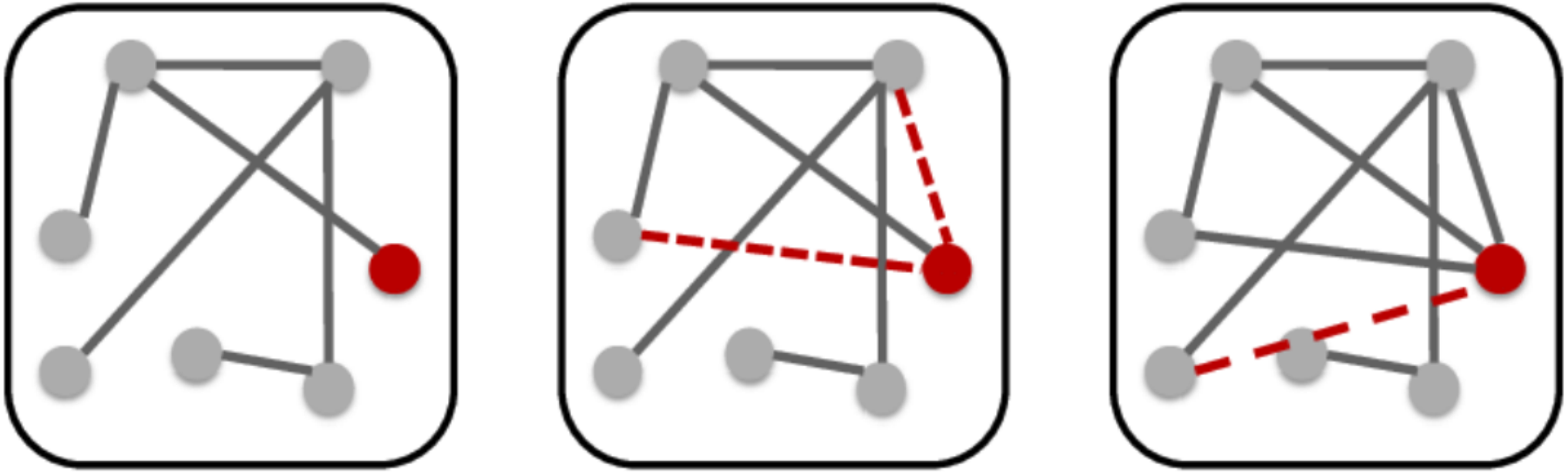
Table 2: Multi-label classification results in BLOGCATALOG

	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	DEEPWALK	**32.4**	**34.6**	**35.9**	**36.7**	**37.2**	**37.7**	**38.1**	**38.3**	**38.5**	**38.7**
	SpectralClustering	27.43	30.11	31.63	32.69	33.31	33.95	34.46	34.81	35.14	35.41
	EdgeCluster	25.75	28.53	29.14	30.31	30.85	31.53	31.75	31.76	32.19	32.84
	Modularity	22.75	25.29	27.3	27.6	28.05	29.33	29.43	28.89	29.17	29.2
	wvRN	17.7	14.43	15.72	20.97	19.83	19.42	19.22	21.25	22.51	22.73
	Majority	16.34	16.31	16.34	16.46	16.65	16.44	16.38	16.62	16.67	16.71
Macro-F1(%)	DEEPWALK	**14.0**	17.3	**19.6**	**21.1**	**22.1**	**22.9**	**23.6**	**24.1**	**24.6**	**25.0**
	SpectralClustering	13.84	**17.49**	19.44	20.75	21.60	22.36	23.01	23.36	23.82	24.05
	EdgeCluster	10.52	14.10	15.91	16.72	18.01	18.54	19.54	20.18	20.78	20.85
	Modularity	10.21	13.37	15.24	15.11	16.14	16.64	17.02	17.1	17.14	17.12
	wvRN	1.53	2.46	2.91	3.47	4.95	5.56	5.82	6.59	8.00	7.26
	Majority	0.45	0.44	0.45	0.46	0.47	0.44	0.45	0.47	0.47	0.47

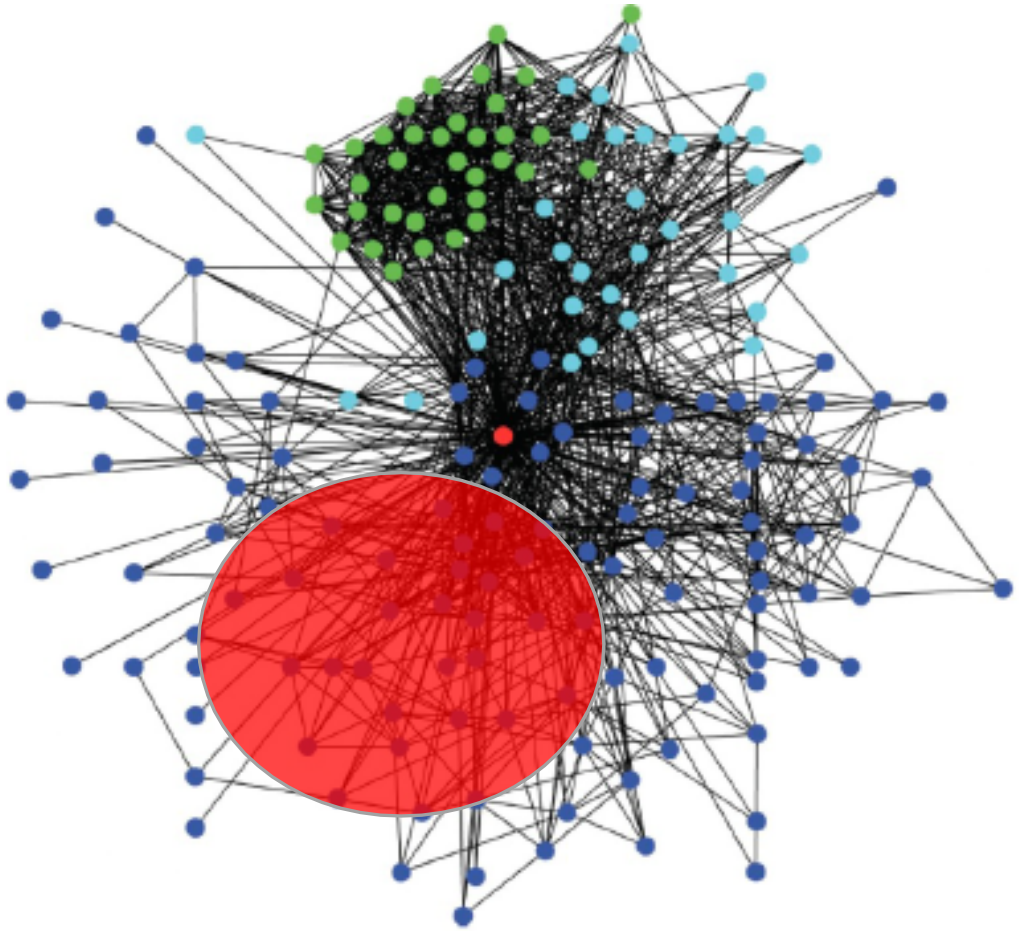
Table 3: Multi-label classification results in FLICKR

https://arxiv.org/pdf/1403.6652.pdf

Link Prediction



Link Prediction



Link Prediction

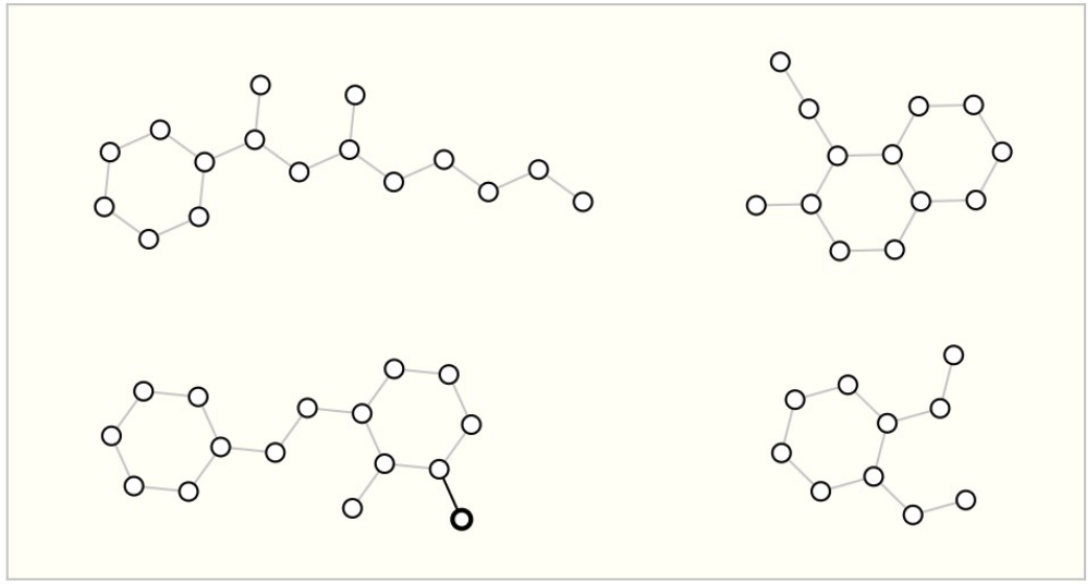
u,v: nodes, f: embedding function

Operator	Symbol	Definition
Average	$\boxplus$	$[f(u) \boxplus f(v)]_i = \frac{f_i(u)+f_i(v)}{2}$
Hadamard	$\boxdot$	$[f(u) \boxdot f(v)]_i = f_i(u) * f_i(v)$
Weighted-L1	$\|\cdot\|_{\bar{1}}$	$\|f(u) \cdot f(v)\|_{\bar{1}i} = \|f_i(u) - f_i(v)\|$
Weighted-L2	$\|\cdot\|_{\bar{2}}$	$\|f(u) \cdot f(v)\|_{\bar{2}i} = \|f_i(u) - f_i(v)\|^2$

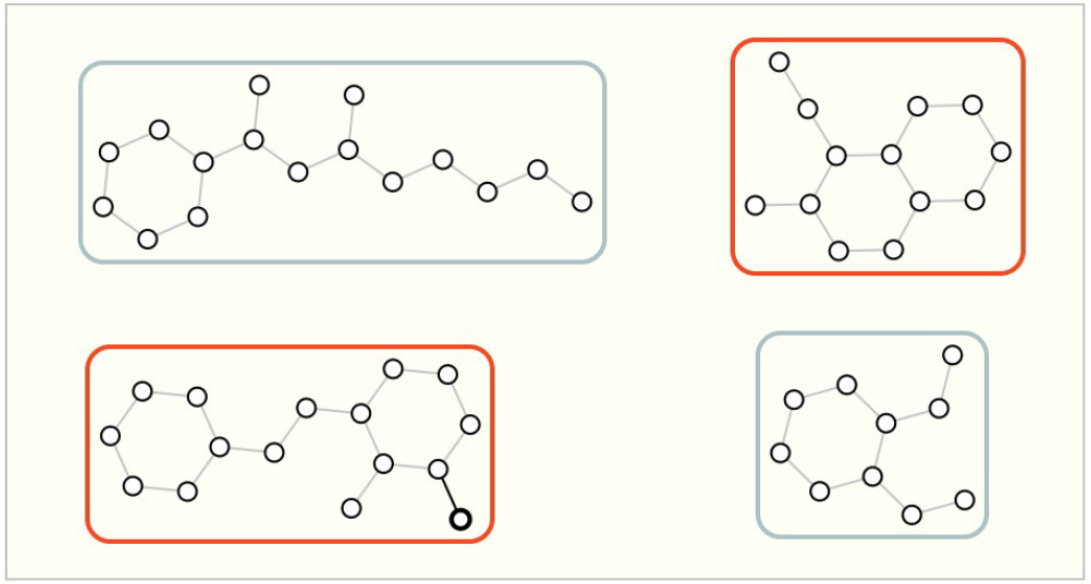
Area Under Curve (AUC) scores for link prediction. Comparison with popular baselines and embedding based methods bootstapped using binary operators: (a) Average, (b) Hadamard, (c) Weighted-L1, and (d) Weighted-L2 (See Table 1 for definitions).

Op	Algorithm	Dataset		
		Facebook	PPI	arXiv
	Common Neighbors	0.8100	0.7142	0.8153
	Jaccard's Coefficient	0.8880	0.7018	0.8067
	Adamic-Adar	0.8289	0.7126	0.8315
	Pref. Attachment	0.7137	0.6670	0.6996
	Spectral Clustering	0.5960	0.6588	0.5812
(a)	DeepWalk	0.7238	0.6923	0.7066
	LINE	0.7029	0.6330	0.6516
	node2vec	0.7266	0.7543	0.7221
	Spectral Clustering	0.6192	0.4920	0.5740
(b)	DeepWalk	**0.9680**	0.7441	0.9340
	LINE	0.9490	0.7249	0.8902
	node2vec	**0.9680**	**0.7719**	**0.9366**
	Spectral Clustering	0.7200	0.6356	0.7099
(c)	DeepWalk	0.9574	0.6026	0.8282
	LINE	0.9483	0.7024	0.8809
	node2vec	0.9602	0.6292	0.8468
	Spectral Clustering	0.7107	0.6026	0.6765
(d)	DeepWalk	0.9584	0.6118	0.8305
	LINE	0.9460	0.7106	0.8862
	node2vec	0.9606	0.6236	0.8477

Graph Level Tasks



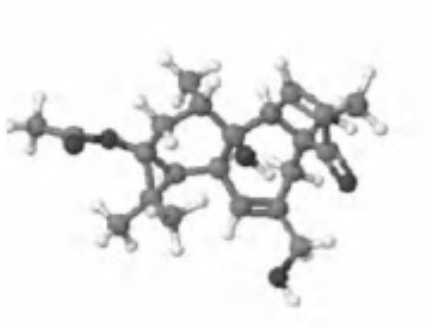
Input: graphs



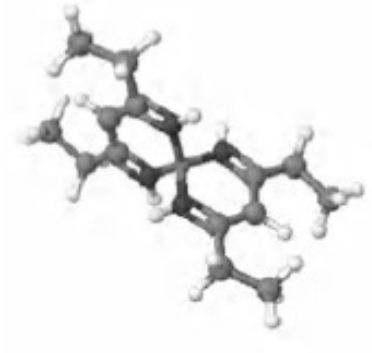
Output: labels for each graph, (e.g., "does the graph contain two rings?")

Graph Classification

分子结构网络



Toxic



non-toxic

化学药品分类

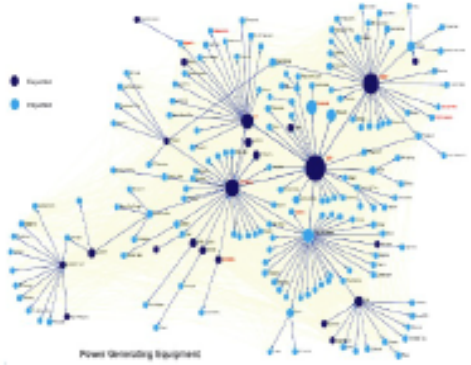
社会网络

Long lived

Dead

网络社区分类

国际贸易网



Industrial

Agricultrual

贸易产品分类

Graph Classification

TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 01/10 pp447–457
DOI: 10.26599/TST.2019.9010055
Volume 25, Number 4, August 2020

Complex Network Classification with Convolutional Neural Network

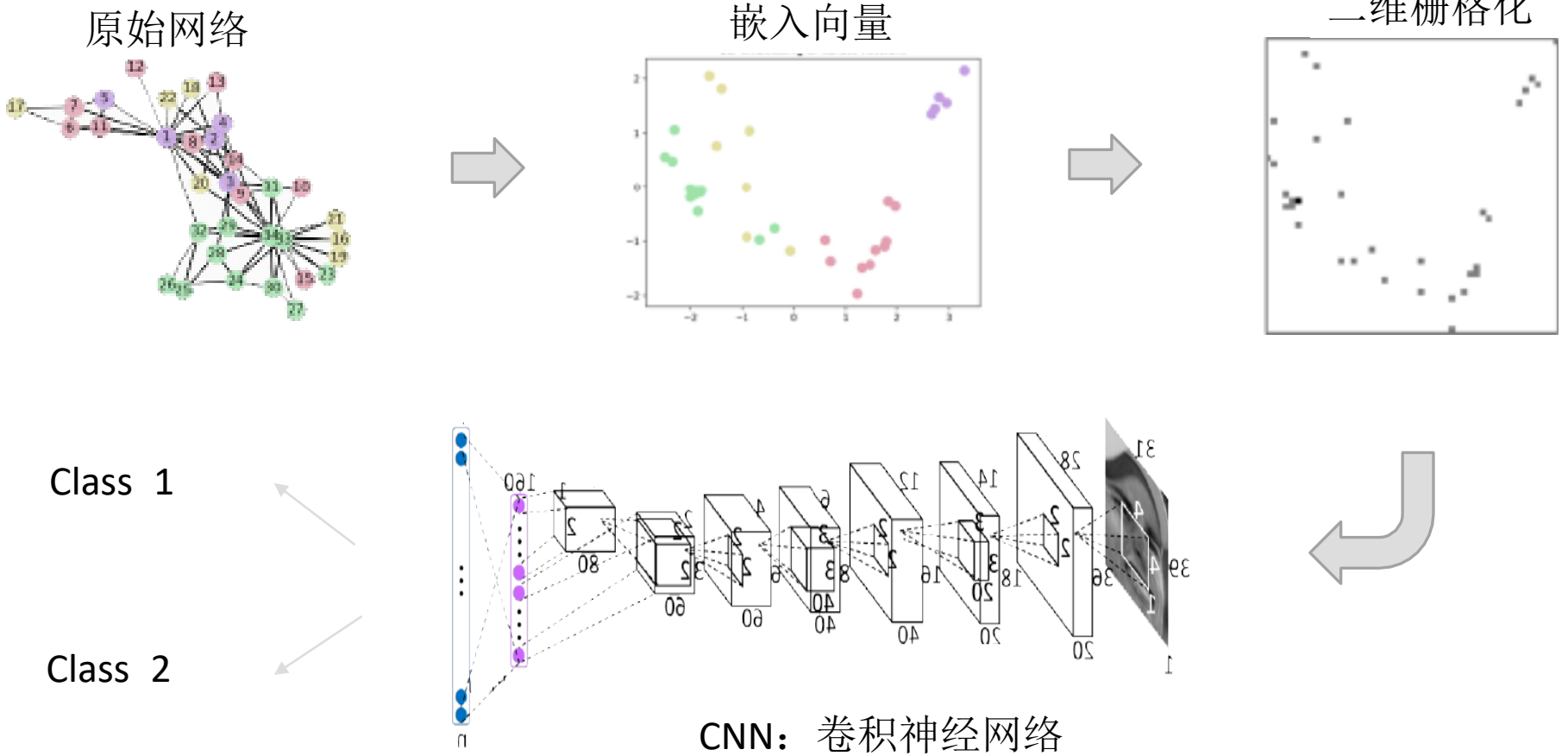
Ruyue Xin, Jiang Zhang*, and Yitong Shao

Abstract: Classifying large-scale networks into several categories and distinguishing them according to their fine structures is of great importance to several real-life applications. However, most studies on complex networks focus on the properties of a single network and seldom on classification, clustering, and comparison between different networks, in which the network is treated as a whole. Conventional methods can hardly be applied on networks directly due to the non-Euclidean properties of data. In this paper, we propose a novel framework of Complex Network Classifier (CNC) by integrating network embedding and convolutional neural network to tackle the problem of network classification. By training the classifier on synthetic complex network data, we show CNC can not only classify networks with high accuracy and robustness but can also extract the features of the networks automatically. We also compare our CNC with baseline methods on benchmark datasets, which shows that our method performs well on large-scale networks.

Key words: complex network; network classification; DeepWalk; Convolutional Neural Network (CNN)

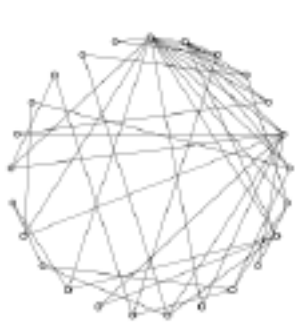
Ruyue Xin,Jiang Zhang,Yitong Shao. Complex Network Classification with Convolutional Neural Network; Tsinghua Science and Technology, Volume 25, Number 4, 2020

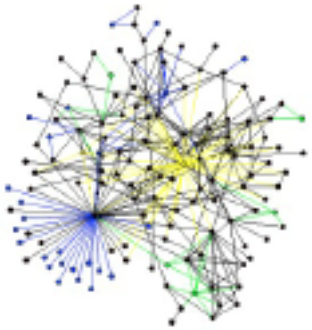
Graph Classification



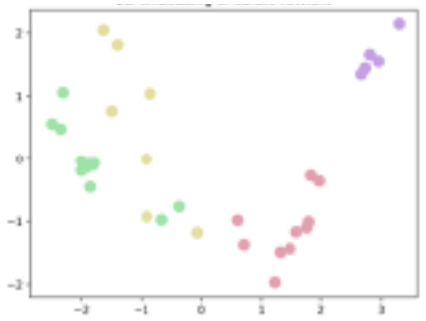
Ruyue Xin,Jiang Zhang,Yitong Shao. Complex Network Classification with Convolutional Neural Network; Tsinghua Science and Technology, Volume 25, Number 4, 2020

Graph Classification

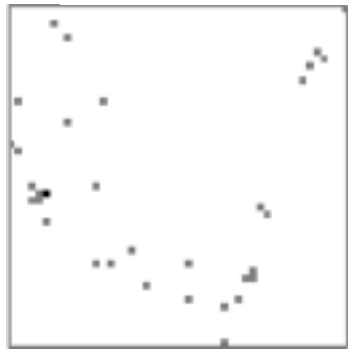




嵌入向量

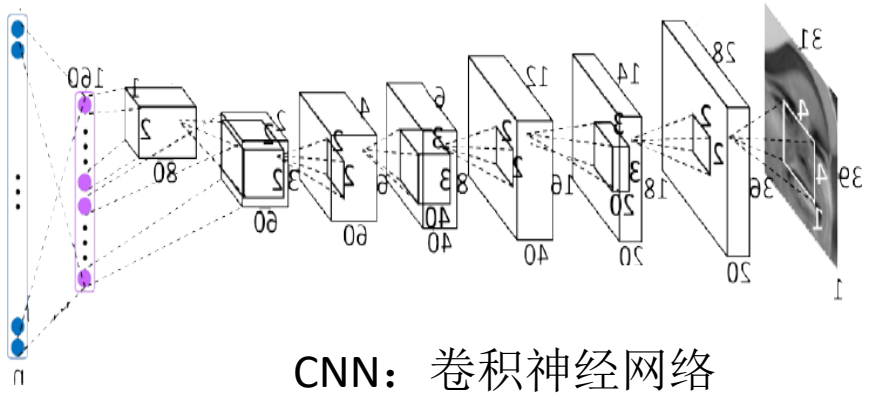


二维栅格化



小世界网络

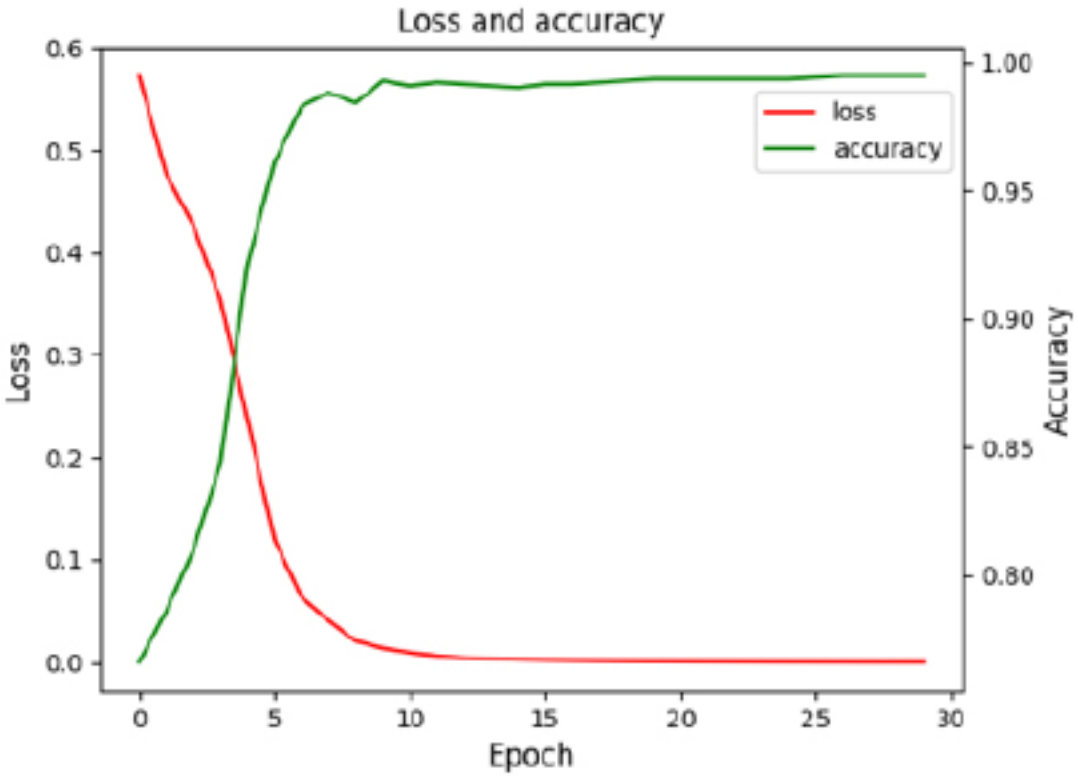
无标度网络

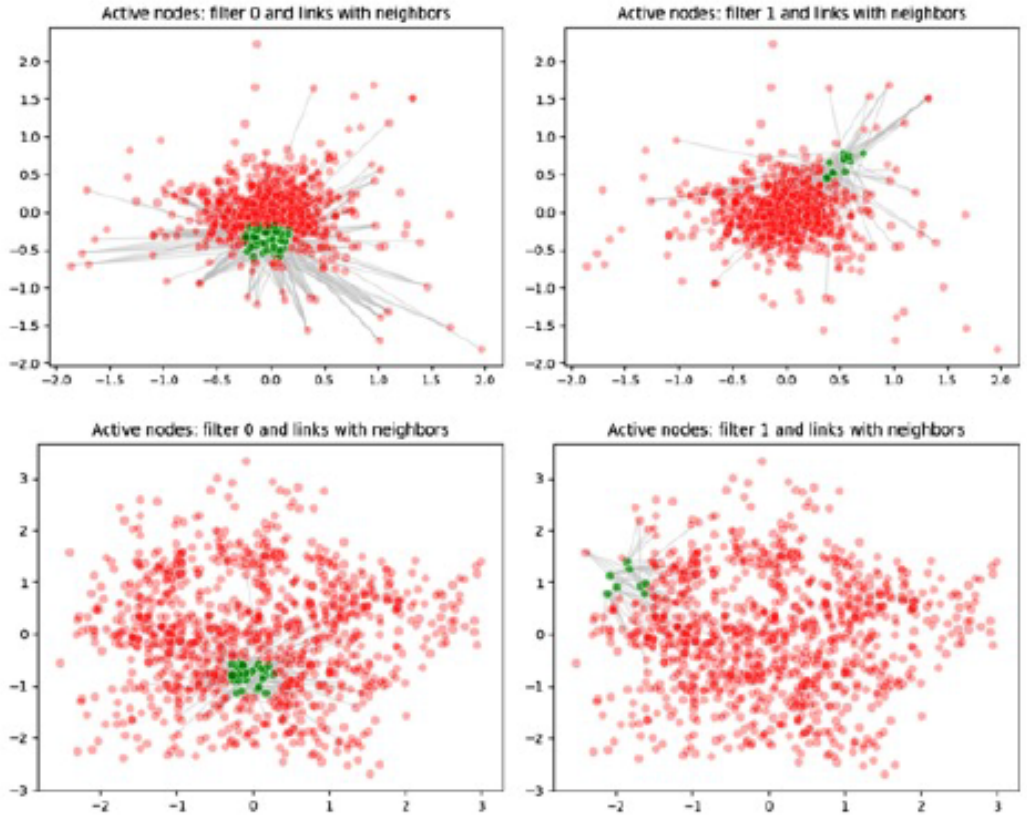


CNN：卷积神经网络

Ruyue Xin,Jiang Zhang,Yitong Shao. Complex Network Classification with Convolutional Neural Network; Tsinghua Science and Technology, Volume 25, Number 4, 2020

Results



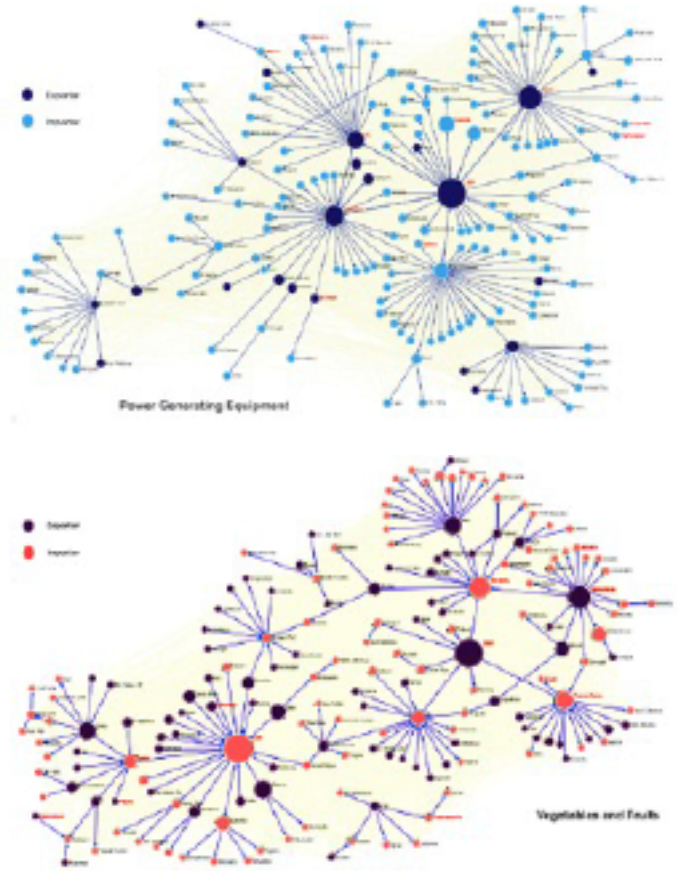


无标度

小世界

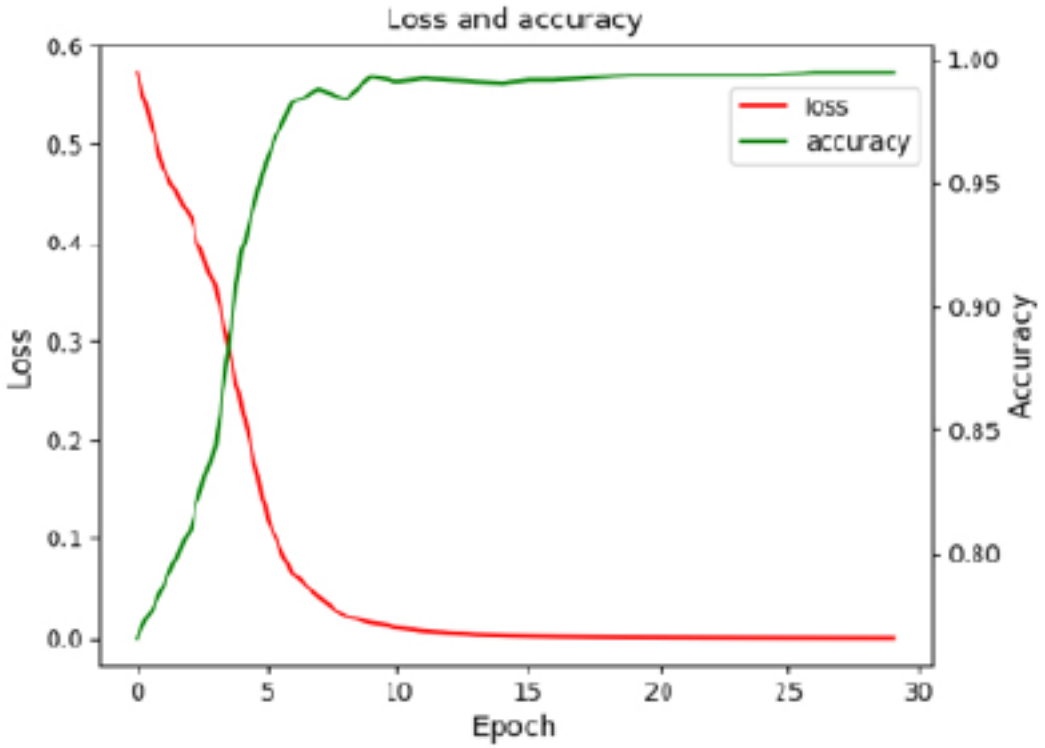
Ruyue Xin,Jiang Zhang,Yitong Shao. Complex Network Classification with Convolutional Neural Network; Tsinghua Science and Technology, Volume 25, Number 4, 2020

Results



Electronic

Agricultrual



The classifier can distinguish agricultural trade networks and electronic product trade networks with accuracy 99%

Ruyue Xin,Jiang Zhang,Yitong Shao. Complex Network Classification with Convolutional Neural Network; Tsinghua Science and Technology, Volume 25, Number 4, 2020

Results

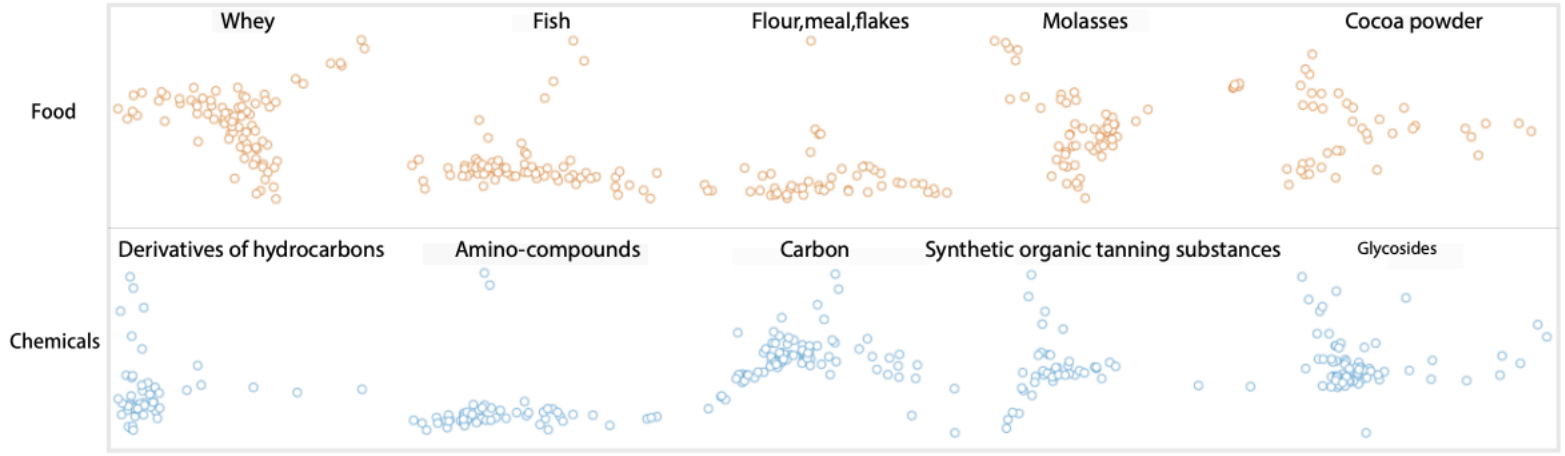


Fig. 7 Network representations of 10 selected products in food (upper) and chemicals (bottom).

Table 2 Comparison of classification accuracy (± standard deviation).

(%)

Method	Dataset				
	NCI1	COLLAB	RE_B	RE_5K	RE_12K
GK	62.28±0.29	72.84 ± 0.28	77.34±0.18	41.01±0.17	31.82±0.08
WL	**80.22±0.51**	**77.82±1.45**	78.52±2.01	50.77±2.02	34.57±1.32
Deep GK	62.48±0.25	73.09±0.25	78.04±0.39	41.27±0.18	32.22±0.10
PSCK, $k = 10$	70.00±1.98	72.60±2.15	86.30±1.58	49.10±0.70	41.32±0.42
CNC_tSNE	63.18±3.35	63.46±1.59	80.17±2.66	46.15±1.55	36.53±0.97
CNC	63.11±0.56	67.79±2.34	**86.72±1.55**	**51.35±3.02**	**41.44±1.64**

Ruyue Xin,Jiang Zhang,Yitong Shao. Complex Network Classification with Convolutional Neural Network; Tsinghua Science and Technology, Volume 25, Number 4, 2020

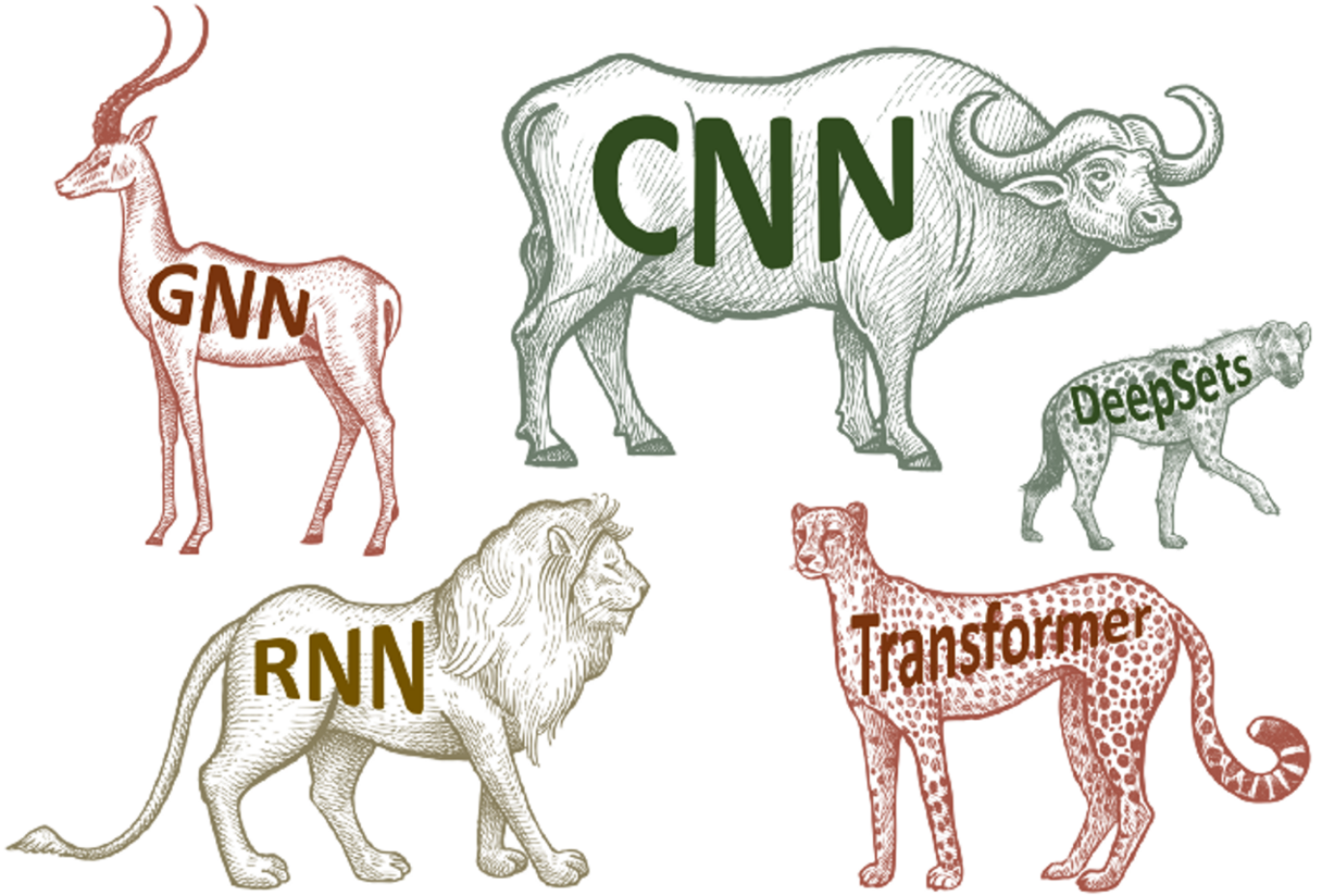
Weak points

- 需要随机游走，比较费时间
- 一个图只能得到一组嵌入，不能应付图的变化
- 很难融入节点、连边属性信息
- 依赖于词向量技术

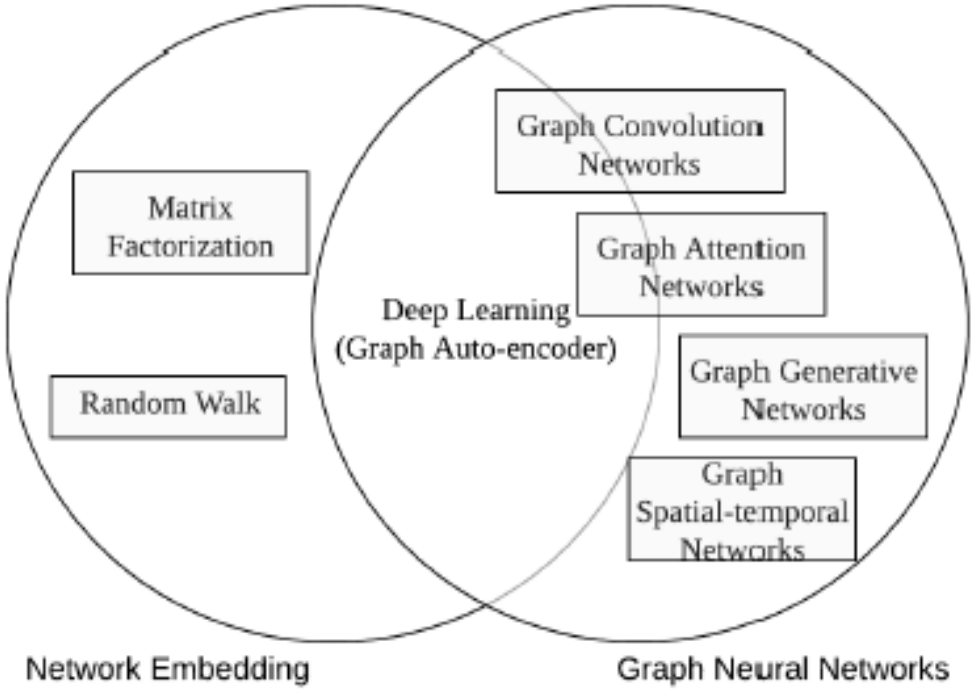
Outline

- Background
- Network Embedding
- Graph Neural Networks
  - Graph Convolution
  - Graph Attention Network
  - Other extension
- Variational Graph Autoencoder
- Graph Generation

Neural Networks

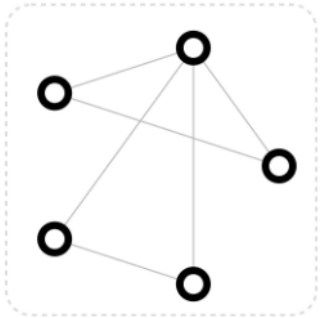


Network Embedding v.s. Graph Neural Networks



: Network Embedding v.s. Graph Neural Networks.

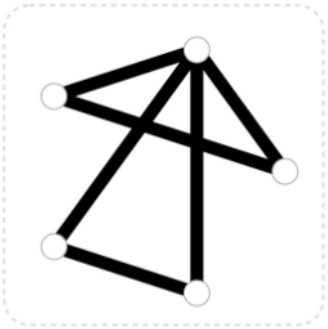
What is Graph Neural Network?



V Vertex (or node) attributes
e.g., node identity, number of neighbors

E Edge (or link) attributes and directions
e.g., edge identity, edge weight

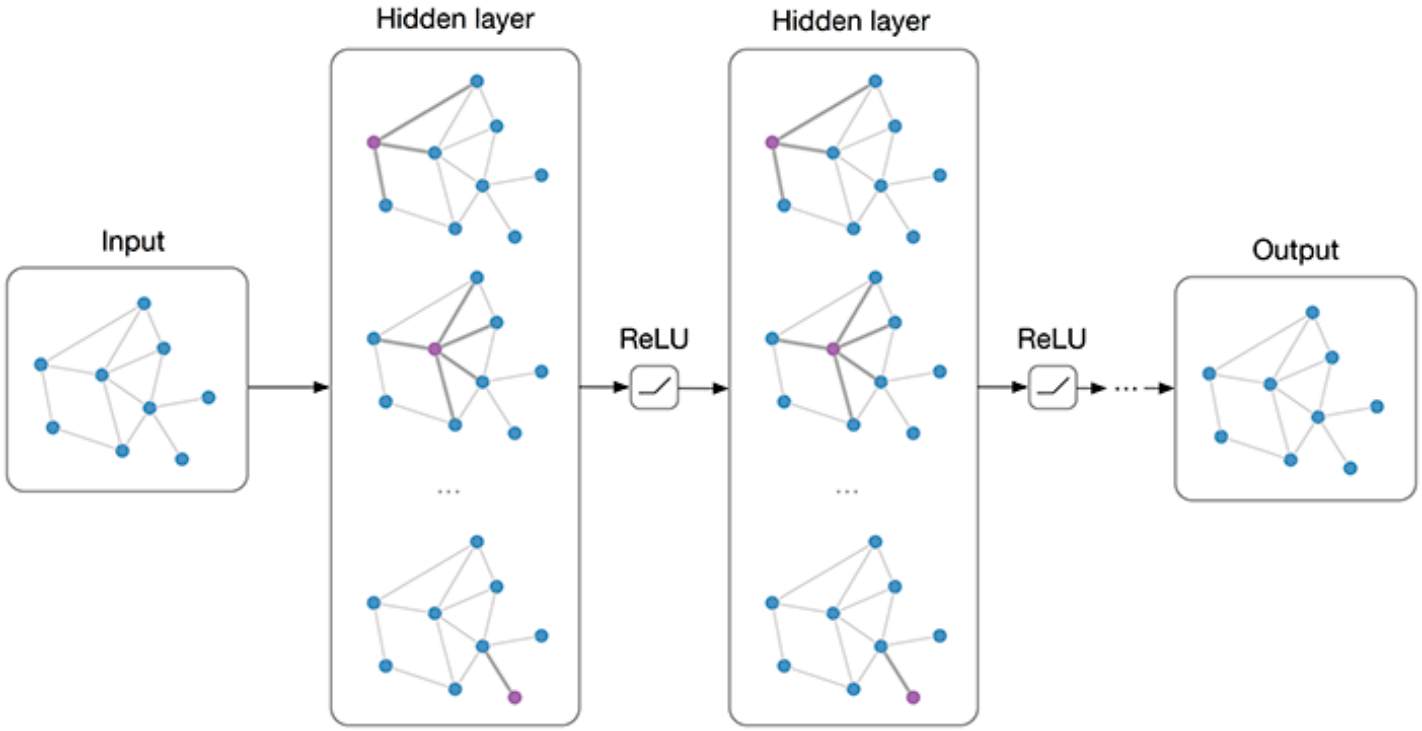
U Global (or master node) attributes
e.g., number of nodes, longest path



V Vertex (or node) attributes
e.g., node identity, number of neighbors

E Edge (or link) attributes and directions
e.g., edge identity, edge weight

U Global (or master node) attributes
e.g., number of nodes, longest path



Principles of GNN

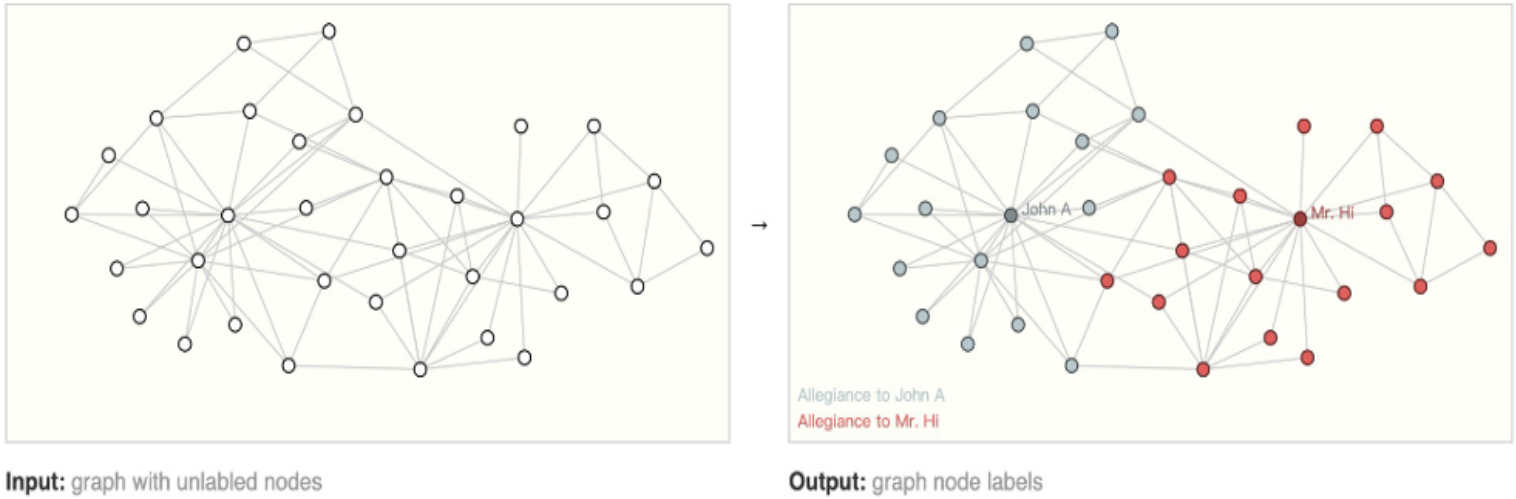
Input:

- Topology (network)
- Node attribution

Output:

- Classification (Label)

Node Classification

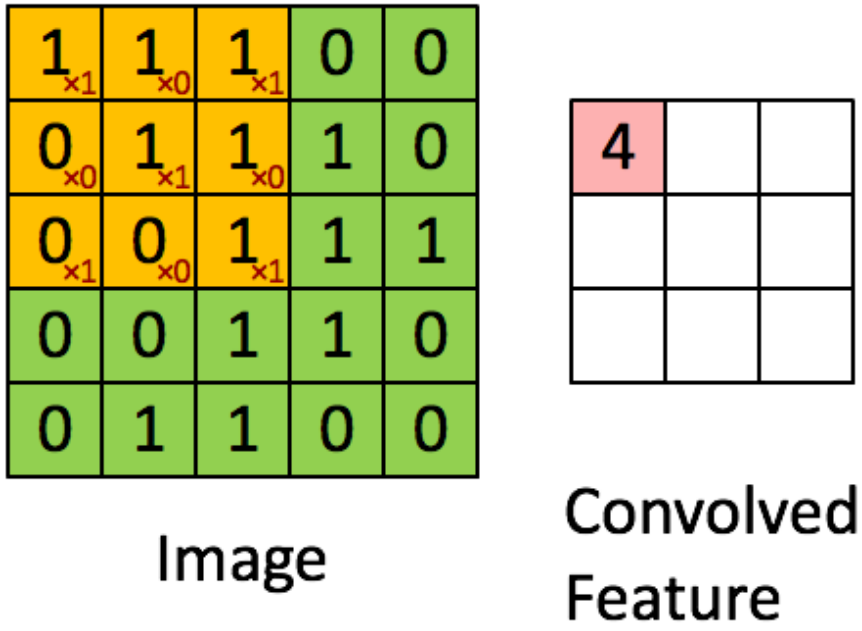


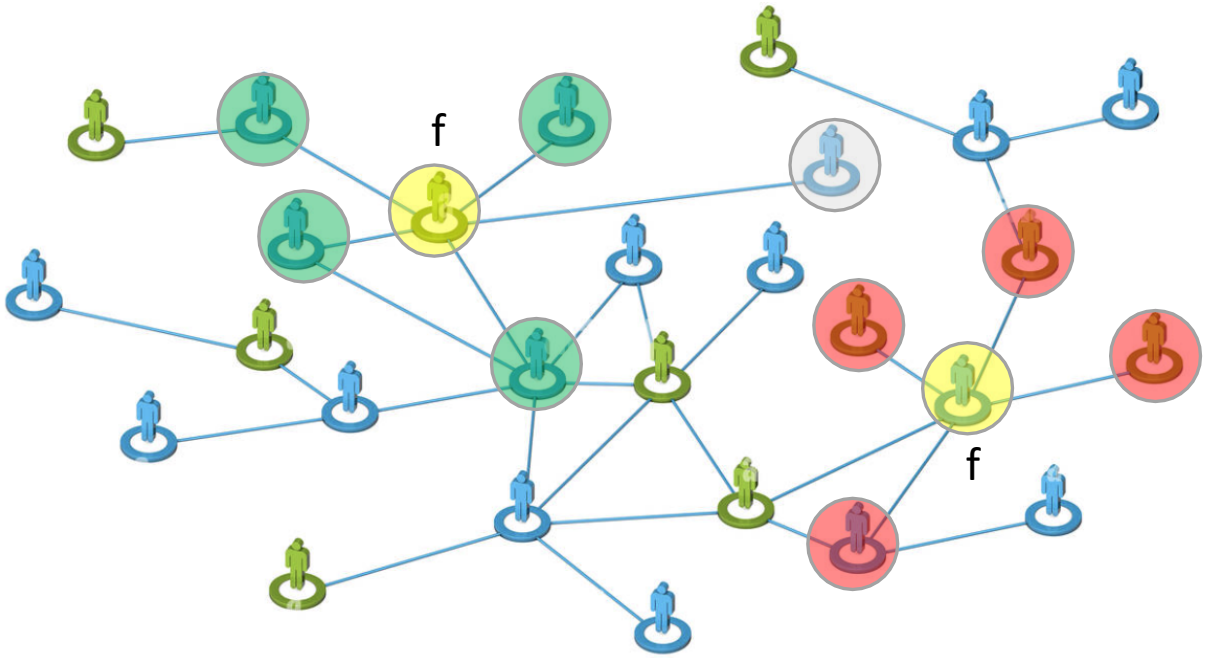
世界是紧密联系的

- 方案1：将每个节点的特征认为相互独立，训练一个MLP
- 方案2：每个节点的分类和自己与邻居的特征都有关

节点对称性：每个节点的信息聚合、分类准则都是一样的！

Node Symmetry





Information Aggregation

Aggregation:

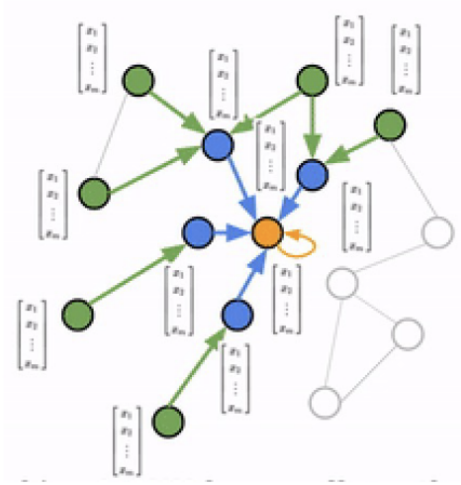
- Use neighbors' information to update my feature

Recursive:

- Aggregation can be recursive
- The orders of aggregation is the layer

Self

First order neighbor



Second order neighbor

Message Passing

$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})\right)$$
$$= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right),$$

Define Two Functions

Aggregate Aggregate information of neighborhood to form message m

Update Update self information according to m and my own feature

Message Passing

$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})\right)$$
$$= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right),$$

Requirement for aggregate：

Permutation invariant $\{1,2,3\} \cong \{1,3,2\} \cong \{2,1,3\} \cong \{2,3,1\} \cong \{3,1,2\} \cong \{3,2,1\}$

Variant Length $\{1,2,3\}, \{1,2\}, \{6\}$

Examples:

$Mean(\{1,2,3\}) = 2,$ $Sum(\{1,2,3\}) = 6,$ $Product(\{1,2,3\}) = 6$

Deep Set

Deep Sets

Manzil Zaheer[1,2], **Satwik Kottur**[1], **Siamak Ravanbhakhsh**[1],
Barnabás Póczos[1], **Ruslan Salakhutdinov**[1], **Alexander J Smola**[1,2]
[1] Carnegie Mellon University [2] Amazon Web Services
{manzilz,skottur,mravanba,bapoczos,rsalakhu,smola}@cs.cmu.edu

Abstract

We study the problem of designing models for machine learning tasks defined on *sets*. In contrast to traditional approach of operating on fixed dimensional vectors, we consider objective functions defined on sets that are invariant to permutations. Such problems are widespread, ranging from estimation of population statistics [1], to anomaly detection in piezometer data of embankment dams [2], to cosmology [3, 4]. Our main theorem characterizes the permutation invariant functions and provides a family of functions to which any permutation invariant objective function must belong. This family of functions has a special structure which enables us to design a deep network architecture that can operate on sets and which can be deployed on a variety of scenarios including both unsupervised and supervised learning tasks. We also derive the necessary and sufficient conditions for permutation equivariance in deep models. We demonstrate the applicability of our method on population statistic estimation, point cloud classification, set expansion, and outlier detection.

https://arxiv.org/pdf/1703.06114.pdf

Deep Set

Aggregation Function must be set function(permutation invariant):

Property 1 *A function* $f : 2^{\mathfrak{X}} \to \mathcal{Y}$ *acting on sets must be permutation* ***invariant*** *to the order of objects in the set,* i.e. *for any permutation* π : $f(\{x_1, \ldots, x_M\}) = f(\{x_{\pi(1)}, \ldots, x_{\pi(M)}\})$.

Theorem 2 *A function* $f(X)$ *operating on a set* X *having elements from a countable universe, is a valid set function,* i.e., ***invariant*** *to the permutation of instances in* X*, iff it can be decomposed in the form* $\rho\left(\sum_{x \in X} \phi(x)\right)$*, for suitable transformations* ϕ *and* ρ.

https://arxiv.org/pdf/1703.06114.pdf

Deep Set

Aggregation Function must be set function(permutation invariant):

Property 1 *A function* $f : 2^{\mathfrak{X}} \to \mathcal{Y}$ *acting on sets must be permutation* ***invariant*** *to the order of objects in the set,* i.e. *for any permutation* π : $f(\{x_1, \ldots, x_M\}) = f(\{x_{\pi(1)}, \ldots, x_{\pi(M)}\})$.

Theorem 2 *A function* $f(X)$ *operating on a set* X *having elements from a countable universe, is a valid set function,* i.e., ***invariant*** *to the permutation of instances in* X, *iff it can be decomposed in the form* $\rho\left(\sum_{x \in X} \phi(x)\right)$, *for suitable transformations* ϕ *and* ρ.

$$\mathbf{m}_{\mathcal{N}(u)} = \mathrm{MLP}_\theta\left(\sum_{v \in N(u)} \mathrm{MLP}_\phi(\mathbf{h}_v)\right)$$

https://arxiv.org/pdf/1703.06114.pdf

Graph Convolutional Network(GCN)

Published as a conference paper at ICLR 2017

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl

ABSTRACT

We present a scalable approach for semi-supervised learning on graph-structured data that is based on an efficient variant of convolutional neural networks which operate directly on graphs. We motivate the choice of our convolutional architecture via a localized first-order approximation of spectral graph convolutions. Our model scales linearly in the number of graph edges and learns hidden layer representations that encode both local graph structure and features of nodes. In a number of experiments on citation networks and on a knowledge graph dataset we demonstrate that our approach outperforms related methods by a significant margin.

https://tkipf.github.io/graph-convolutional-networks/