



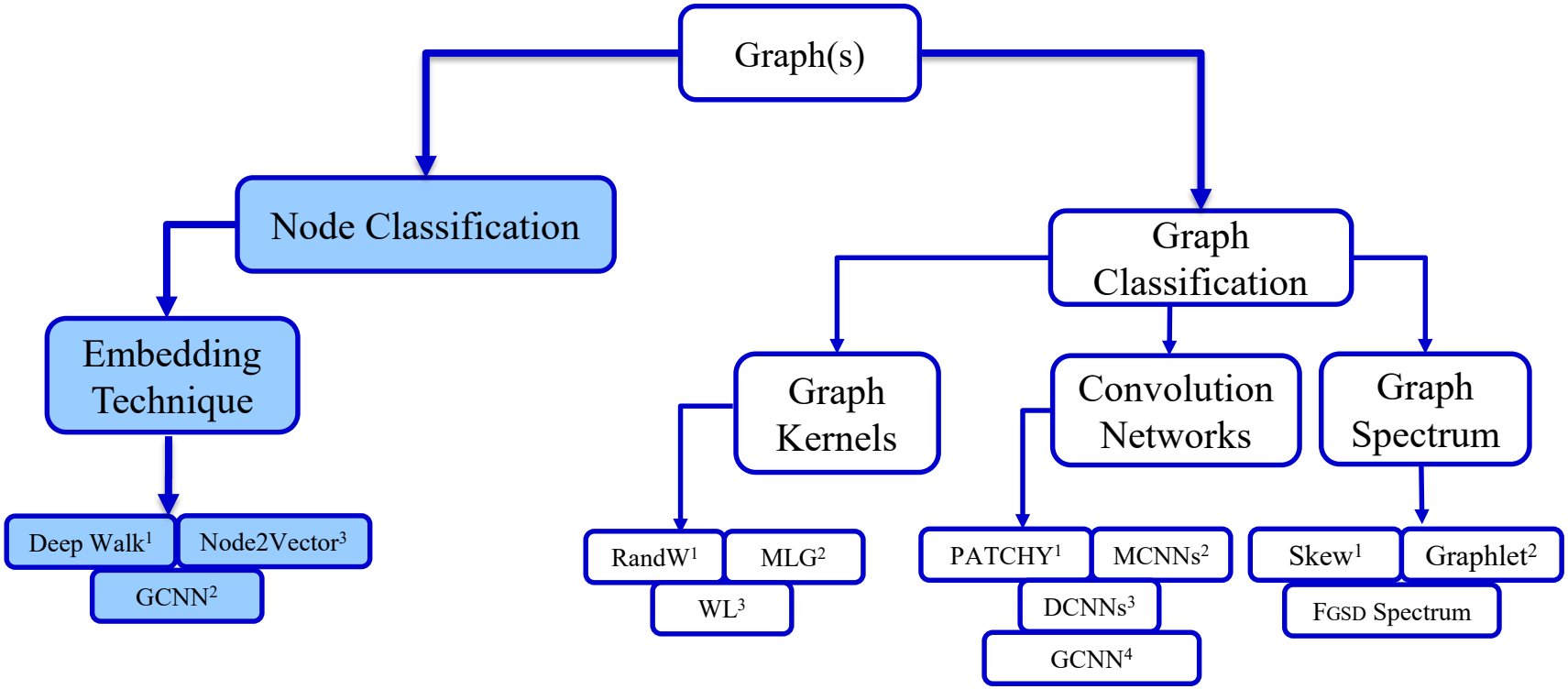
# From word-embedding to node-embedding for learning over graphs

**Fatemeh Sheikholeslami**

DTC and Dept. of ECE, University of Minnesota

March 26, 2018

# Taxonomy of learning over graphs





[1a] Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations." *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*. ACM, New York, NY, Aug. 2014.

[1b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," In *Advances in neural information processing systems*, pp. 3111-3119, Lake Tahoe, USA, 2013.

[2] Thomas N. Kipf, Max Welling, Semi-Supervised Classification with Graph Convolutional Networks., ICLR, Toulon, France, April 2017.

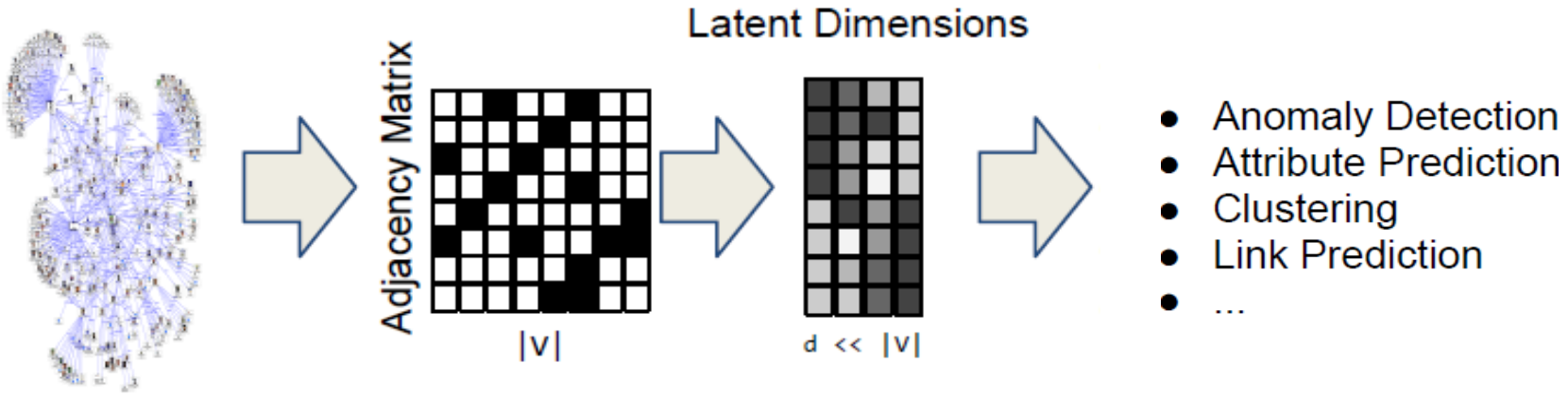
[3] Aditya Grover and Jure Leskovec. "node2vec: Scalable feature learning for networks." *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge Discovery and Data mining*. ACM, San Francisco, CA, Aug. 2016.

# Outline

- Taxonomy of learning over graphs
- Graph learning algorithms inspired by language models
  - ✓ Word2vec (Skip-gram)
  - ✓ DeepWalk
  - ✓ Node2vec
- Graph convolutional networks
- Simulation tests

# Graph Representation

- For machine learning tasks, the first step is to represent the network
  - Such as adjacency matrix
- Feature extraction (node embedding)
  - Transform the adjacency matrix into a lower dimensional latent representation
  - Use deep learning techniques developed for language modeling



# Language modeling

- Representation learning of words from documents
- The learned representations capture inherent structure
  - e.g., learn mapping $\phi : v \in V \rightarrow \mathbb{R}^{|V| \times d}$ such that

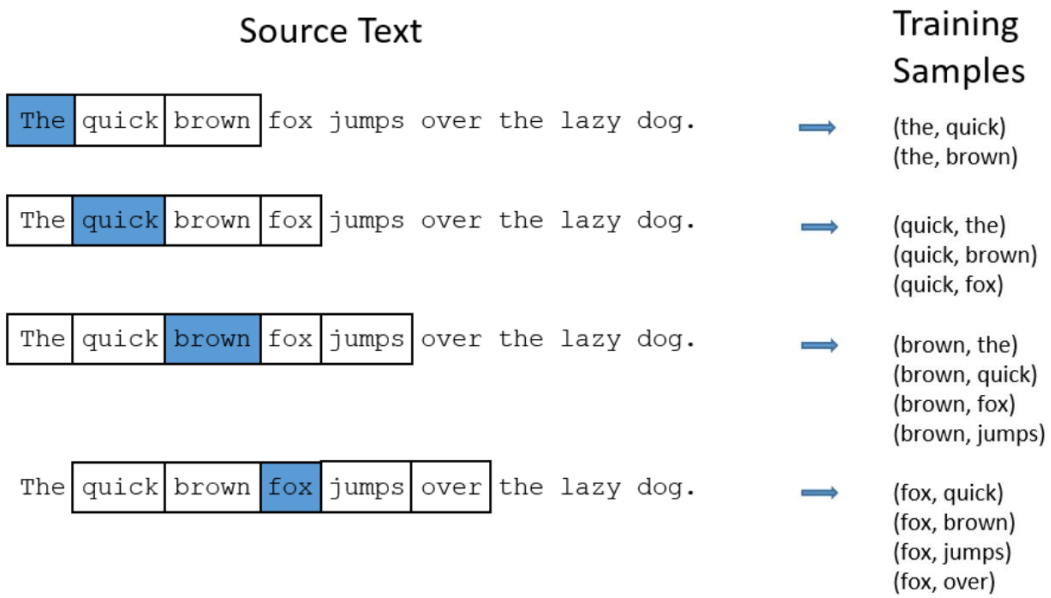
$$\|\phi(\text{rose}) - \phi(\text{daisy})\| < \|\phi(\text{rose}) - \phi(\text{tiger})\|$$

  - Recent approaches are based on word co-occurrence
  - Word2vec
    - ✓ Continuous-bag-of-words
    - ✓ Skip-gram
    - ✓ GloVe

# Word co-occurance maximization

- Consider source text “The quick brown fox jumps over the lazy dog.”
- Skip-gram: with one word, predict surrounding words
  - Max. the likelihood of context words $(w_{t-m}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+m})$ given center word $w_t$

$$\frac{1}{T}\sum_{t=1}^{T} \log P\Big((w_{t+m}, ..., w_{t-1}, w_{t+1}, ..., w_{t+m})|w_t\Big) \simeq \frac{1}{T}\sum_{t=1}^{T}\sum_{j=-m,..,m;j\neq 0} \log P(w_{t+j}|w_t)$$

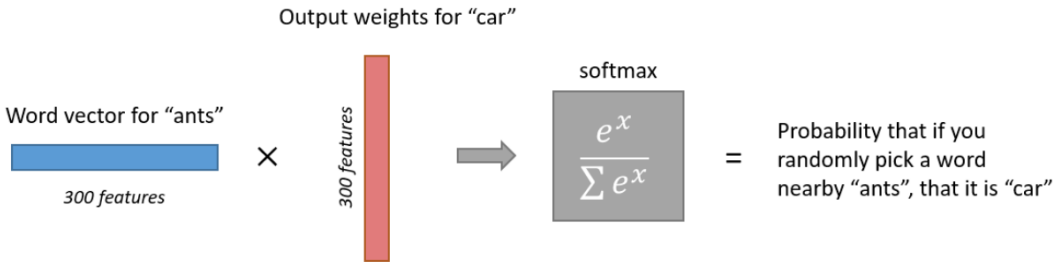



  - Window-size *m* is usually 5-10

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

# Skipgram model

- How to model $\log P(w_{t+j}|w_t)$ ?
- Every word has 2 vectors for representation
  - $\mathbf{v}_w$ : when w is the center word
  - $\mathbf{u}_w$ : when w is the outside word (context word)
  - Softmax function $$P(w_{t+j}|w_t) = \frac{\exp(\mathbf{u}_{w_{t+j}}.\mathbf{v}_{w_t})}{\sum_{w'} \exp(\mathbf{u}_{w'}.\mathbf{v}_{w_t})}$$

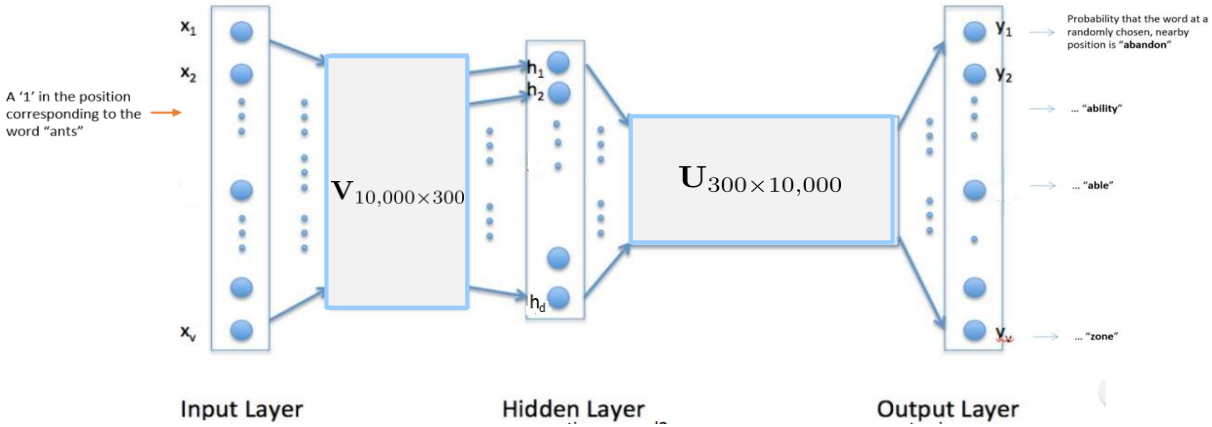


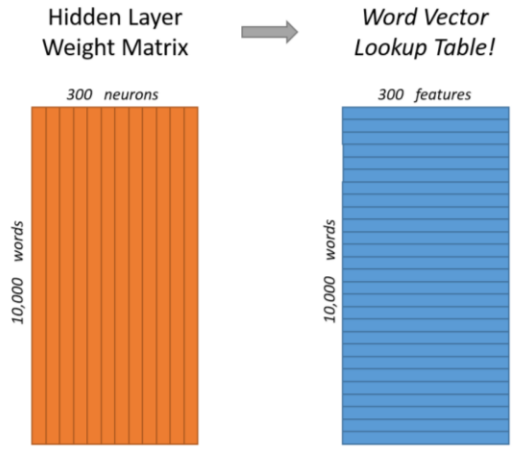

- Entire parameter set $\boldsymbol{\theta} \in \mathbb{R}^{2dV}$

$$\boldsymbol{\theta} = \begin{bmatrix} v_{\text{abandon}} \\ .. \\ v_{\text{zone}} \\ u_{\text{abandon}} \\ .. \\ u_{\text{zone}} \end{bmatrix}$$

# Neural network model

- Consider a vocabulary size of $|V| = 10,000$ and set d=300
- Shallow (one hidden layer) neural network model




- Neural network input: one-hot vector corresponding to the "center word" of interest
- Maximization objective $J(\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T} J_t(\boldsymbol{\theta})$
- Gradient descent update rule $\boldsymbol{\theta}^{new} = \boldsymbol{\theta}^{old} - \eta \nabla J(\boldsymbol{\theta}^{old})$
  - ✓ Stochastic gradient descent: randomly pick sample *t* $\boldsymbol{\theta}^{new} = \boldsymbol{\theta}^{old} - \eta \nabla J_t(\boldsymbol{\theta}^{old})$

Final representation

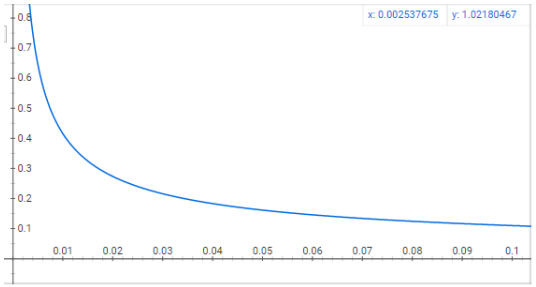
$$\boldsymbol{\Phi} = \mathbf{U}^{\top} + \mathbf{V}$$

# Algorithmic tweaks

- Word pairs and phrases
  - Find words that appear frequently together, e.g., “New York” and “Boston Globe”.
  - “Phrase learning” is performed beforehand

- Subsampling of frequent words
  - Counter the imbalance between rare and frequent words
  - Sample word $w_i$ w/ frequency $f(w_i)$ w. p. $P(w_i)$

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \qquad t \simeq 10^{-5}$$



# Algorithmic tweaks- negative sampling

- Typically extremely large vocabulary set ( $|V| \simeq 10^5 - 10^7$ )
- Expensive to calculate the gradient of $P(w_o|w_c) = \frac{\exp(\mathbf{u}_{w_o}.\mathbf{v}_{w_c})}{\sum_{w'} \exp(\mathbf{u}_{w'}.\mathbf{v}_{w_c})}$
  - All $|V|d$ weights in the second layer of the NN (***U***) are involved and updated!
    - Recall that the output of the network is a one-hot vector.
    - That is, one output is 1, and *all* of the other millions of output neurons are 0.
    - Randomly select just a small number of “negative” words (say 5) to update the weights for.
      - “negative” words are the ones for which we want the network to output a 0
      - “positive” word is the word for which the network outputs 1
  - Approximate the normalizer via few “negative samples”
  - Probabilistic selection of negative samples $P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n} \left(f(w_j)^{3/4}\right)}$

# Word analogies

- Nearest words to frog

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

litoria

leptodactylidae

rana

eleutherodactylus

- Test for linear relationships

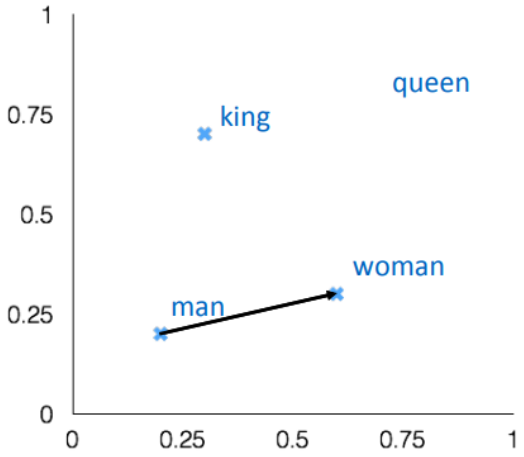
a:b :: c:?

$$d = \arg\max_{x} \frac{(\phi_b - \phi_a + \phi_c)^\top \phi_x}{\|\phi_b - \phi_a + \phi_c\|}$$

- Example

man:woman :: king:?

| | | |
|---|---|---|
| + | king | [ 0.30 0.70 ] |
| - | man | [ 0.20 0.20 ] |
| + | woman | [ 0.60 0.30 ] |
| | queen | [ 0.70 0.80 ] |



# Test results

- Semantic-Syntactic Word Relationship test set
  - Five types of semantic questions
  - Nine types of syntactic questions

| Type of relationship | Word Pair 1 | | Word Pair 2 | |
|---|---|---|---|---|
| Common capital city | Athens | Greece | Oslo | Norway |
| All capital cities | Astana | Kazakhstan | Harare | Zimbabwe |
| Currency | Angola | kwanza | Iran | rial |
| City-in-state | Chicago | Illinois | Stockton | California |
| Man-Woman | brother | sister | grandson | granddaughter |
| Adjective to adverb | apparent | apparently | rapid | rapidly |
| Opposite | possibly | impossibly | ethical | unethical |
| Comparative | great | greater | tough | tougher |
| Superlative | easy | easiest | lucky | luckiest |
| Present Participle | think | thinking | read | reading |
| Nationality adjective | Switzerland | Swiss | Cambodia | Cambodian |
| Past tense | walking | walked | swimming | swam |
| Plural nouns | mouse | mice | dollar | dollars |
| Plural verbs | work | works | speak | speaks |

- Accuracy on (subset of) the Semantic-Syntactic Word Relationship test set.

| Dimensionality / Training words | 24M | 49M | 98M | 196M | 391M | 783M |
|---|---|---|---|---|---|---|
| 50 | 13.4 | 15.7 | 18.6 | 19.1 | 22.5 | 23.2 |
| 100 | 19.4 | 23.1 | 27.8 | 28.7 | 33.4 | 32.2 |
| 300 | 23.2 | 29.2 | 35.3 | 38.6 | 43.7 | 45.9 |
| 600 | 24.0 | 30.1 | 36.5 | 40.8 | 46.6 | 50.4 |

- Comparison of architectures using models trained on the same data, with d=640

| Model Architecture | Semantic-Syntactic Word Relationship test set | | MSR Word Relatedness |
|---|---|---|---|
| | Semantic Accuracy [%] | Syntactic Accuracy [%] | Test Set [20] |
| RNNLM | 9 | 36 | 35 |
| NNLM | 23 | 53 | 47 |
| CBOW | 24 | 64 | 61 |
| Skip-gram | 55 | 59 | 56 |

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," In *Advances in neural information processing systems*, pp. 3111-3119, Lake Tahoe, USA, 2013.

# From language modeling to graphs

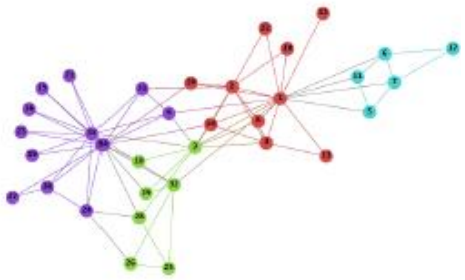
- Correspondence
  - Words <--> Nodes
  - Sentences <--> Node sequences

- Generating node sequences
  - Using random walks
    - short random walks = sentences

- Connection
  - Vertex frequency in random walks on scale-free graphs follows a power law.
  - Words frequency in a natural language corpus follows a power law.

# Deepwalk

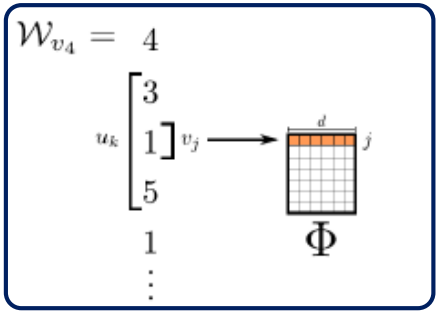
1. Input: Graph



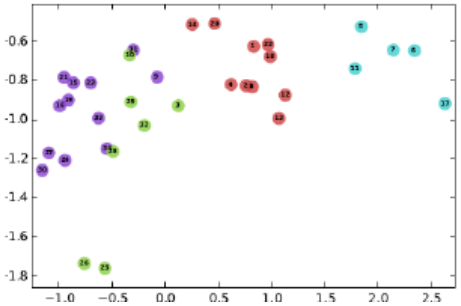
2. Random walk

$$\mathcal{W}_{v_4} = v_4 \rightarrow v_3 \rightarrow v_1 \rightarrow v_5 \rightarrow v_1 \rightarrow v_{46} \rightarrow v_{51} \rightarrow v_{89}$$

3. Representation mapping




4. Output: Representation




5. Representation-based inference such as classification, clustering, etc.

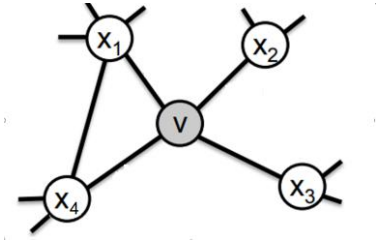
Bryan Perozzi,, Rami Al-Rfou, and Steven Skiena, "Deepwalk: Online learning of social representations," *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*. ACM, New York, NY, Aug. 2014.

# Random walks

- Probability of going from node v to node x

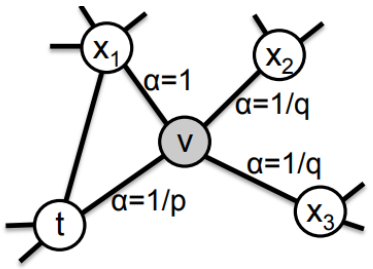
$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v,x) \in E \\ 0 & \text{otherwise} \end{cases}$$

  - Common choice $\pi_{vx} = w_{vx}$




- 2nd order random walk [Grover et al. 2016)]
  - Consider a random walk that just traversed edge (t, v) and now resides at node v
  - Probability of going to node x is set to $\pi_{vx} \sim \alpha_{p,q}(t,x) w_{vx}$

$$\alpha_{p,q}(t,x) \sim \begin{cases} 1/p & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ 1/q & \text{if } d_{tx} = 2 \end{cases}$$




    - p: Return parameter
    - q: In-out parameter

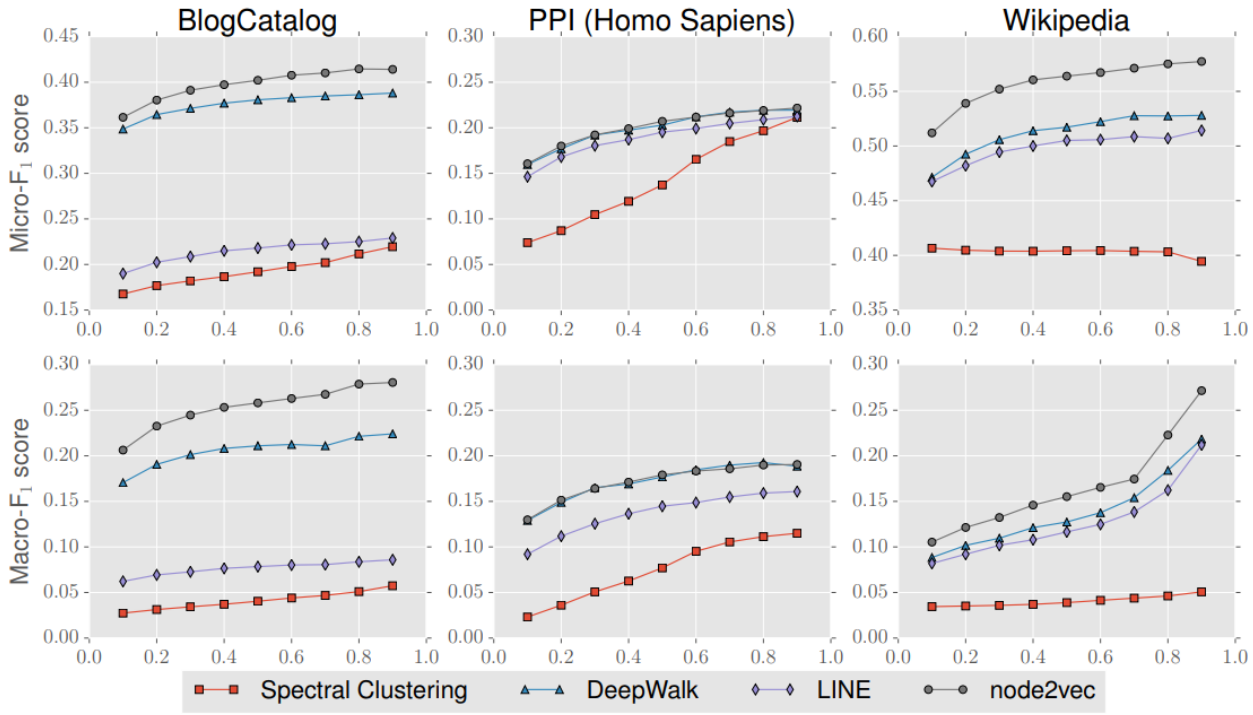

Aditya Grover and Jure Leskovec. "node2vec: Scalable feature learning for networks." *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data mining*. ACM, San Francisco, CA, Aug. 2016.

# Test results

- Datasets

| | BlogCatalog | PPI | Wikipedia |
|---|---|---|---|
| N | 10,312 | 3,890 | 4,777 |
| \|E\| | 333,983 | 76,584 | 184,812 |
| K | 39 | 50 | 40 |

- Performance vs. ratio of available labels



# Roadmap

- Taxonomy of learning over graphs
- Graph learning algorithms inspired by NLP techniques
  - Skipgram
  - DeepWalk
  - Node2vec
- Graph convolutional networks
- Simulation tests

# Learning over featured networks

- Consider a graph(known Adj.) of partly labeled N nodes w/ features $\mathbf{X} \in \mathbb{R}^{N \times F}$

- Common practice

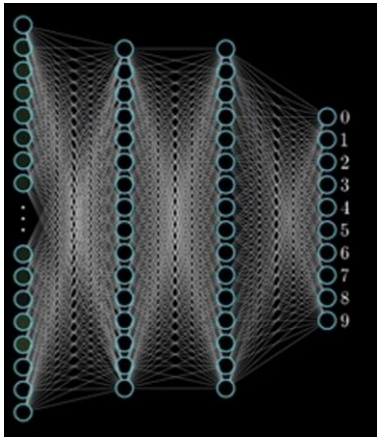
  minimize $\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{\text{reg}}$ , e.g. learn function *f(.)*, w/ $\mathcal{L}_0 = \text{fitting term}$

  $$\mathcal{L}_{\text{reg}} = \sum_{i,j} A_{ij} \| f(\mathbf{x}_i) - f(\mathbf{x}_j) \|^2$$

- "Graph convolutional" approach

  - Use graph structure directly via a neural network model f(***X, A***)

# Graph convolutional networks



- Model
  - Neural network of depth *L*, with $\mathbf{X} \in \mathbb{R}^{N \times F}$ as input
  - $\mathbf{H}^l \in \mathbb{R}^{N \times D_l}$: Output matrix of the *l*-th layer

$$\mathbf{H}^{l+1} = \sigma(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$$

- $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ , $\tilde{\mathbf{D}}_{ii} = \sum_i \tilde{A}_{ij}$ , $\mathbf{H}^{(0)} = \mathbf{X}$ , $\sigma(.)$ : activation fun.

- Semi-supervised node classification
  - Given one-hot labels $\mathbf{Y}$ of size $N_L \times K$ for $N_L$ nodes

$$\min_{\mathbf{W}_0, \mathbf{W}_1} -\sum_{n=1}^{N_L}\sum_{k=1}^{K} Y_{nk} \ln Z_{nk} \qquad \mathbf{Z} = f(\mathbf{X}, \mathbf{A}) = \mathrm{softmax}\Big(\check{\mathbf{A}}\,\mathrm{Relu}(\check{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)})\,\mathbf{W}^{(1)}\Big)$$

  - No explicit graph-based regularization

Thomas N. Kipf and Max Welling, "Semi-Supervised Classification with Graph Convolutional Networks," ICLR, Toulon, France, April 2017.

# Test results

- Datasets

| Dataset | Type | Nodes | Edges | Classes | Features | Label rate |
|---|---|---|---|---|---|---|
| Citeseer | Citation network | 3,327 | 4,732 | 6 | 3,703 | 0.036 |
| Cora | Citation network | 2,708 | 5,429 | 7 | 1,433 | 0.052 |
| Pubmed | Citation network | 19,717 | 44,338 | 3 | 500 | 0.003 |
| NELL | Knowledge graph | 65,755 | 266,144 | 210 | 5,414 | 0.001 |

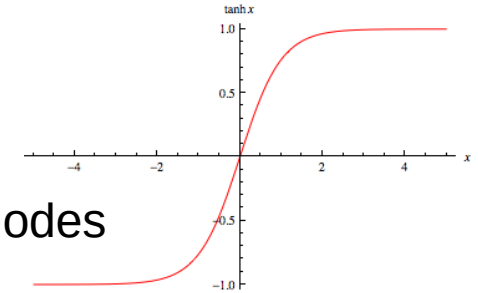
- Classification accuracy (in percent)

| Method | Citeseer | Cora | Pubmed | NELL |
|---|---|---|---|---|
| ManiReg [3] | 60.1 | 59.5 | 70.7 | 21.8 |
| SemiEmb [28] | 59.6 | 59.0 | 71.1 | 26.7 |
| LP [32] | 45.3 | 68.0 | 63.0 | 26.5 |
| DeepWalk [22] | 43.2 | 67.2 | 65.3 | 58.1 |
| ICA [18] | 69.1 | 75.1 | 73.9 | 23.1 |
| Planetoid* [29] | 64.7 (26s) | 75.7 (13s) | 77.2 (25s) | 61.9 (185s) |
| **GCN** (this paper) | **70.3** (7s) | **81.5** (4s) | **79.0** (38s) | **66.0** (48s) |
| GCN (rand. splits) | $67.9 \pm 0.5$ | $80.1 \pm 0.5$ | $78.9 \pm 0.7$ | $58.4 \pm 1.7$ |

# Semisupervised node embedding via GCN

- Can be modified to serve as a feature extractor for nodes in a graph

$$\mathbf{Z} = f(\mathbf{X}, \mathbf{A}) = \mathrm{softmax}\Big(\check{\mathbf{A}}\,\mathrm{Tanh}\Big(\check{\mathbf{A}}\,\mathrm{Tanh}(\check{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)})\,\mathbf{W}^{(1)}\Big)\mathbf{W}^{(2)}\Big)$$

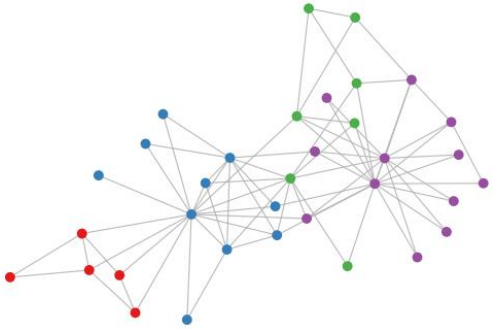


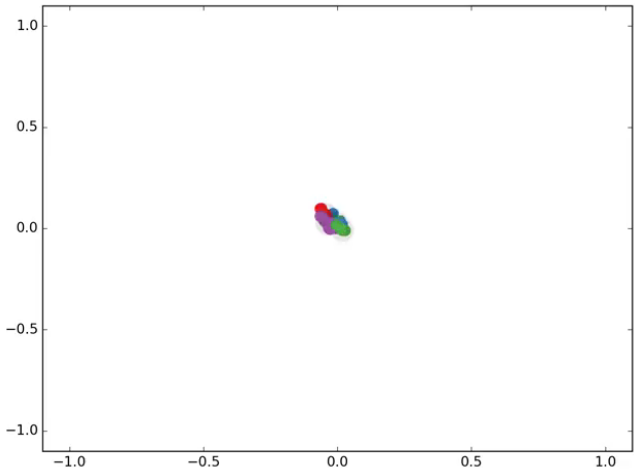

- Train model via cross-entropy minimization on the labeled nodes

- The hidden layer output is interpreted as the embedding vector

- Example: Karate club (with no features)

$$N = 34$$
$$D_1 = 4$$
$$D_2 = 2$$
$$D_3 = 4$$
$$\mathbf{X} = \mathbf{I}_{34}$$



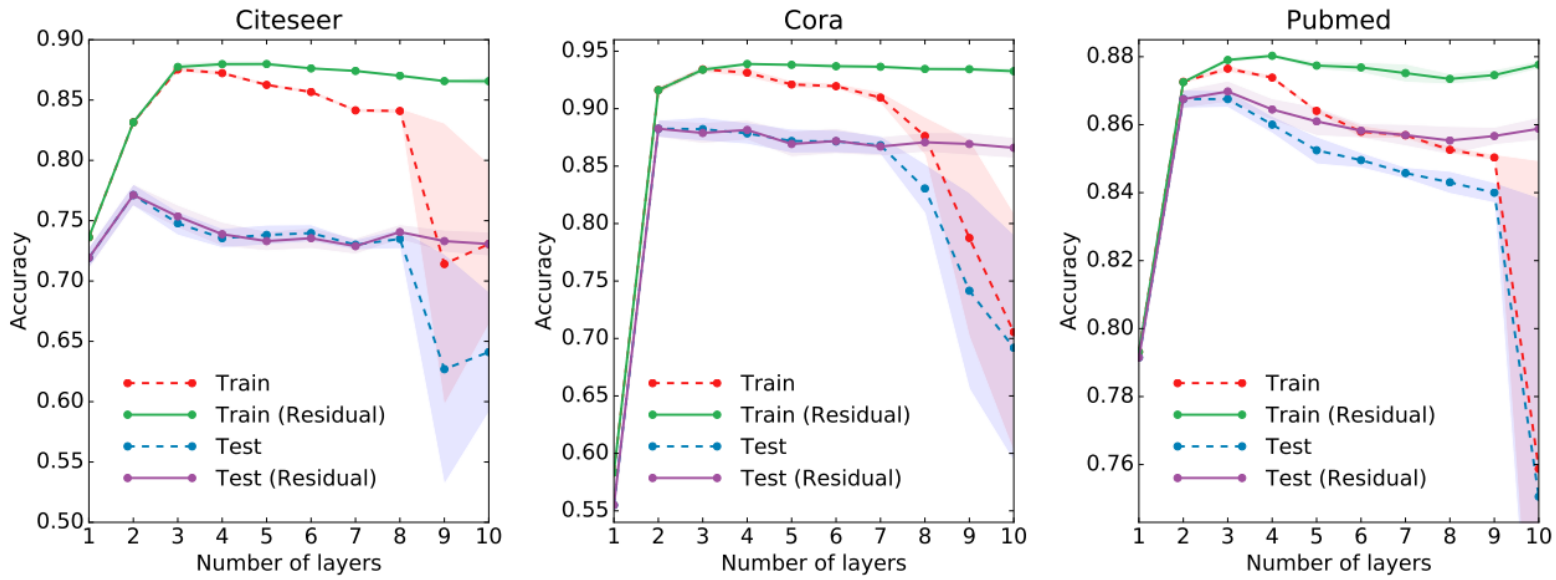


# Effects of model depth

- ResNet models for better performance on deep neural networks

$$\mathbf{H}^{l+1} = \sigma(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) + \mathbf{H}^{(l)}$$

- Performance



# Summary

- Node embedding via language modeling tools
  - Skip-gram
  - DeepWalk
  - Node2vec
- Node embedding via GCN
  - Semi-supervised classification
  - Embedding via hidden layer outputs

*Thank you!*