CentraleSupélec

ESSEC BUSINESS SCHOOL

Network Science Analytics

Option Applied Math and M.Sc. in DSBA

Lecture 6A

Representation learning in graphs

Fragkiskos Malliaros

Friday, March 9, 2018

Announcements

- The second assignment is out
  - Due on April 1st at 23:00

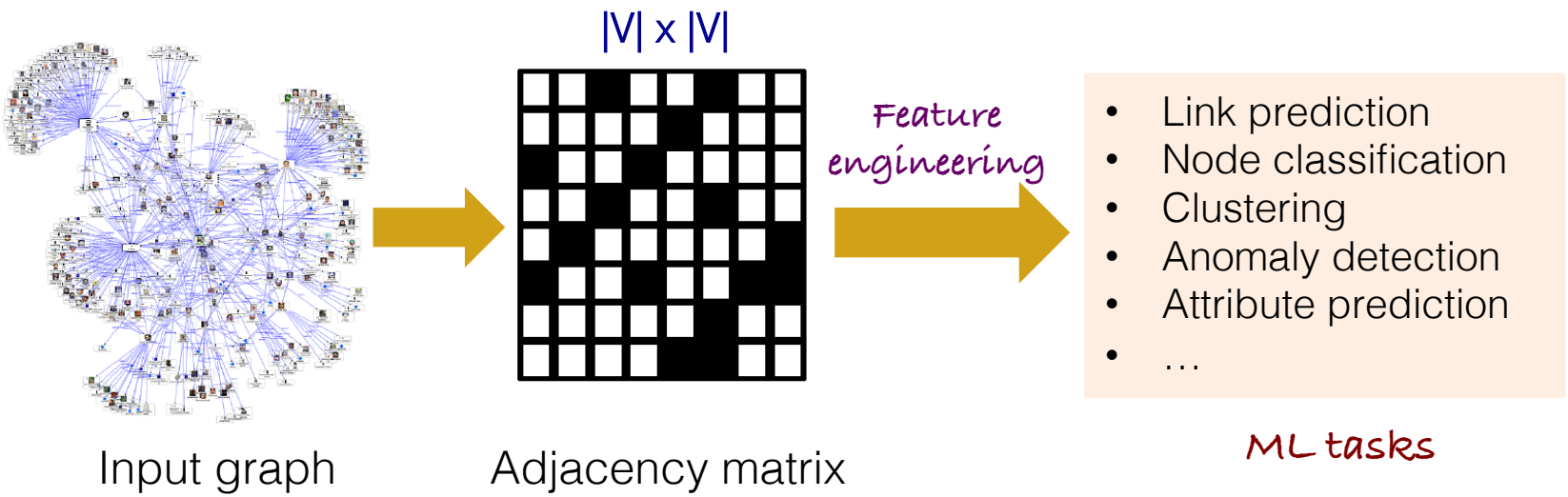
Acknowledgements

- The lecture is partially based on material by
  - Jure Leskovec, Stanford University
  - Aaron Clauset, CU Boulder
  - Manos Papaggelis, York University
  - Gonzalo Mateos, University of Rochester
  - Albert-László Barabási, Northeastern University
  - Christos Faloutsos, CMU
  - Panagiotis Tsaparas, University of Ioannina
  - Danai Koutra, University of Michigan
  - B Aditya Prakash, Virginia Tech
  - Brian Perozzi, StonyBrook University
  - R. Zafarani, M. A. Abbasi, and H. Liu, Social Media Mining: An Introduction, Cambridge University Press, 2014. Free book and slides at http://socialmediamining.info/

Thank you!

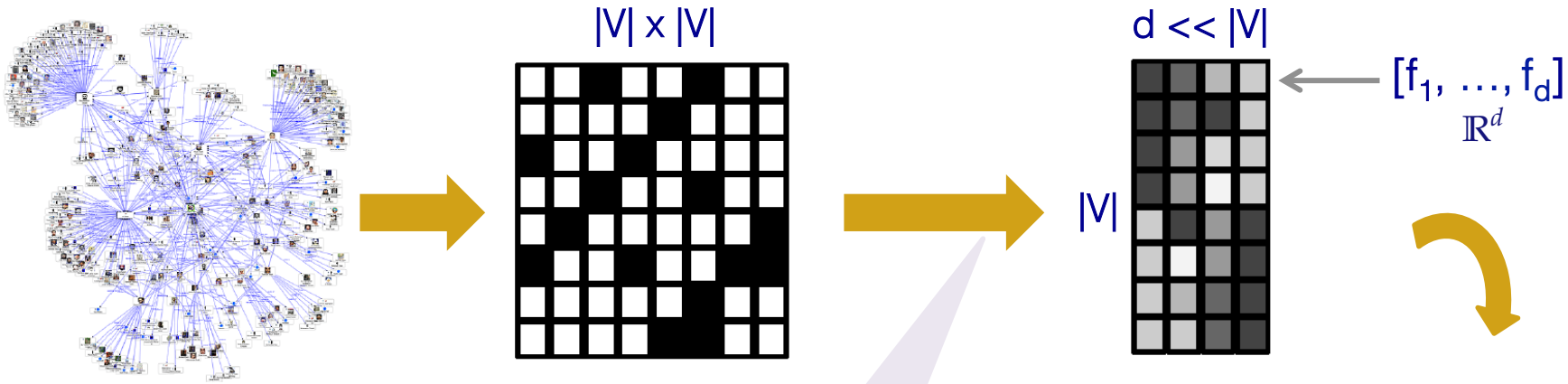
Feature Extraction From Graphs

- The first step of any ML algorithm for graphs is to extract graph features
  - Node features (e.g., degree)
  - Pairs of nodes (e.g., number of common neighbors)
  - Groups of nodes (e.g., community assignments)



Graph Representation

- Create features by transforming the graph into a lower dimensional latent representation

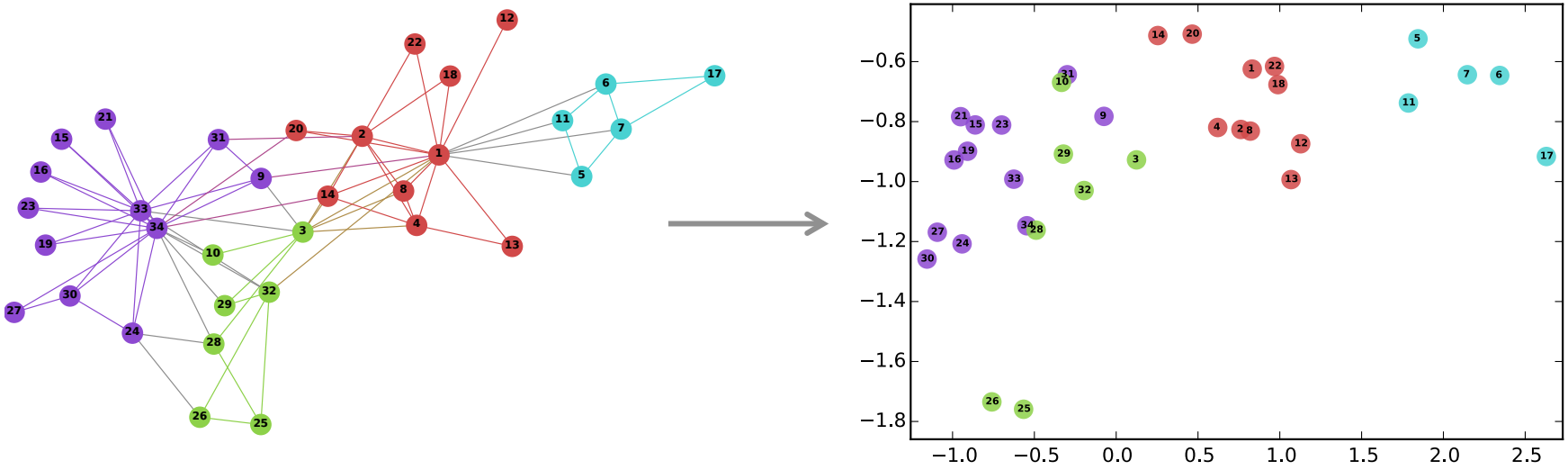


How to learn a latent representation of a graph?

- Link prediction
- Node classification
- Clustering
- Anomaly detection
- Attribute prediction
- …

ML tasks

Example: Community Detection



Input graph

Learn latent representation

Other applications: classification, link prediction, …

[Perozzi et al., KDD '14]

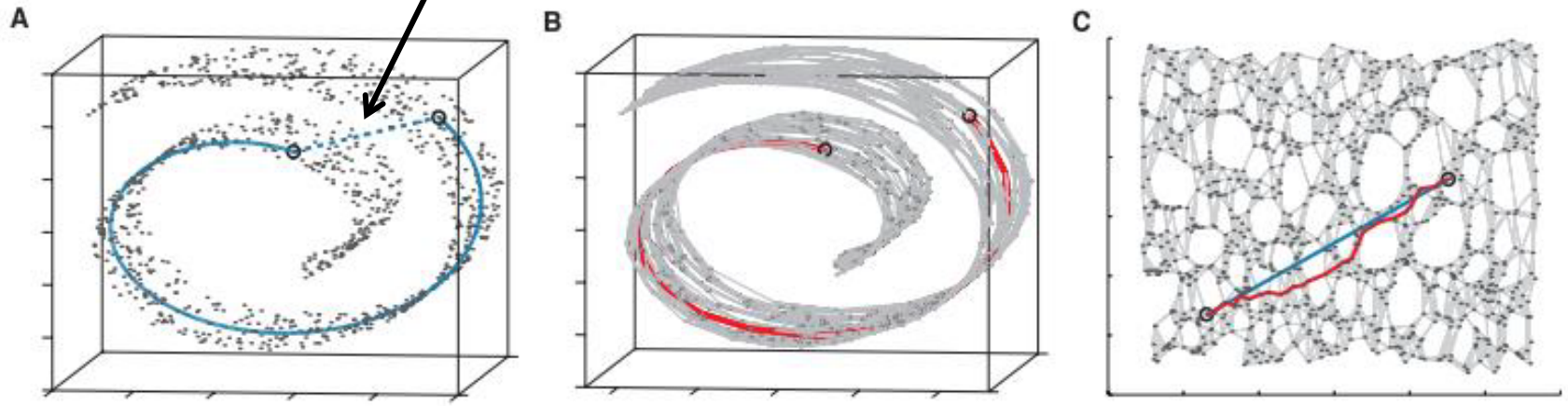
Non-linear dimensionality reduction

Isomap and LLE

Graph Embedding – Isomap (1/2)

- Goal: Find the low-dimensional representation for a dataset
  - Approximately preserve the geodesic distances of the data pairs

The Euclidean distance may not reflect the intrinsic similarity between two data points



Swiss-role dataset

Neighborhood graph (approximation to the geodesic path – red curve)

Two-dimensional embedding (approximation to the geodesic distance – blue curve)

Graph Embedding – Isomap (2/2)

1. Construct neighborhood graph
   - Using connectivity algorithms, such as k-NNs

2. Find the shortest path distances between nodes in the graph
   - Good estimate of the geodesic distance between two points
   - Create distance matrix D

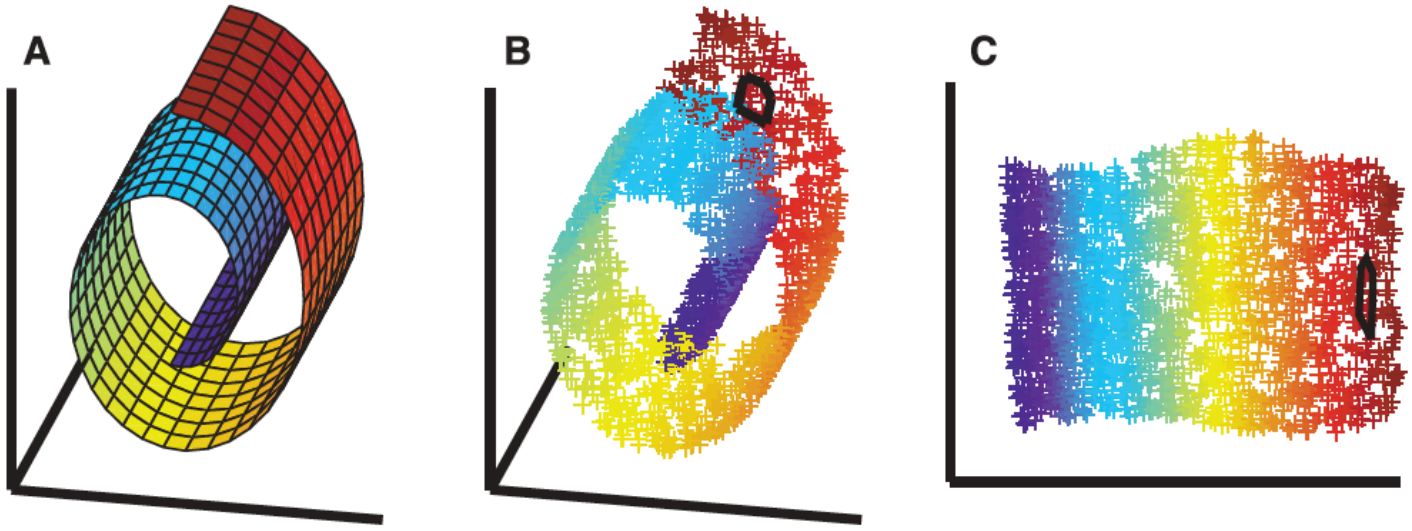
3. Construct d-dimensional embedding
   - Apply Multidimensional Scaling (MDS)
   - Preserves the pairwise distances between points – geodesic distances instead of Euclidean distances

     (Use the top-d eigenvalues and the corresponding eigenvectors of D to construct the embedding)

[Tanenbaum et al., Science '00]

LLE: Locally Linear Embedding (1/2)

- Goal: Maps the data to a lower dimensional space that preserves the neighborhoods of the points
  - Nonlinear dimensionality reduction



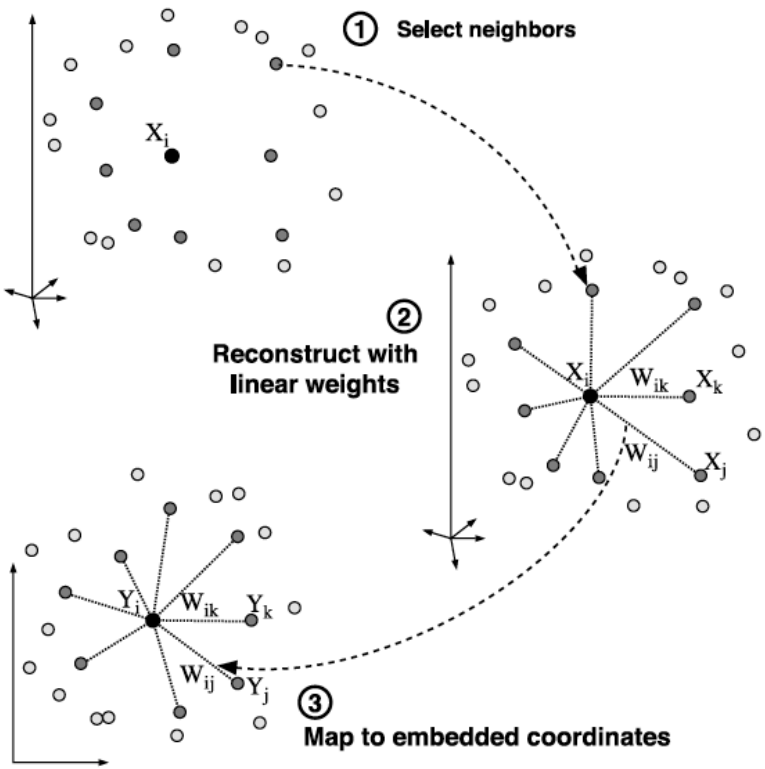
Swiss-role dataset

The color coding illustrates neighborhood-preserving mapping achieved by LLE

LLE: Locally Linear Embedding (2/2)

1. Find a set of the nearest neighbors of each data point X_i
   - E.g., K-NNs
2. Compute a set of weights W that best linearly reconstruct X_i from its neighbors
   - X_i as a linear combination of its neighbors
3. Compute the low-dimensional embedding vectors Y_i by minimizing the embedding cost function

$$\Phi(Y) = \sum_i \left| Y_i - \sum_j W_{ij} - Yj \right|^2$$



[Roweis and Saul, Science '00]

Overview

- A plethora of methods for unsupervised feature learning on graphs
  - Traditionally designed for non-graph data
  - Most of these approaches involve eigendecomposition of a data matrix
  - Expensive for large graphs

- The rich network structures and properties of real networks are not incorporated
  - Poor performance on various prediction tasks over networks

DeepWalk algorithm

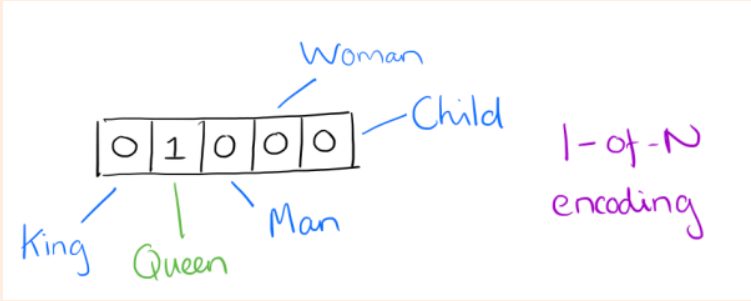
KDD ‘14

Word Embeddings: word2vec

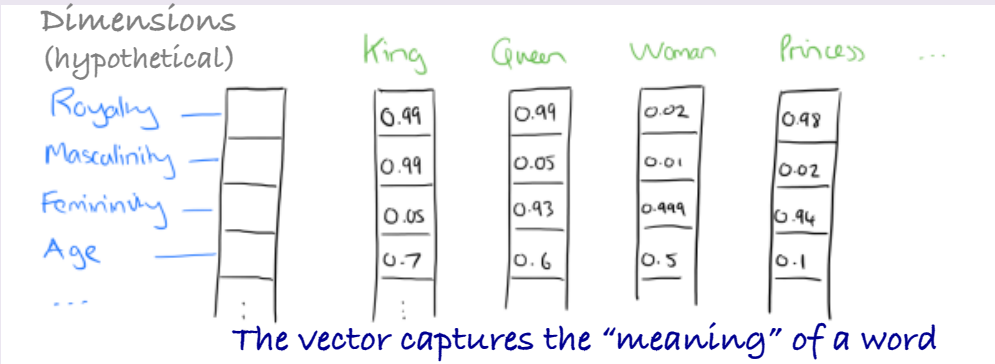
- Input: a large corpus of text
- Output: a vector representation of each word in the corpus
- Typically, shallow two-layer neural networks are used

- Position of word vectors in the vector space
  - Words that share common contexts in the corpus, are located in close proximity to one another in the new space

One-hot encoding



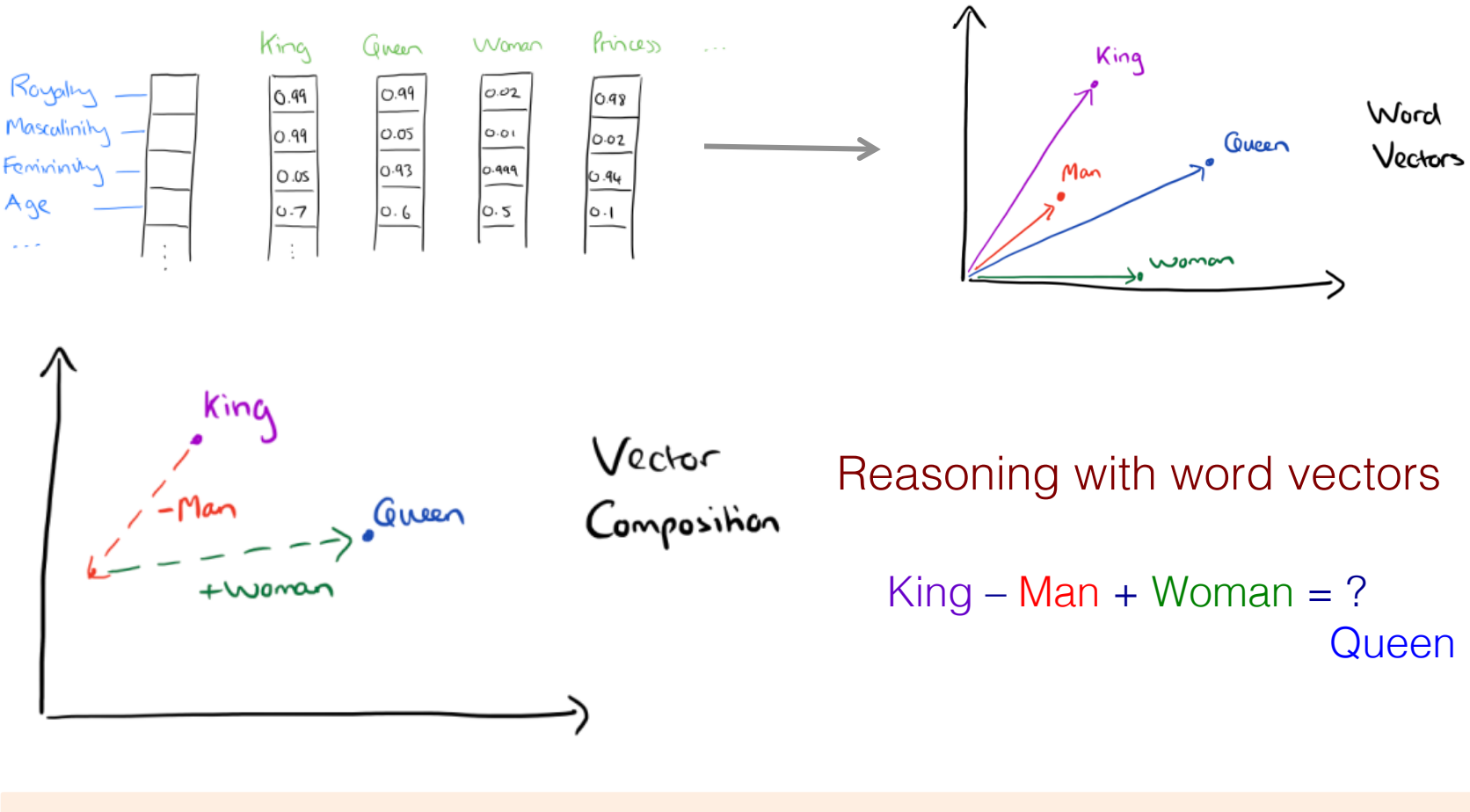
Distributed representation



[Mikolov et al. '13]

https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/

Word Vectors



Reasoning with word vectors

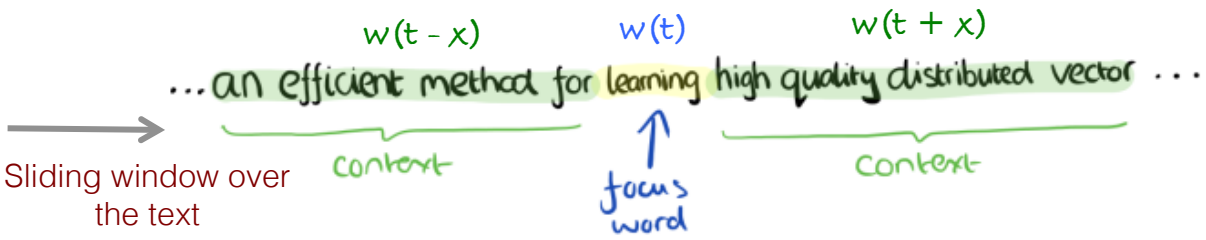
King – Man + Woman = ?

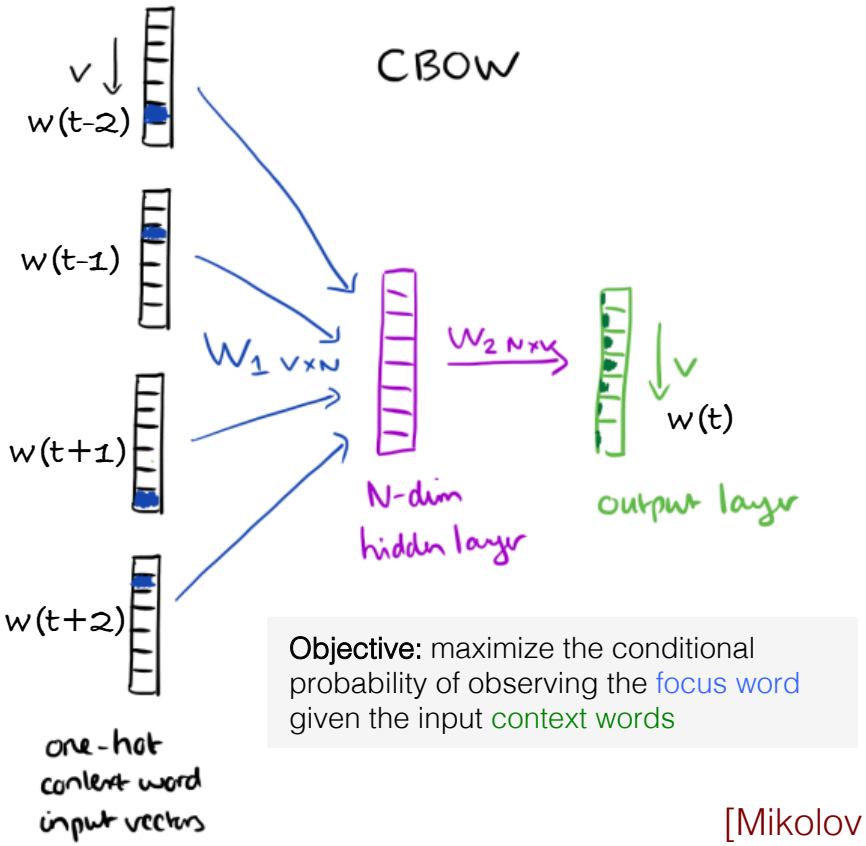
Queen

- Two basic model architectures for learning word representations
  - Continues Bag-of-Words (CBOW) model: predicts the current word using surrounding contexts
  - SkipGram model: predicts surrounding words using the current word

How to Learn Distributed Word Vectors

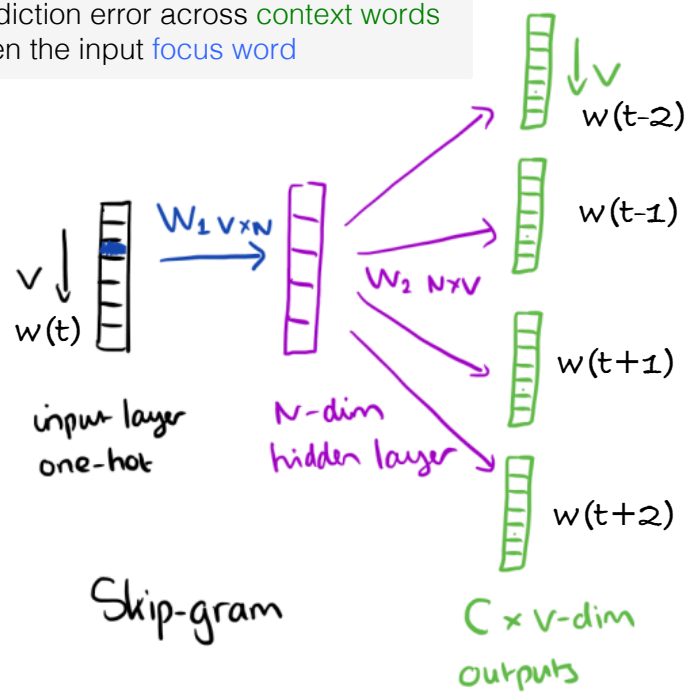
The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of precise syntactic and semantic word relationships.





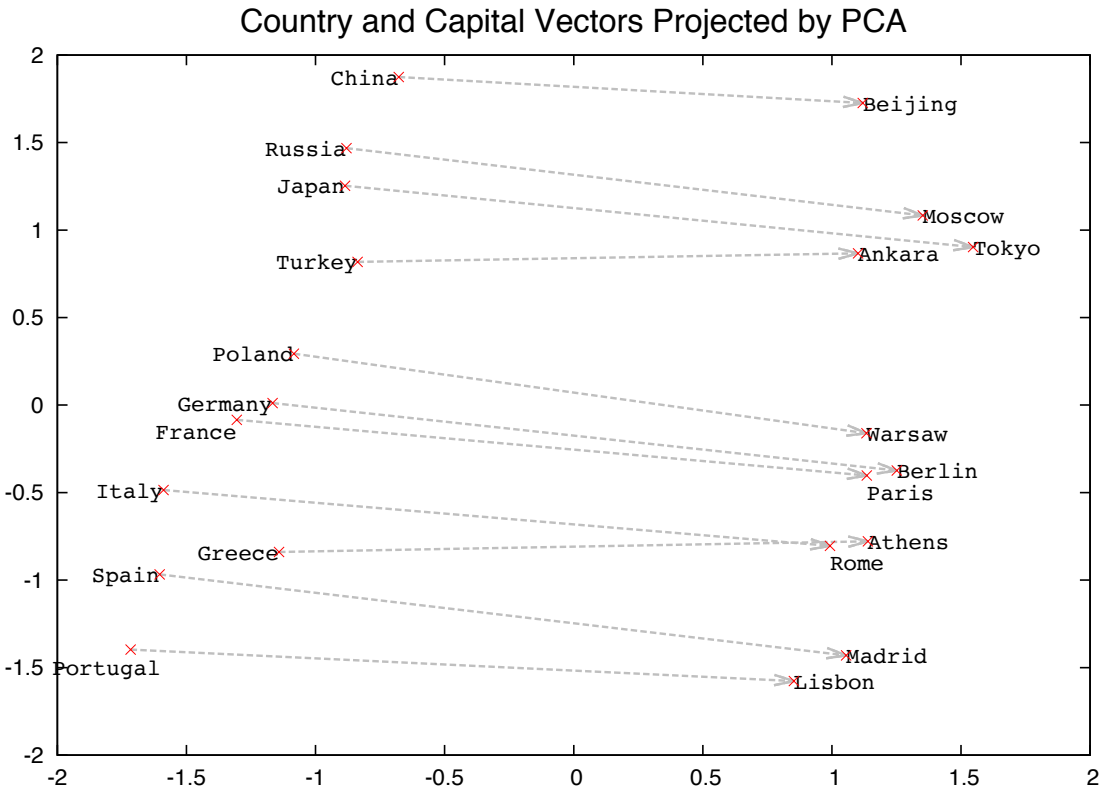
Objective: maximize the conditional probability of observing the focus word given the input context words

Objective: minimize the summed prediction error across context words given the input focus word



[Mikolov et al. '13]

Example of Word Vectors



- Ability of the model to automatically organize concepts and learn implicitly the relationships between them
- During the training we did not provide any supervised information about what a capital city means

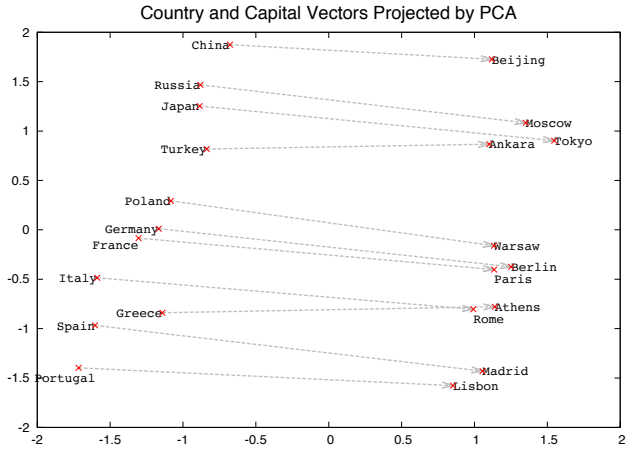
[Mikolov et al. '13]

SkipGram for Graphs?

- SkipGram: learns low dimensional embeddings of words
  - Based on word co-occurrence
  - Distributional hypothesis: Similar words appear in similar contexts
  - An embedding encodes information about what words are likely to be nearby a given word

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

We expect that the learned representations capture inherent structure

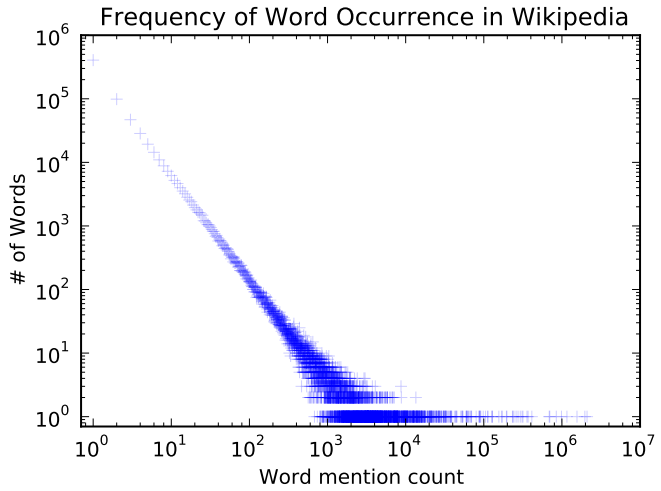


- A graph can be viewed as a document where the nodes correspond to words
  - How to formalize this?
  - DeepWalk

Word Frequency

ctains open and the moon shining in on the
nd the cold , close moon " . And neither o
the night with the moon shining so bright
in the light of the moon . It all boils do
ly under a crescent moon , thrilled by ice
the seasons of the moon ? Home , alone ,
dazzling snow , the moon has risen full an
d the temple of the moon , driving out of

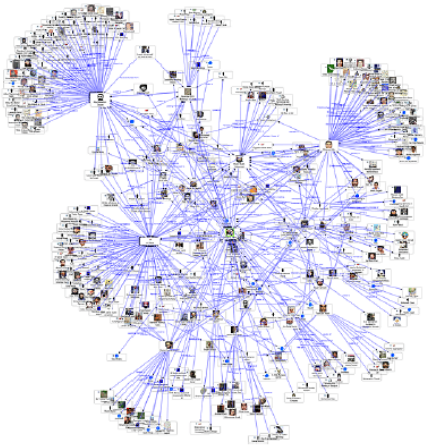
Natural language corpus



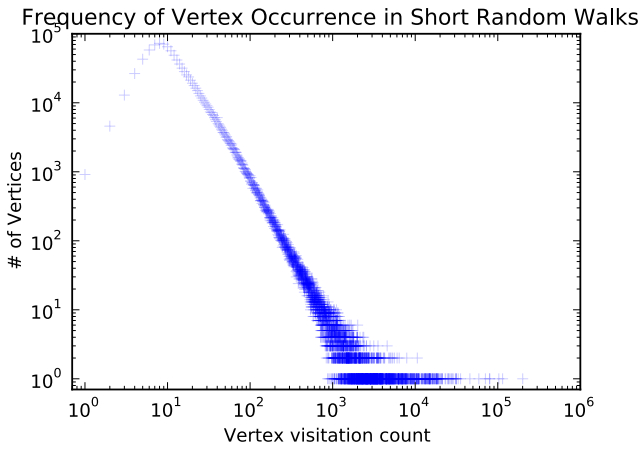
Frequency of words occurrence in a natural language corpus (e.g., Wikipedia) follows a power-law distribution

From Words to Nodes

$$
\begin{array}{cccccc}
v_{71} \rightarrow & v_{24} \rightarrow & v_5 \rightarrow & v_1 \rightarrow & v_{17} \rightarrow & v_{80} \rightarrow \\
v_{92} \rightarrow & v_2 \rightarrow & v_3 \rightarrow & v_1 \rightarrow & v_{12} \rightarrow & v_{73} \rightarrow \\
v_{37} \rightarrow & v_{34} \rightarrow & v_9 \rightarrow & v_1 \rightarrow & v_{10} \rightarrow & v_{94} \rightarrow \\
v_{73} \rightarrow & v_{64} \rightarrow & v_5 \rightarrow & v_1 \rightarrow & v_{12} \rightarrow & v_1 \rightarrow \\
v_{75} \rightarrow & v_{14} \rightarrow & v_6 \rightarrow & v_1 \rightarrow & v_{13} \rightarrow & v_{61} \rightarrow
\end{array}
$$

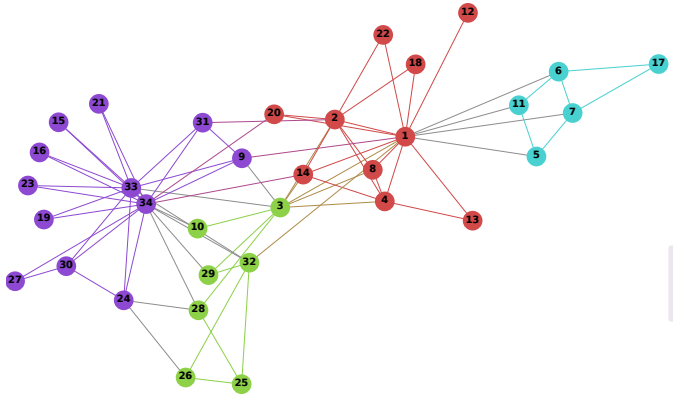


- Short truncated random walks over the network correspond to sentences in a textual corpus
- Random walk distance is known to be good node similarity feature



[Perozzi et al. '14]

DeepWalk - Overview



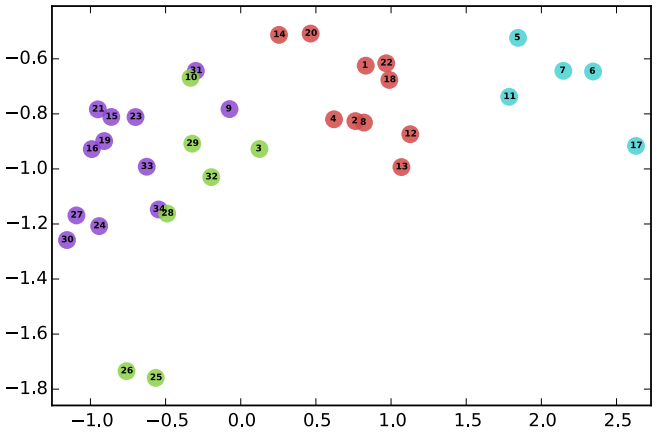
(1) Input graph

(2) Random walks

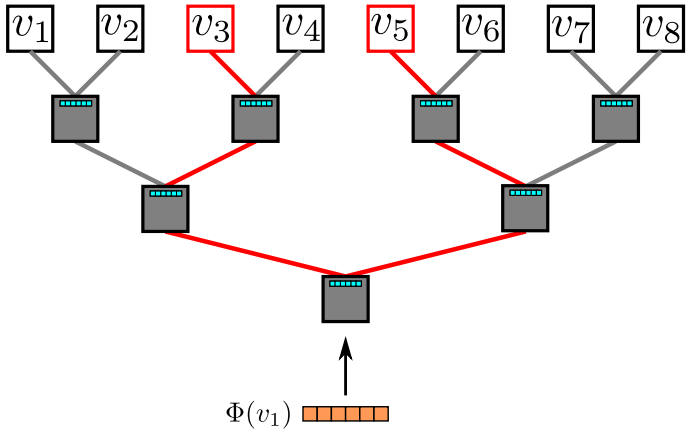
$$\mathcal{W}_{v_4} = 4$$

$$u_k \begin{bmatrix} 3 \\ 1 \\ 5 \\ 1 \\ \vdots \end{bmatrix} v_j \rightarrow \Phi \quad (d \times j)$$

(3) Representation mapping

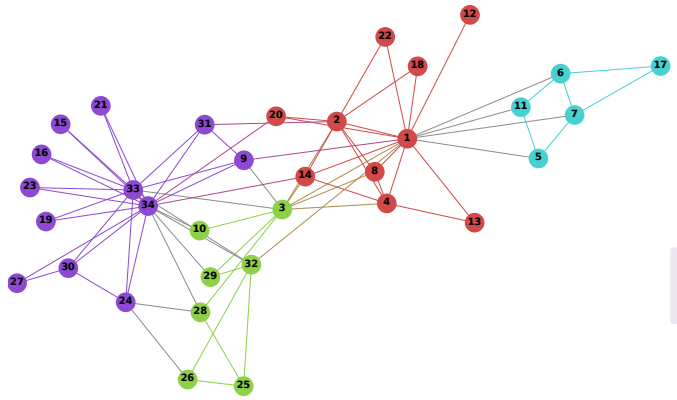


(5) Output: Representation



(4) Hierarchical Softmax

DeepWalk - Overview



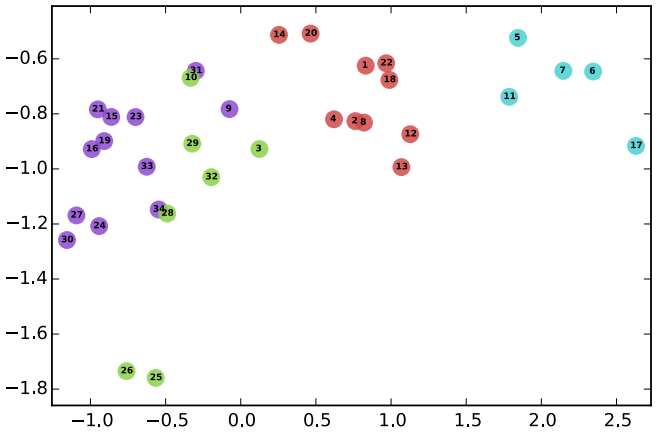
(1) Input graph

(2) Random walks

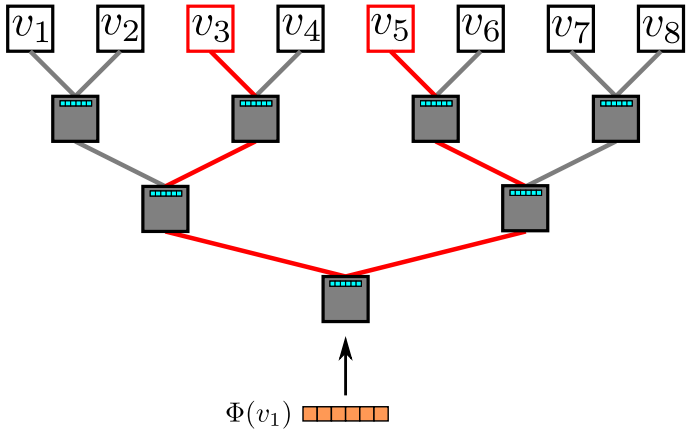
$$\mathcal{W}_{v_4} = \begin{matrix} 4 \\ \begin{bmatrix} 3 \\ 1 \\ 5 \end{bmatrix} \\ 1 \\ \vdots \end{matrix}$$

u_k v_j → Φ (d × j)

(3) Representation mapping



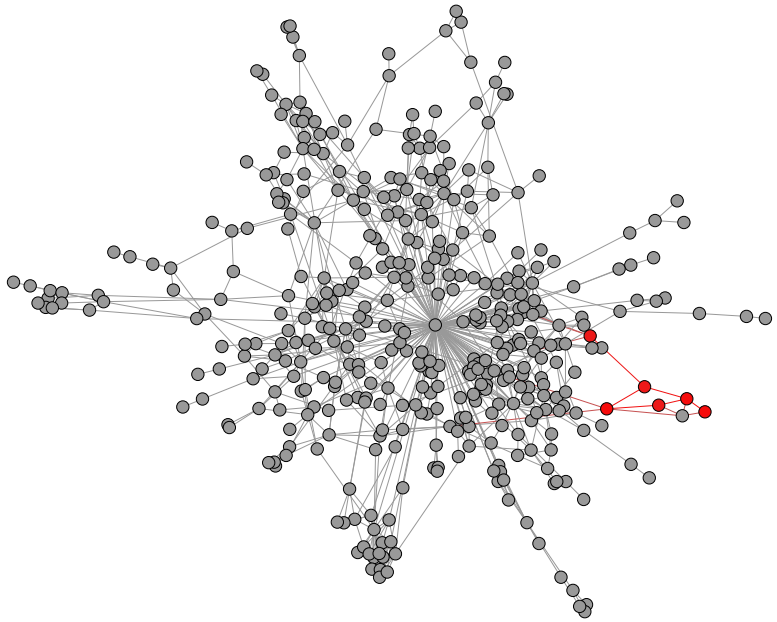
(5) Output: Representation



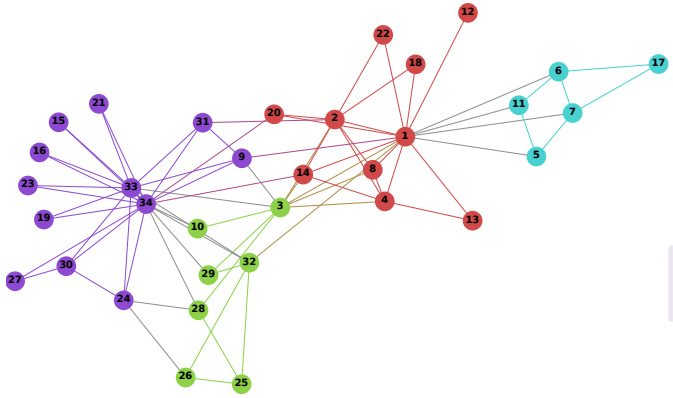
(4) Hierarchical Softmax

Random Walk Generation

- We generate γ random walks for each node in the graph
- Each short random walk has length t
- Pick the next node uniformly at random from current node's neighbors
  - Random walks can have restarts (e.g., teleportation step) but no improvement was observed in the experiments



DeepWalk - Overview



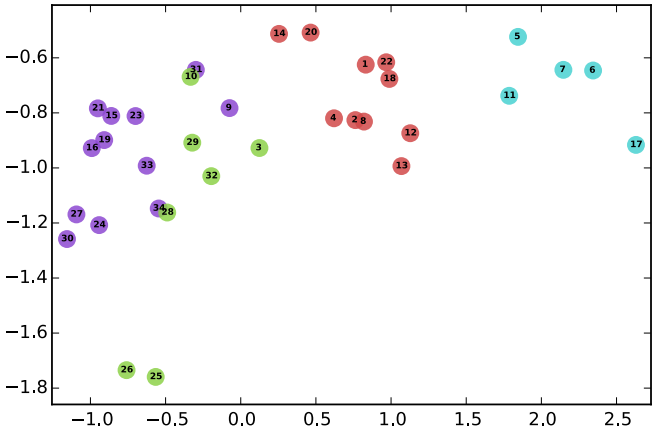
(1) Input graph

(2) Random walks

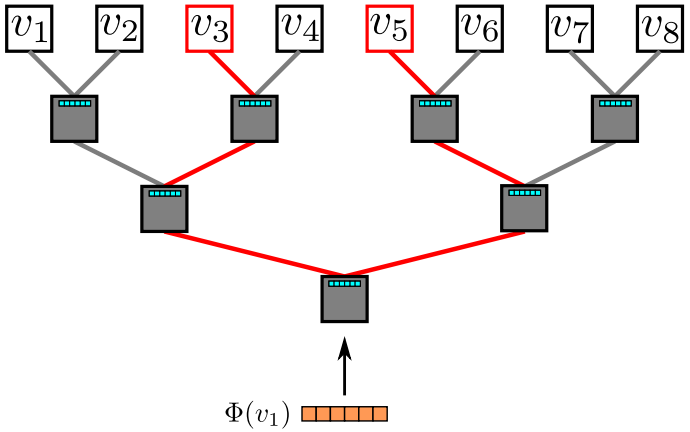
$\mathcal{W}_{v_4} = 4$

$u_k \begin{bmatrix} 3 \\ 1 \\ 5 \\ 1 \\ \vdots \end{bmatrix} v_j \rightarrow \Phi$ (d × j)

(3) Representation mapping



(5) Output: Representation



(4) Hierarchical Softmax

Representation Mapping

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

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

$$\mathcal{W}_{v_4} = \begin{matrix} 4 \\ u_k \\ \\ \\ \\ \\ \end{matrix} \begin{bmatrix} 3 \\ 1 \\ 5 \\ 1 \\ \vdots \end{bmatrix} v_j \longrightarrow \begin{matrix} d \\ \\ \\ \\ \\ \Phi \end{matrix} \; j$$

We slide the window w over the random walk Wv_4 mapping the central node v_1 to its representation $\Phi(v_1)$

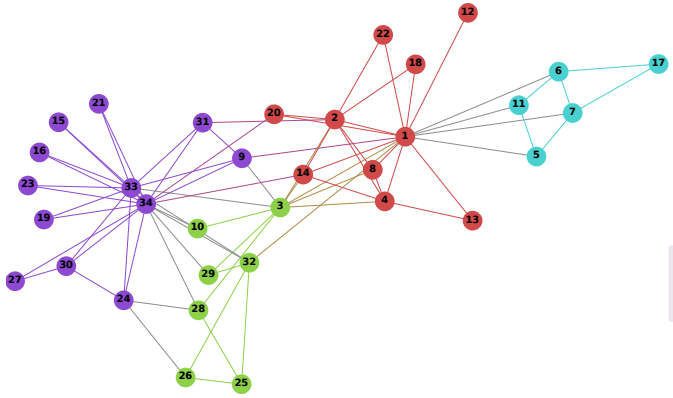
- Map the node under focus (v_1) to its representation $\Phi(v_1)$
- Define a window of size w
- If w=1 (window size) and $v = v_1$

Maximize: $\Pr(v_3|\Phi(v_1))$

$\Pr(v_5|\Phi(v_1))$

SkipGram model: predicts surrounding nodes based on v_1

DeepWalk - Overview



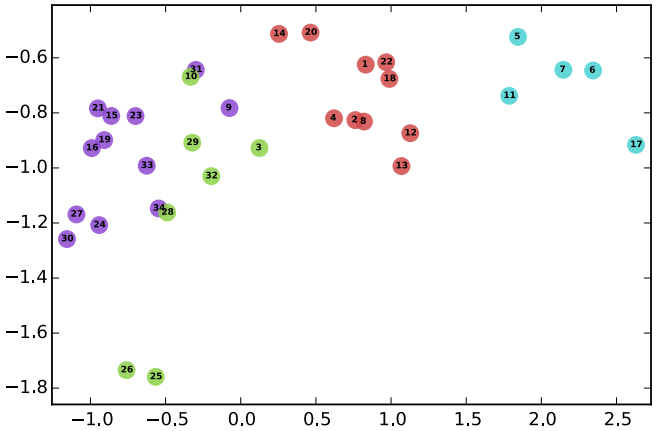
(1) Input graph

(2) Random walks

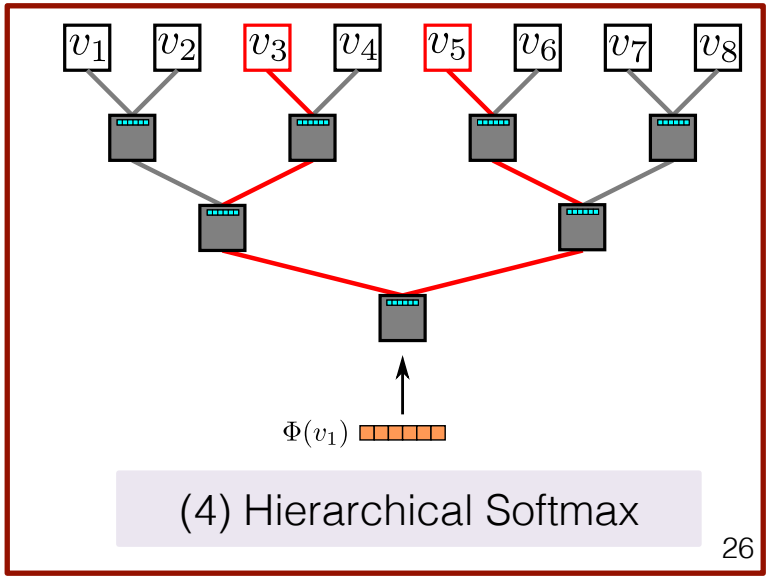
$$\mathcal{W}_{v_4} = 4$$

$$u_k \begin{bmatrix} 3 \\ 1 \\ 5 \\ 1 \\ \vdots \end{bmatrix} v_j \rightarrow \Phi \quad (d \times j)$$

(3) Representation mapping



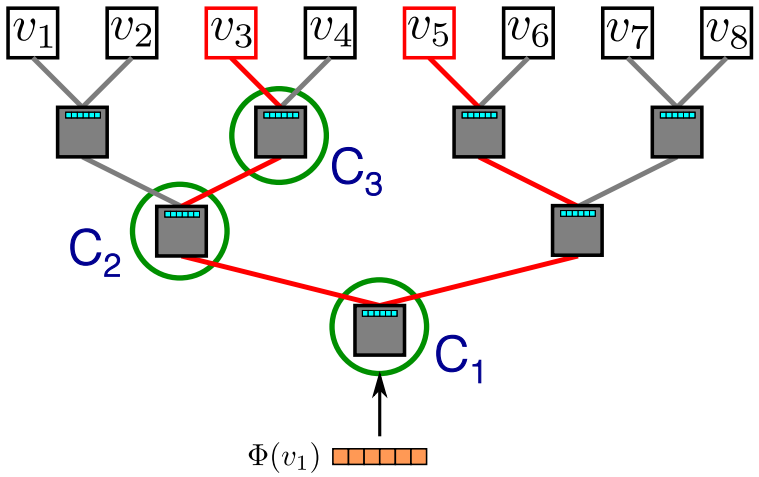
(5) Output: Representation



(4) Hierarchical Softmax

Hierarchical Softmax

- Given the representation $\Phi(v_1)$ of v_1, we would like to maximize the probability of its neighbors v_3 and v_5 in the random walk
  - $\Pr(v_3|\Phi(v_1))$ and $\Pr(v_5|\Phi(v_1))$
  - We can learn such posterior distribution using logistic regression
  - Huge number of labels (equal to |V|)



Each of {C_1, C_2, C_3} is a logistic binary classifier

- Consider the nodes of the graph as leaves of a balanced binary tree
- Maximizing $\Pr(v_3|\Phi(v_1))$ is equivalent to maximizing the probability of the path from the root to the node

$$\Pr(v_3|\Phi(v_1))$$

product

$$\Pr(\text{left} \mid \Phi(v_1); C_1)$$
$$\Pr(\text{right} \mid \Phi(v_1); C_2)$$
$$\Pr(\text{left} \mid \Phi(v_1); C_3)$$

Evaluate log(|V|) nodes to obtain the probability of a node

Experiments – Datasets and Baselines

- Datasets
  - BlogCatalog: social relationships provided by blogger authors
  - Flickr: social network photo sharing website
  - Youtube

Name	BlogCatalog	Flickr	YouTube
$\|V\|$	10,312	80,513	1,138,499
$\|E\|$	333,983	5,899,882	2,990,443
$\|\mathcal{Y}\|$	39	195	47
Labels	Interests	Groups	Groups

- Baseline methods
  - Spectral Clustering: eigenvectors that correspond to the **d**-smallest eigenvalues of the normalized Laplacian
  - Modularity: top-**d** eigenvectors of the modularity matrix
  - EdgeCluster: k-means on a representation extracted by an edge partition scheme
  - weighted-vote Relational Neighbor: relational classifier

[Perozzi et al. '14]

Source code: https://github.com/phanein/deepwalk

Multi-label Classification - BlogCatalog

	% 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

DeepWalk has good performance especially when labels are sparse (a small fraction of the graph is labeled)

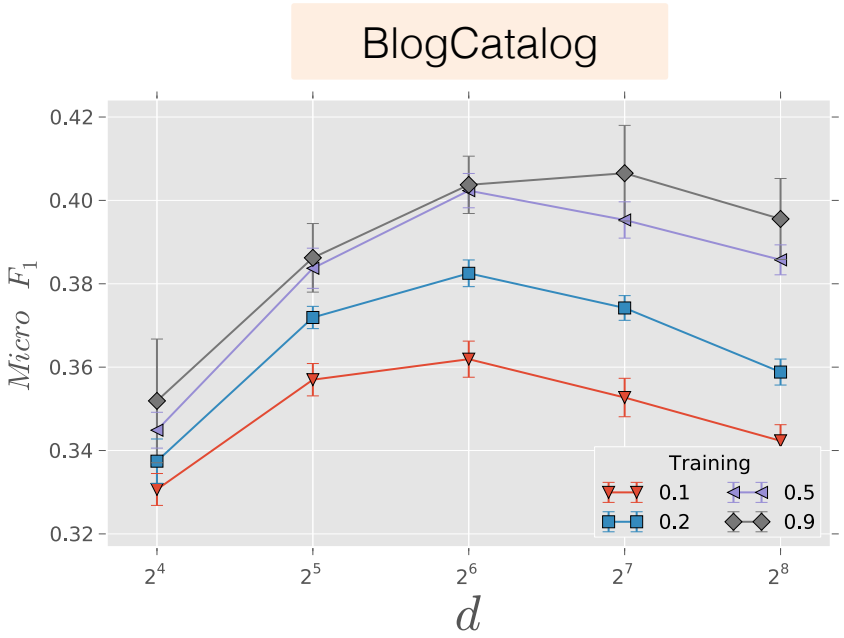
[Perozzi et al. '14]

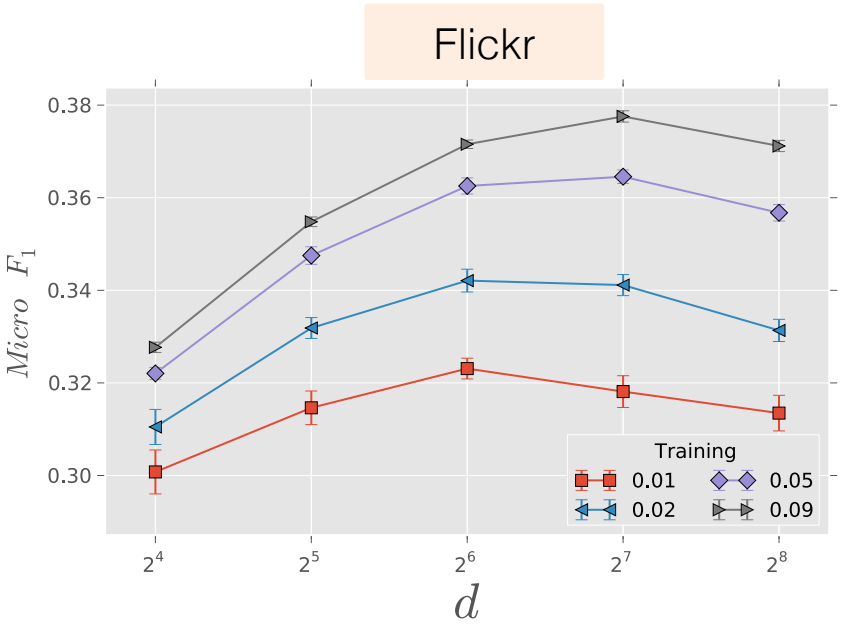
Multi-label Classification - YouTube

	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	DeepWalk	**37.95**	**39.28**	**40.08**	**40.78**	**41.32**	**41.72**	**42.12**	**42.48**	**42.78**	**43.05**
	SpectralClustering	—	—	—	—	—	—	—	—	—	—
	EdgeCluster	23.90	31.68	35.53	36.76	37.81	38.63	38.94	39.46	39.92	40.07
	Modularity	—	—	—	—	—	—	—	—	—	—
	wvRN	26.79	29.18	33.1	32.88	35.76	37.38	38.21	37.75	38.68	39.42
	Majority	24.90	24.84	25.25	25.23	25.22	25.33	25.31	25.34	25.38	25.38
Macro-F1(%)	DeepWalk	**29.22**	**31.83**	**33.06**	**33.90**	**34.35**	**34.66**	**34.96**	**35.22**	**35.42**	**35.67**
	SpectralClustering	—	—	—	—	—	—	—	—	—	—
	EdgeCluster	19.48	25.01	28.15	29.17	29.82	30.65	30.75	31.23	31.45	31.54
	Modularity	—	—	—	—	—	—	—	—	—	—
	wvRN	13.15	15.78	19.66	20.9	23.31	25.43	27.08	26.48	28.33	28.89
	Majority	6.12	5.86	6.21	6.1	6.07	6.19	6.17	6.16	6.18	6.19

- Spectral methods do not scale to large graphs
- DeepWalk performs much better than the baselines that scale to large graphs (|V|=1,1M, |E|=3M)

Stability vs. Dimensions d





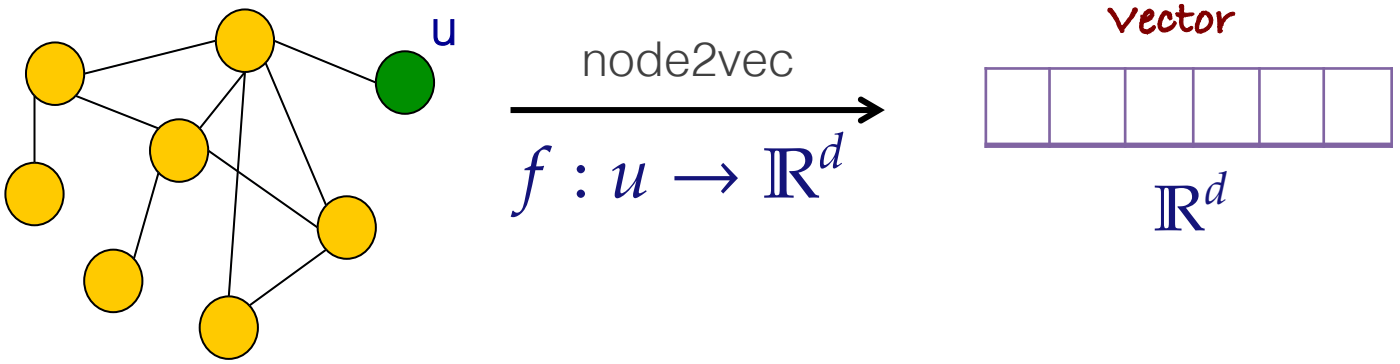
- Vary the number of latent dimensions d and the amount of training data available

node2vec algorithm

KDD ‘16

Unsupervised Feature Learning

- Idea: learn node embeddings such that nearby nodes are close together
  - Similar to DeepWalk



- Given a node u, how do we define nearby nodes?
  - $N_s(u)$: neighborhood of u obtained by some strategy S
  - In the case of DeepWalk, this was a random walk

[Grover and Leskovec '16]

node2vec – The Framework

- Goal: find embedding f(u) that predicts nearby nodes $N_s(u)$

$$\max_f \sum_{u \in V} \log \Pr(N_S(u)|f(u))$$

Need to make the opt problem tractable

1. Conditional independence assumption

$$\Pr(N_S(u)|f(u)) = \prod_{n_i \in N_S(u)} \Pr(n_i|f(u))$$

Assume that the likelihood of observing a neighborhood node is independent of observing any other neighborhood node given the feature representation of the source

2. Symmetry in feature space

$$\Pr(n_i|f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} (f(v) \cdot f(u))}$$

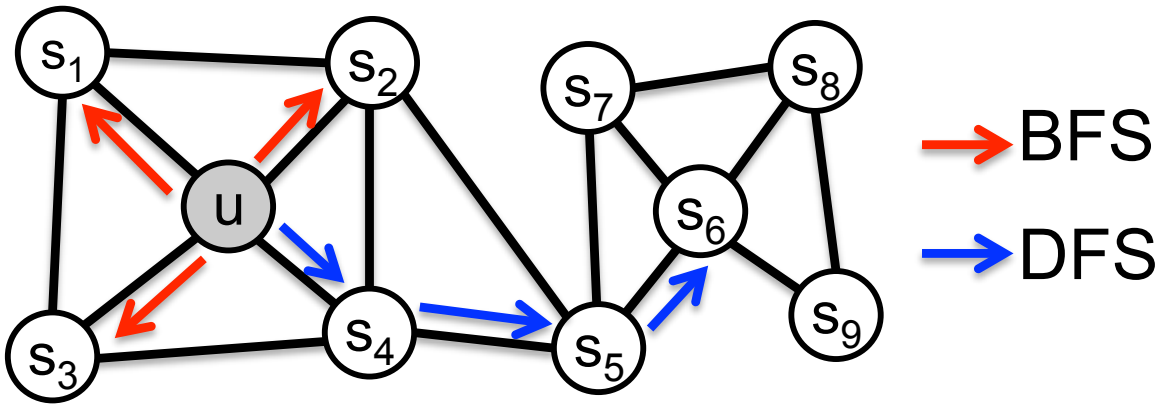
A source node and a neighborhood node have a symmetric effect over each other in feature space

Model the conditional likelihood of every source-neighborhood node pair as a softmax unit parameterized by a dot product of their features

Estimate f(u) using SGD

How to Estimate $N_s(u)$

Sampling neighborhoods $N_s(u)$ of a source node u as a form of local search

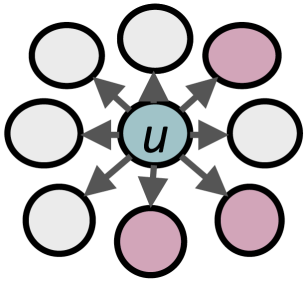


$$N_{BFS}(u) = \{s_1, s_2, s_3\}$$ Local microscopic view

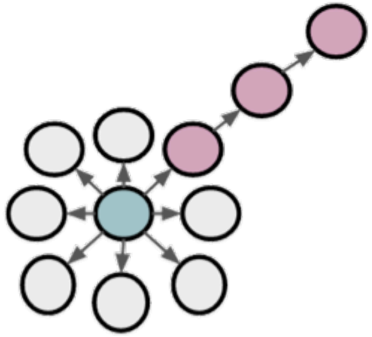
$$N_{DFS}(u) = \{s_4, s_5, s_6\}$$ Global macroscopic view

[Grover and Leskovec '16]

BFS vs. DFS



BFS: micro-view of neighborhood



DFS: macro-view of neighborhood

Those searching strategies can capture two properties of social networks that lead to different concepts of similarity

Homophily and structural equivalence

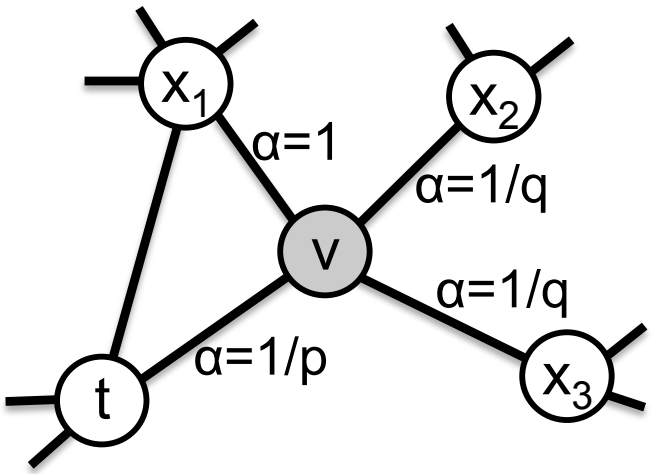
(communities) (structural roles (e.g., hubs))

Interpolating Between BFS and DFS

- Sampling strategy which allows to interpolate between BFS and DFS
- Biased random walk S that, given a node u generates neighborhood $N_s(u)$
- Two parameters:
  - Return parameter p
    - Return back to the previous node
  - In-out parameter q
    - Moving outwards (DFS) or inwards (BFS)

Biased Random Walk

Biased 2nd order random walk



- Suppose that the walk just traversed (t, v) and now resides in node v
- Transition probabilities: $\pi_{vx} = \alpha_{pq}(t, x) * w_{vx}$

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p}, & \text{if } dist(t,x) = 0 \\ 1, & \text{if } dist(t,x) = 1 \\ \frac{1}{q}, & \text{if } dist(t,x) = 2 \end{cases}$$

- BFS-like: low value of p
- DFS-like: low value of q

The node2vec Algorithm

Input: dimensions d, walks per node r, walk length ℓ, parameters p, q

Output: node embeddings

1. Compute random walk probabilities
2. Simulate r random walks of length ℓ starting from each node u
3. Optimize the node2vec objective function using SGD

Source code: snap.stanford.edu/node2vec/

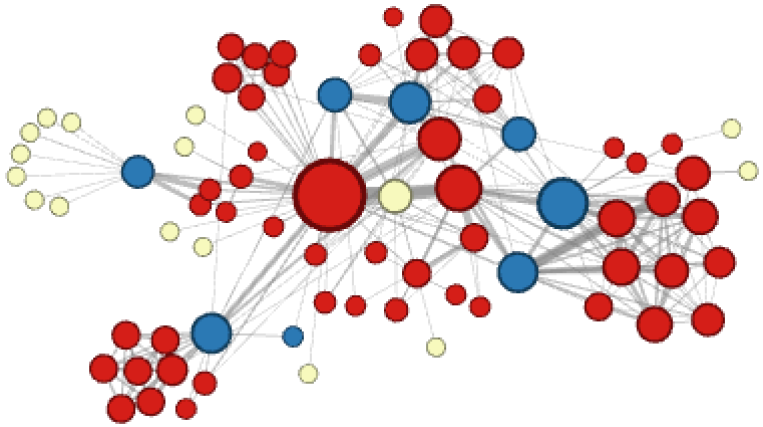
[Grover and Leskovec '16]

Experiments: Parameters p, q

Homophily

p=1, q=0.5

Structural equivalence
(structural roles)

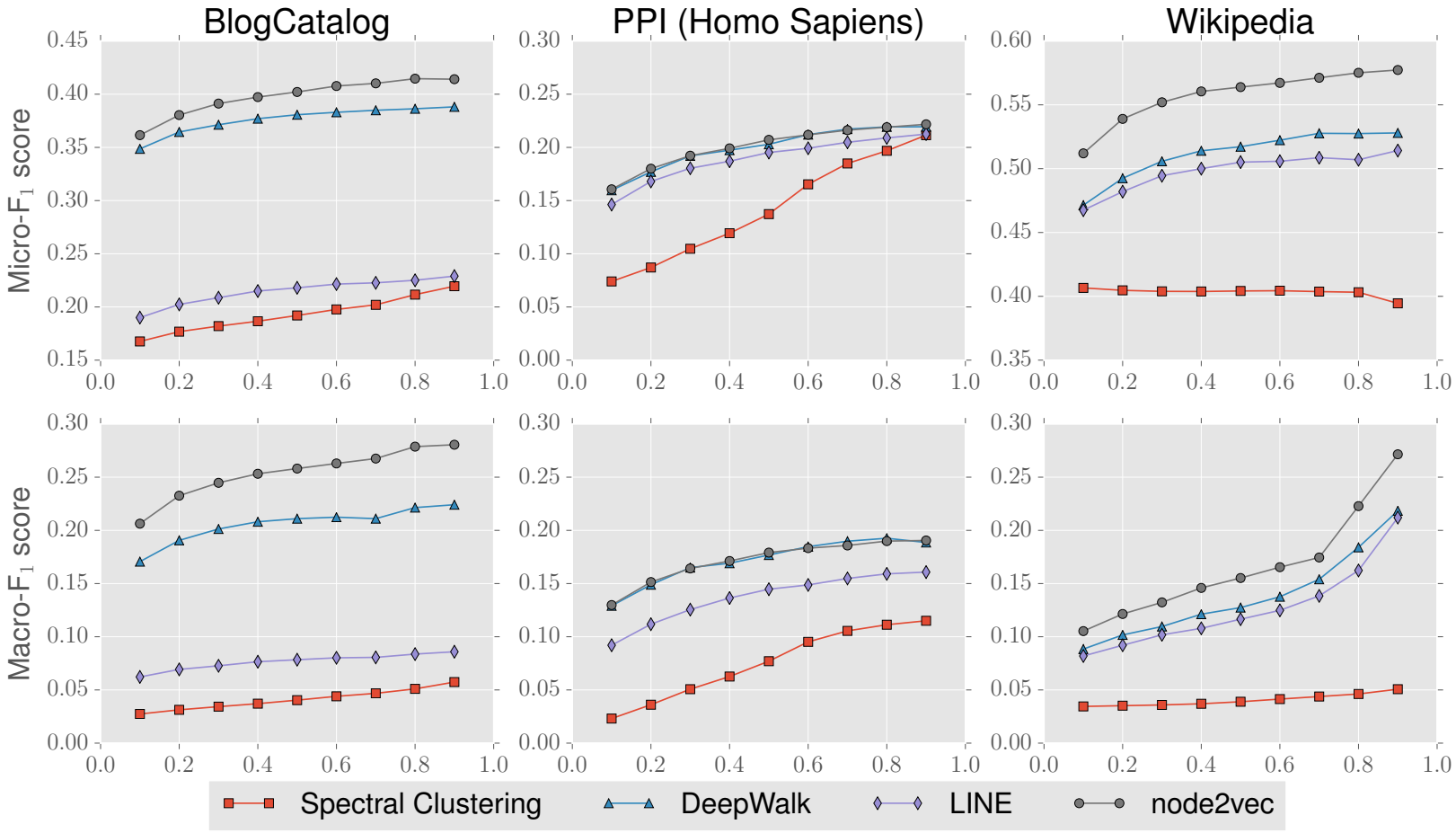


p=1, q=2

Les Misérables graph

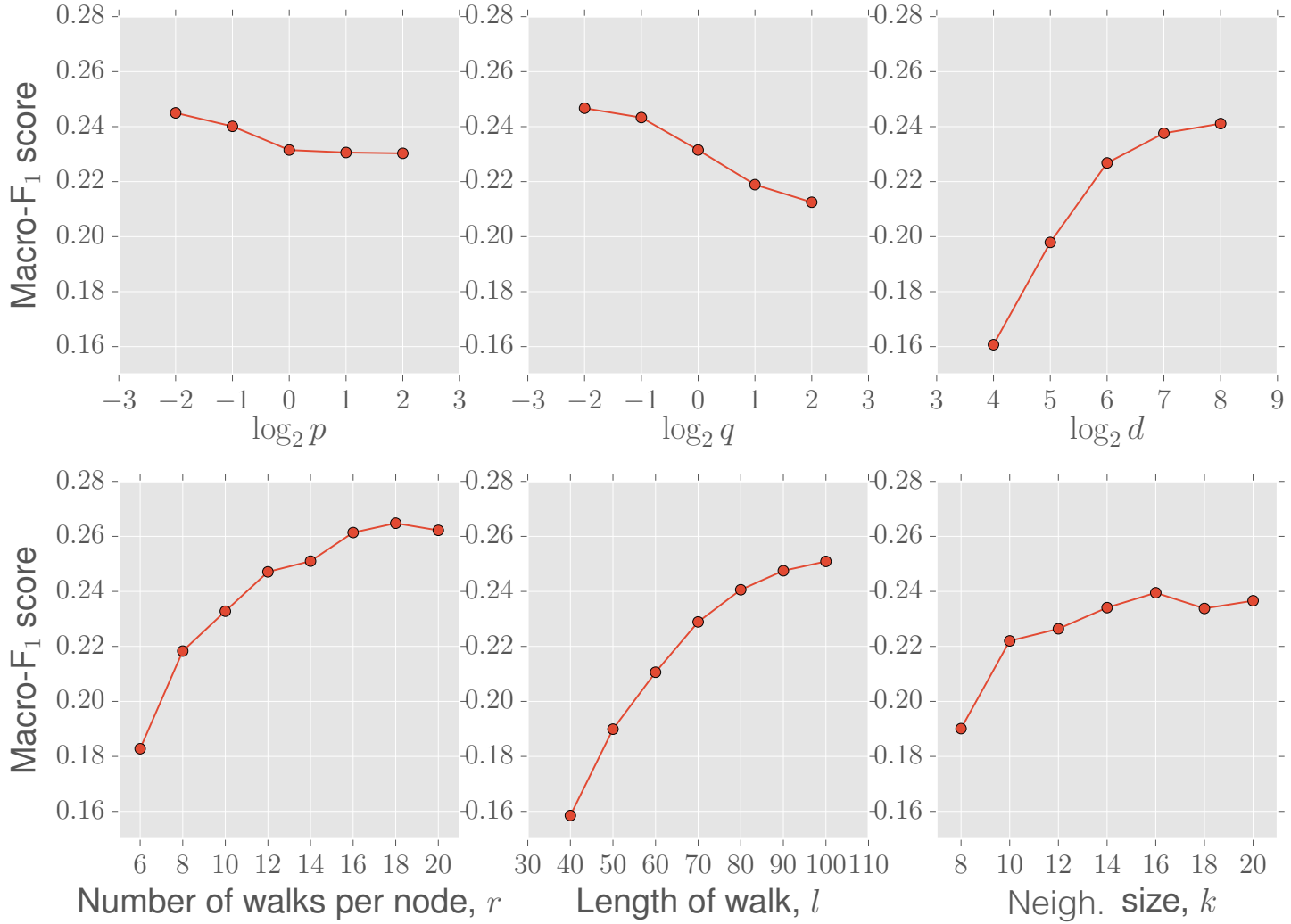
- node2vec with d=16
- Colors correspond to clusters based on k-means

Experiments – Multi-label Classification



Accuracy (y-axis) vs. fraction of labeled data (x-axis)

Performance vs. Parameters



LINE algorithm

WWW ‘15

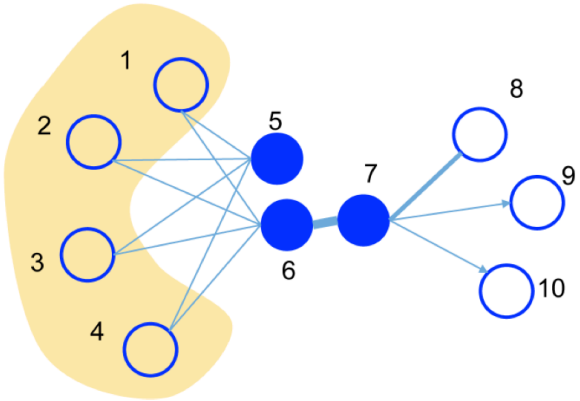
Problem Statement

- How to embed large networks into low-dimensional spaces?
- Requirements
  - Globality/Locality: It is desirable to preserve both local and global network structure when seeking for node representations
  - Scalability: When considering network with millions of nodes and billions of edges: traditional methods (nonlinear dimensionality reduction) suffer from lack of scalability

[Tang et al. '15]

Intuition – Local and Global Structures

- Local structure: observed edges in the network
  - First order proximity
  - Most traditional embedding methods (e.g., Isomap) capture first order proximity
- Global structure: nodes with shared neighbors are likely to be similar (homophily)



- Nodes 6 and 7: first-order proximity
  - Should be represented closely in the embedded space
- Nodes 5 and 6: second-order proximity
  - Same for those nodes

LINE algorithm: Form an objective function that optimizes both local and global network structure

[Tang et al. '15]

LINE with First-order Proximity (1/2)

Model the probability of an edge (i, j) between v_i and v_j as

Embeddings space

$$p_1(v_i, v_j) = \frac{1}{1 + \exp(-\vec{u}_i^T \cdot \vec{u}_j)}$$

$\vec{u}_i \in \mathbb{R}^d$ Low dimensional vector representation of node v_i

Joint probability between v_i and v_j

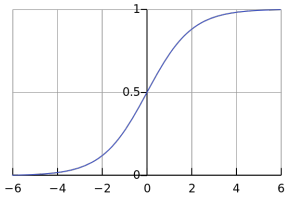
Original (graph) space

$$\hat{p}_1(i, j) = \frac{w_{ij}}{\sum_{(i,j) \in E} w_{ij}}$$

edge

Empirical distribution over the space V×V

Logistic function



Find vectors $\vec{u}_i \in \mathbb{R}^d$ to make those distributions to be as close as possible

LINE with First-order Proximity (2/2)

How to preserve first-order proximity?

$$O_1 = d(\hat{p}_1(\cdot,\cdot), p_1(\cdot,\cdot))$$

Minimize the distance between two distributions

$$KL(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

KL-divergence

$$O_1 = KL(\hat{p}_1(\cdot,\cdot), p_1(\cdot,\cdot))$$

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

By finding those $\{\vec{u}_i\}_{i=1..|V|}$ that minimize O_1, we can represent every node in the d-dimensional space

LINE with Second-order Proximity (1/3)

- It assumes that nodes sharing many connections to other nodes are similar to each other
- Each node plays two roles:
  - The node itself
  - A specific "context" of other nodes

- For each node v_i, we model the conditional distribution $p_2(\bullet|v_i)$ over all the "contexts" (all the nodes in the network)
- Assumption of second-order proximity: Nodes with similar distributions $p_2(\bullet|v_i)$ over the "contexts" are similar

LINE with Second-order Proximity (2/3)

For directed edge (i, j), model the probability of context v_j generated by node v_i (i.e., probability of an edge from v_i to v_j)

Embeddings space

$$p_2(v_j|v_i) = \frac{\exp(\vec{u}_j'^T \cdot \vec{u}_i)}{\sum_{k=1}^{|V|} \exp(\vec{u}_k'^T \cdot \vec{u}_i)}$$

$\vec{u}_i \in \mathbb{R}^d$ Low dimensional vector representation of node v_i

Conditional distribution $p_2(\bullet|v_i)$ over the contexts

Original (graph) space

$$\hat{p}_2(v_j|v_i) = \frac{w_{ij}}{d_i}$$

Out-degree of node i $\quad d_i = \sum_{k \in N(i)} w_{ik}$

Empirical distribution over the space V×V

Make those distributions to be as close as possible

LINE with Second-order Proximity (3/3)

To preserve second-order proximity, minimize the distance between true and empirical distributions

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

Minimize the distance between two distributions

- λ_i : represents the prestige of node i in the graph
- Set $\lambda_i = d_i$

KL-divergence

$$O_2 = -\sum_{(i,j) \in E} w_{ij} \log p_2(v_j|v_i)$$

By finding those $\{\vec{u}_i\}_{i=1..|V|}$ and $\{\vec{u}'_i\}_{i=1..|V|}$ that minimize O_2, we can represent every node i with d-dimensional space $\vec{u}_i$

Combining Both Models

- Goal: Embed the networks by preserving both the first-order and second-order proximity

1. Train the LINE model for first-order proximity
2. Train the LINE model for second-order proximity

Train separately

- Then, concatenate the embeddings trained by the two methods for each node

Experiments - Datasets

Word co-occurrence network

	Language Network	Social Network	
Name	WIKIPEDIA	FLICKR	YOUTUBE
Type	undirected,weighted	undirected,binary	undirected,binary
|V|	1,985,098	1,715,256	1,138,499
|E|	1,000,924,086	22,613,981	2,990,443
Avg. degree	504.22	26.37	5.25
#Labels	7	5	47
#train	70,000	75,958	31,703

Citation Network	
DBLP(AUTHORCITATION)	DBLP(PAPERCITATION)
dircted,weighted	directed,binary
524,061	781,109
20,580,238	4,191,677
78.54	10.73
7	7
20,684	10,398

[Tang et al. '15]

Experiments – Word Analogy

- Language network: word analogy
  - Find solution to ("China", "Beijing" → "France, "?")
  - Given word embeddings, find word d^* whose embedding u_d is closest to vector $u_{Beijing} - u_{China} + u_{France}$

"Paris"

Proximity in terms of cosine similarity

Algorithm	Semantic (%)	Syntactic (%)	Overall (%)	Running time
GF	61.38	44.08	51.93	2.96h
DeepWalk	50.79	37.70	43.65	16.64h
SkipGram	69.14	57.94	63.02	2.82h
LINE-SGD(1st)	9.72	7.48	8.50	3.83h
LINE-SGD(2nd)	20.42	9.56	14.49	3.94h
LINE(1st)	58.08	49.42	53.35	2.44h
LINE(2nd)	**73.79**	**59.72**	**66.10**	2.55h

Line (2nd) outperforms other embedding methods in the word analogy task

Experiments – Document Classification

Metric	Algorithm	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	GF	79.63	80.51	80.94	81.18	81.38	81.54	81.63	81.71	81.78
	DeepWalk	78.89	79.92	80.41	80.69	80.92	81.08	81.21	81.35	81.42
	SkipGram	79.84	80.82	81.28	81.57	81.71	81.87	81.98	82.05	82.09
	LINE-SGD(1st)	76.03	77.05	77.57	77.85	78.08	78.25	78.39	78.44	78.49
	LINE-SGD(2nd)	74.68	76.53	77.54	78.18	78.63	78.96	79.19	79.40	79.57
	LINE(1st)	79.67	80.55	80.94	81.24	81.40	81.52	81.61	81.69	81.67
	LINE(2nd)	79.93	80.90	81.31	81.63	81.80	81.91	82.00	82.11	82.17
	LINE(1st+2nd)	**81.04****	**82.08****	**82.58****	**82.93****	**83.16****	**83.37****	**83.52****	**83.63****	**83.74****
Macro-F1	GF	79.49	80.39	80.82	81.08	81.26	81.40	81.52	81.61	81.68
	DeepWalk	78.78	79.78	80.30	80.56	80.82	80.97	81.11	81.24	81.32
	SkipGram	79.74	80.71	81.15	81.46	81.63	81.78	81.88	81.98	82.01
	LINE-SGD(1st)	75.85	76.90	77.40	77.71	77.94	78.12	78.24	78.29	78.36
	LINE-SGD(2nd)	74.70	76.45	77.43	78.09	78.53	78.83	79.08	79.29	79.46
	LINE(1st)	79.54	80.44	80.82	81.13	81.29	81.43	81.51	81.60	81.59
	LINE(2nd)	79.82	80.81	81.22	81.52	81.71	81.82	81.92	82.00	82.07
	LINE(1st+2nd)	**80.94****	**81.99****	**82.49****	**82.83****	**83.07****	**83.29****	**83.42****	**83.55****	**83.66****

- Classification of Wikipedia articles
  - Choose articles from 7 categories
- How to obtain the document vectors for classification?
  - Average of the corresponding word vector representations

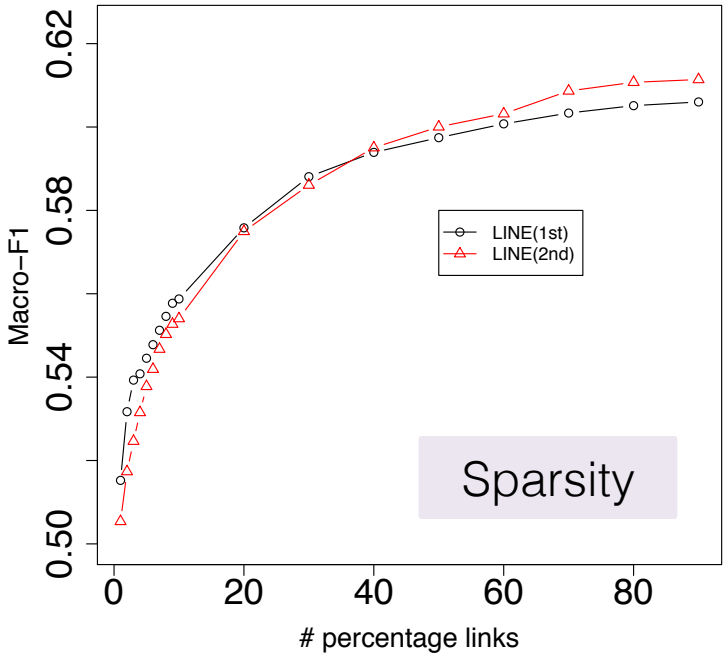
Experiments – Classification on Flickr

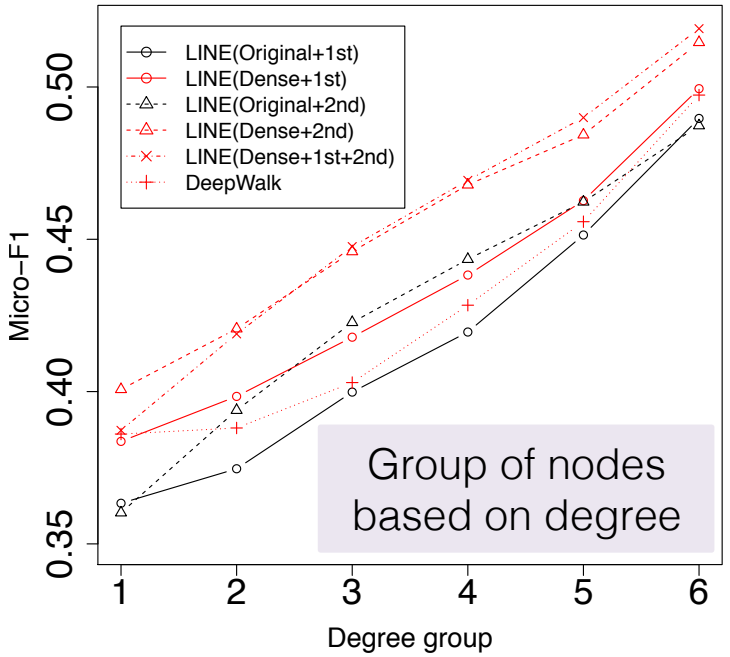
Metric	Algorithm	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	GF	53.23	53.68	53.98	54.14	54.32	54.38	54.43	54.50	54.48
	DeepWalk	60.38	60.77	60.90	61.05	61.13	61.18	61.19	61.29	61.22
	DeepWalk(256dim)	60.41	61.09	61.35	61.52	61.69	61.76	61.80	61.91	61.83
	LINE(1st)	63.27	63.69	63.82	63.92	63.96	64.03	64.06	64.17	64.10
	LINE(2nd)	62.83	63.24	63.34	63.44	63.55	63.55	63.59	63.66	63.69
	LINE(1st+2nd)	**63.20****	**63.97****	**64.25****	**64.39****	**64.53****	**64.55****	**64.61****	**64.75****	**64.74****
Macro-F1	GF	48.66	48.73	48.84	48.91	49.03	49.03	49.07	49.08	49.02
	DeepWalk	58.60	58.93	59.04	59.18	59.26	59.29	59.28	59.39	59.30
	DeepWalk(256dim)	59.00	59.59	59.80	59.94	60.09	60.17	60.18	60.27	60.18
	LINE(1st)	62.14	62.53	62.64	62.74	62.78	62.82	62.86	62.96	62.89
	LINE(2nd)	61.46	61.82	61.92	62.02	62.13	62.12	62.17	62.23	62.25
	LINE(1st+2nd)	**62.23****	**62.95****	**63.20****	**63.35****	**63.48****	**63.48****	**63.55****	**63.69****	**63.68****

- Multi-label classification on the Flickr network
  - Assign every node to one or more communities
- First-order proximity (LINE 1st) performs slightly better than LINE 2nd
  - In social networks, 1st order proximity is more important than 2nd order proximity
  - Effect of sparsity: # number of neighbors per node is small

Performance w.r.t. Network Sparsity

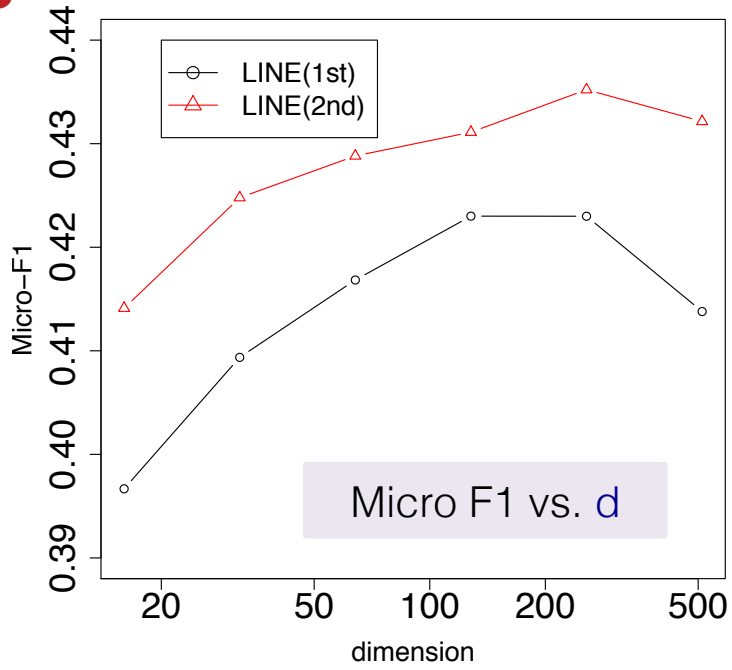
flickr

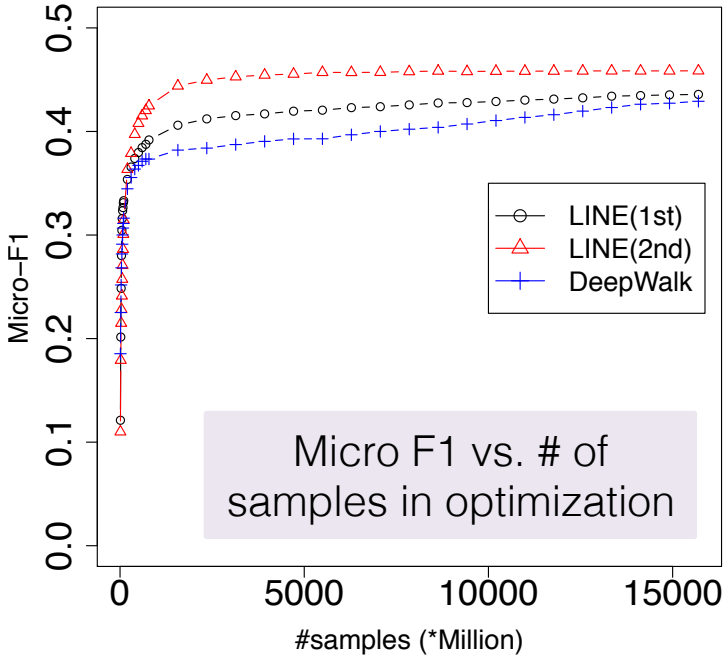




- For sparse instances of the graph LINE (1st) performs better
- The performance of different models increases when the degrees of the vertices increase

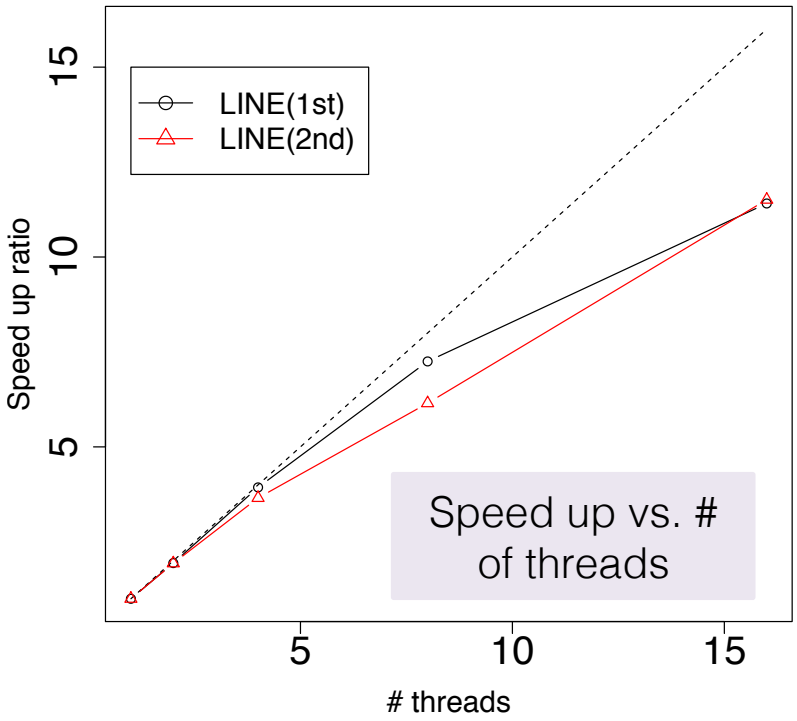
Parameter Sensitivity





- The performance drops when the dimension becomes too large
- LINE (2^{nd}) converges faster

Scalability



- The speed up is quite close to linear

Summary

- Carefully designed objective functions (O_1 and O_2) preserve first- and second-order node proximity
- The learning (optimization) phase of LINE can be slow
  - Apply edge sampling and negative sampling to reduce running time
- Experiments on various datasets and tasks support the good performance of LINE

Summary of Embedding Methods

- Very useful tool
  - Feature learning vs. feature engineering
- Many other methods have recently been proposed
  - Capture important structural information of the graph
  - E.g., community preserving network embeddings
- Graph embeddings
  - Applications in graph classification
  - Replace graph kernels

Currently very active research topic

Thank You!

DiscoverGreece.com