CSE8803/CX4803

Machine Learning in Computational Biology

Lecture 15:
Representation Learning in Graphs (Network Embeddings)

Yunan Luo

Today’s plan

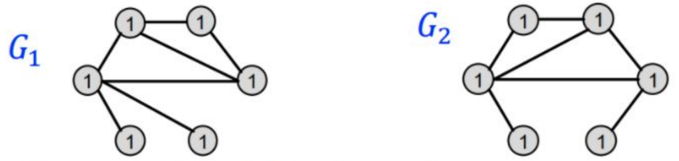
- Graph representation learning
- Student presentation

Logistics

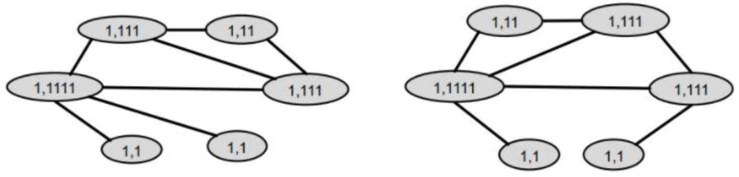
Questions posted on Ed during the weekend

Weisfeiler-Lehman kernel

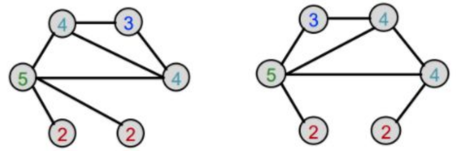
- Assign initial colors



- Aggregate neighboring colors

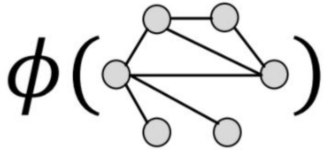


- Hash aggregated colors

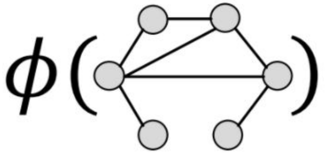


Hash table

1,1	-->	2
1,11	-->	3
1,111	-->	4
1,1111	-->	5



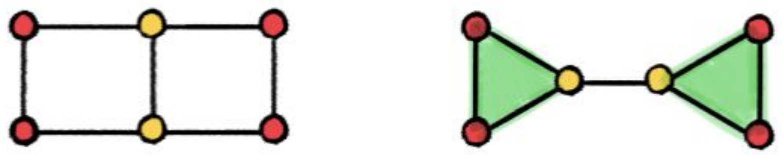
Colors
1,2,3,4,5,6,7,8,9,10,11,12,13
= [6,2,1,2,1,0,2,1,0, 0, 0, 2, 1]
Counts



1,2,3,4,5,6,7,8,9,10,11,12,13
= [6,2,1,2,1,1,1,0,1, 1, 1, 0, 1]

Weisfeiler-Lehman kernel

- If the WL color assignments are different, the two graphs are not isomorphic
- If the WL color assignments are the same, are the two graphs are isomorphic?
  - possibly, but not necessarily

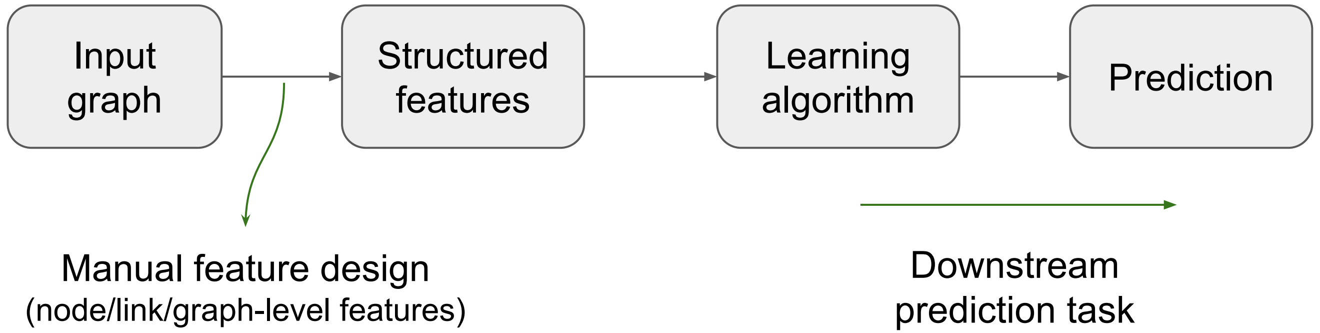


https://blog.twitter.com/engineering/en_us/topics/insights/2021/provably-expressive-graph-neural-networks

- How many iterations needed for the color refinement?
  - To test the isomorphism between two graphs, *N* (#nodes) iterations
  - Feature vectors, can choose a predefined step *h* <= *N*

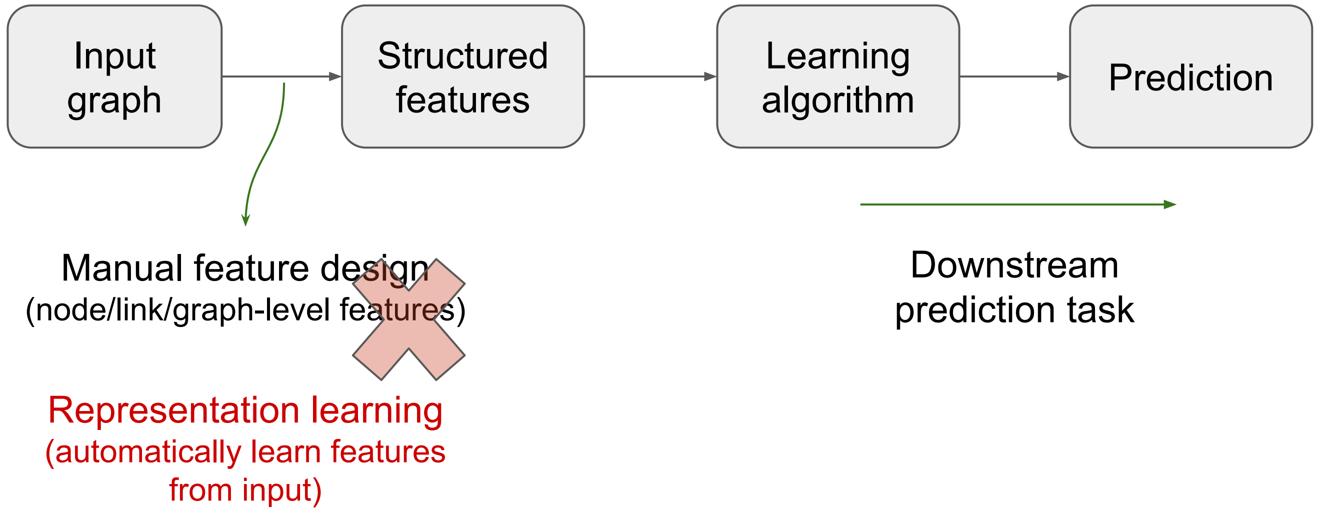
Last lecture: traditional ML for graphs

- Given an input graph, extract node, link and graph-level features, learn a model (SVM, neural network, etc.) that maps features to labels.



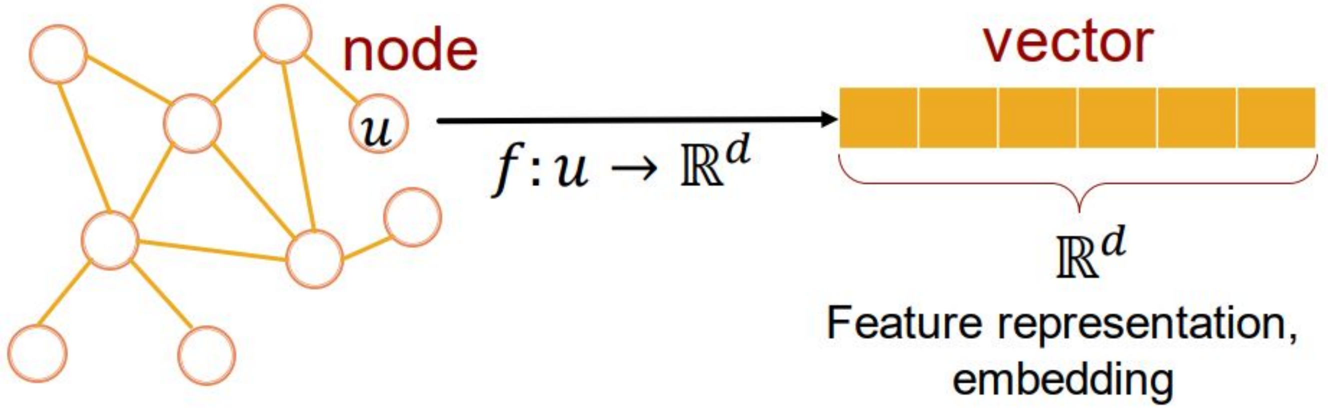
Graph representation learning

- Graph representation learning: learn a feature for each node from the graph input automatically



Graph representation learning

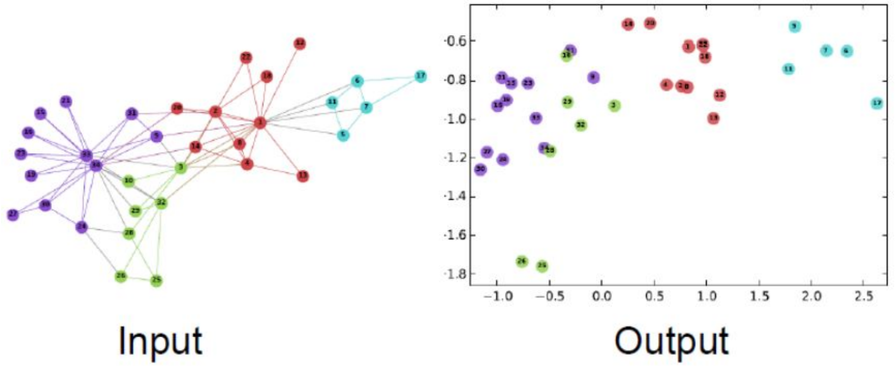
- **Goal**: Efficient task-independent feature learning for machine learning with graphs



Adapted from CS224W, Jure Leskovec

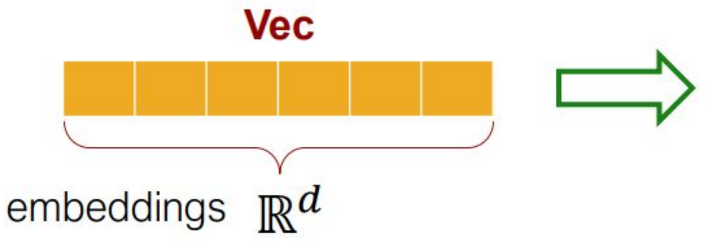
Why embedding?

- Similarity of embeddings between nodes indicates their similarity in the network.



DeepWalk: Online Learning of Social Representations. KDD 2014

- Potentially used for many downstream predictions

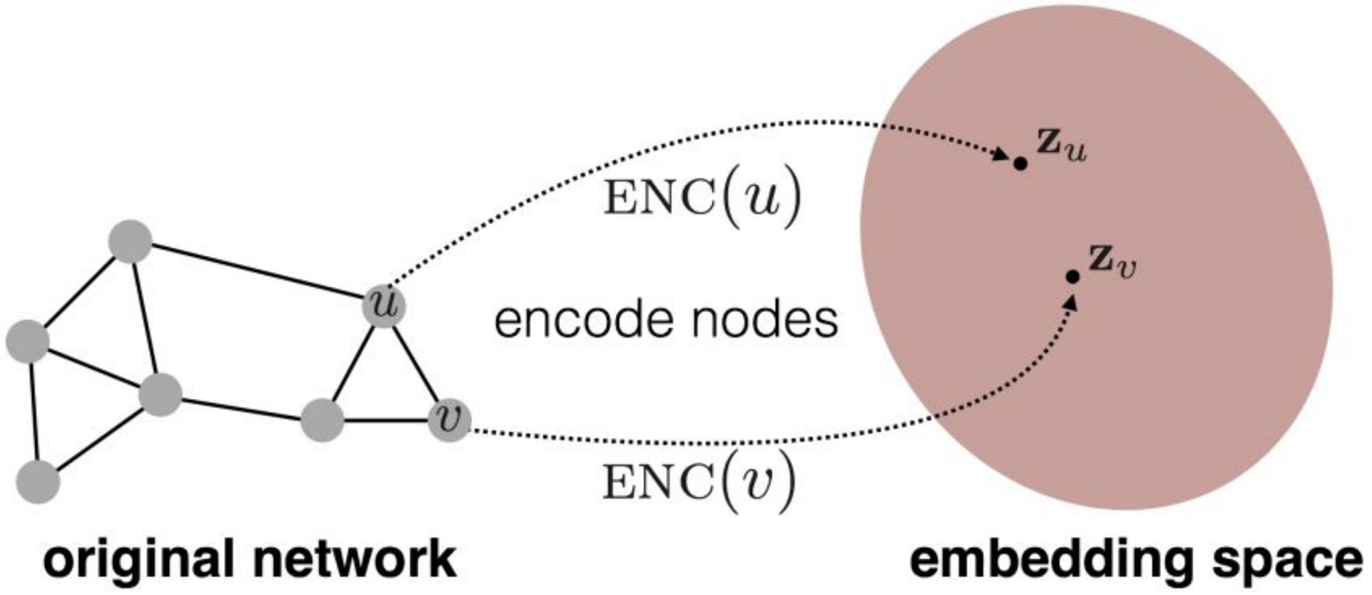


Tasks
- Node classification
- Link prediction
- Graph classification
- Anomalous node detection
- Clustering
- ....

Adapted from CS224W, Jure Leskovec

Embedding nodes

- **Goal**: encode nodes so that similarity in the embedding space (e.g., dot product) approximates similarity in the graph



Adapted from CS224W, Jure Leskovec

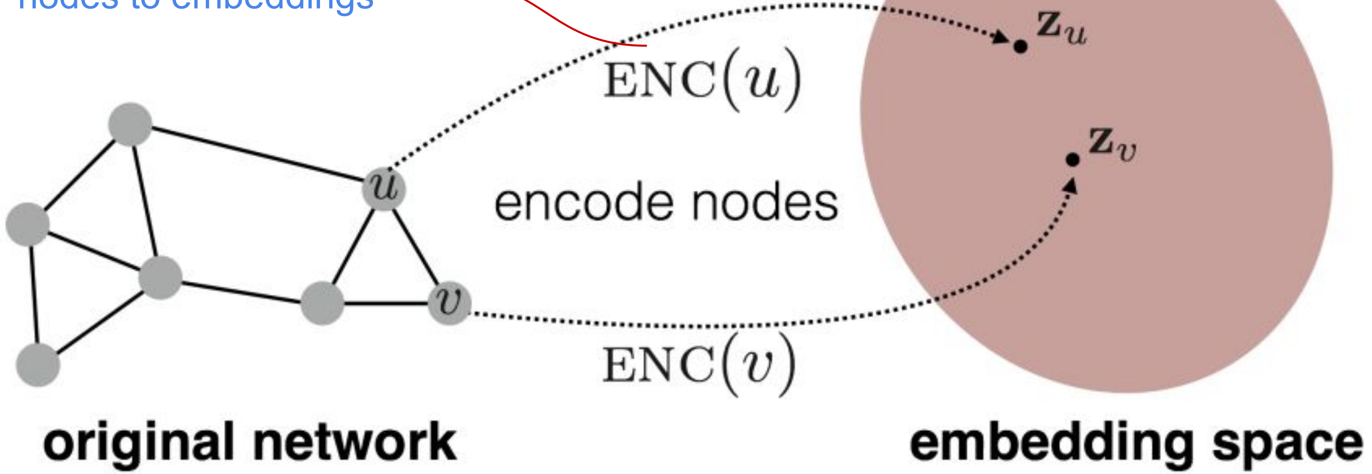
Embedding nodes

$$\text{similarity}(u, v) \approx \mathbf{z}_v^{\mathrm{T}} \mathbf{z}_u$$

In the original network

Similarity of the embeddings

$\mathbf{ENC}(v)$: an encoder maps from nodes to embeddings

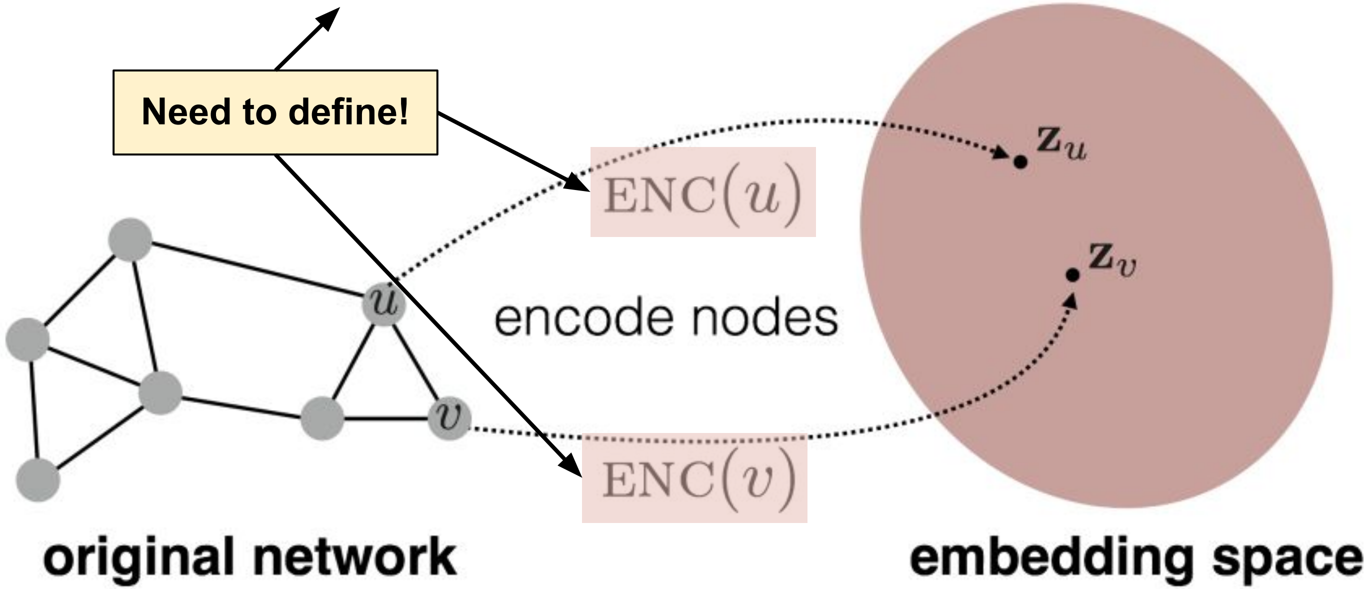


Embedding nodes

$$\text{similarity}(u, v) \approx \mathbf{z}_v^{\mathrm{T}} \mathbf{z}_u$$

In the original network

Similarity of the embeddings



Adapted from CS224W, Jure Leskovec

Two key components

- **Encoder**: maps each node to a low-dimensional vector

$$\mathrm{ENC}(\boldsymbol{v}) = \mathbf{z}_v$$

d-dimensional vector (embedding)

Node in the input graph

- **Similarity function**: specifies how the relationships in vector space map to the relationships in the original network

$$\mathrm{similarity}(u, v) \approx \mathbf{z}_v^{\mathrm{T}}\mathbf{z}_u$$

Similarity of $\boldsymbol{u}$ and $\boldsymbol{v}$ in the original network

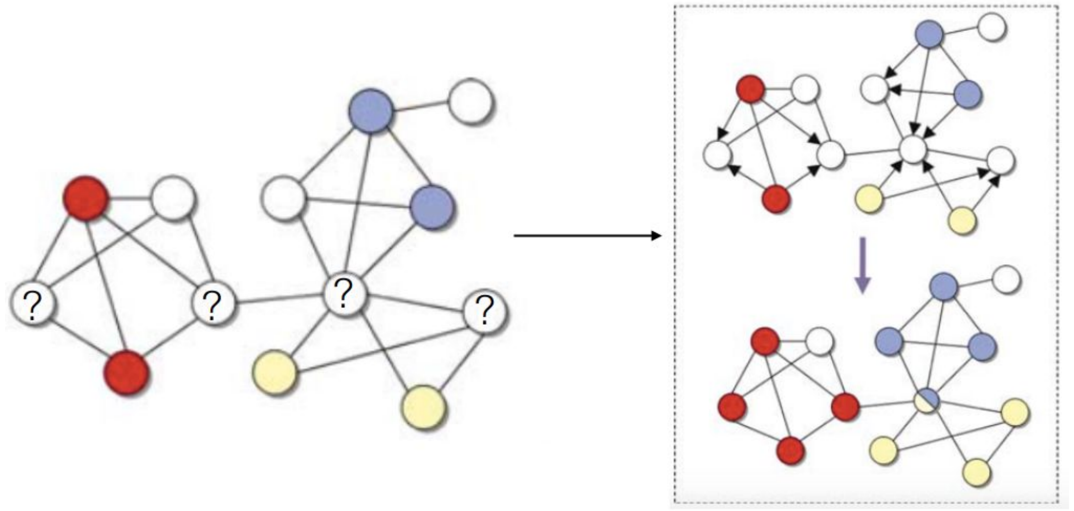
Dot product between node embeddings

How to define node similarity in the network?

- Key choice of methods is how to define node similarity.
- Should two nodes have a similar embedding if they …
  - are linked?
  - share neighbors?
  - …
- This lecture: define node similarity based “topological roles” of each node with respect to other nodes.
- Two graph representation learning algorithms:
  - Diffusion component analysis [DCA] (Cho et al, 2016, *Cell Systems*)
  - Node2vec (Grover et al, 2016, *KDD*)

Motivating example

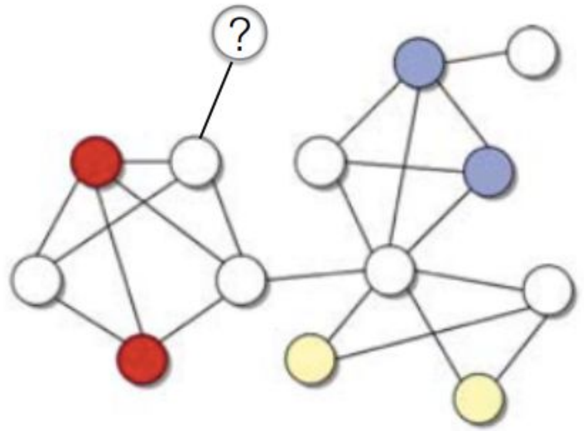
Example: protein function prediction



Voting by direct neighbors

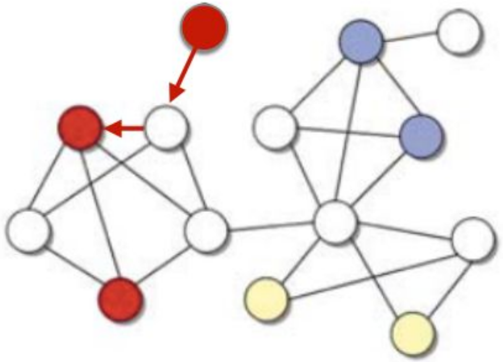
Adapted from CS598, Jian Peng

If there is no direct neighbor with known function



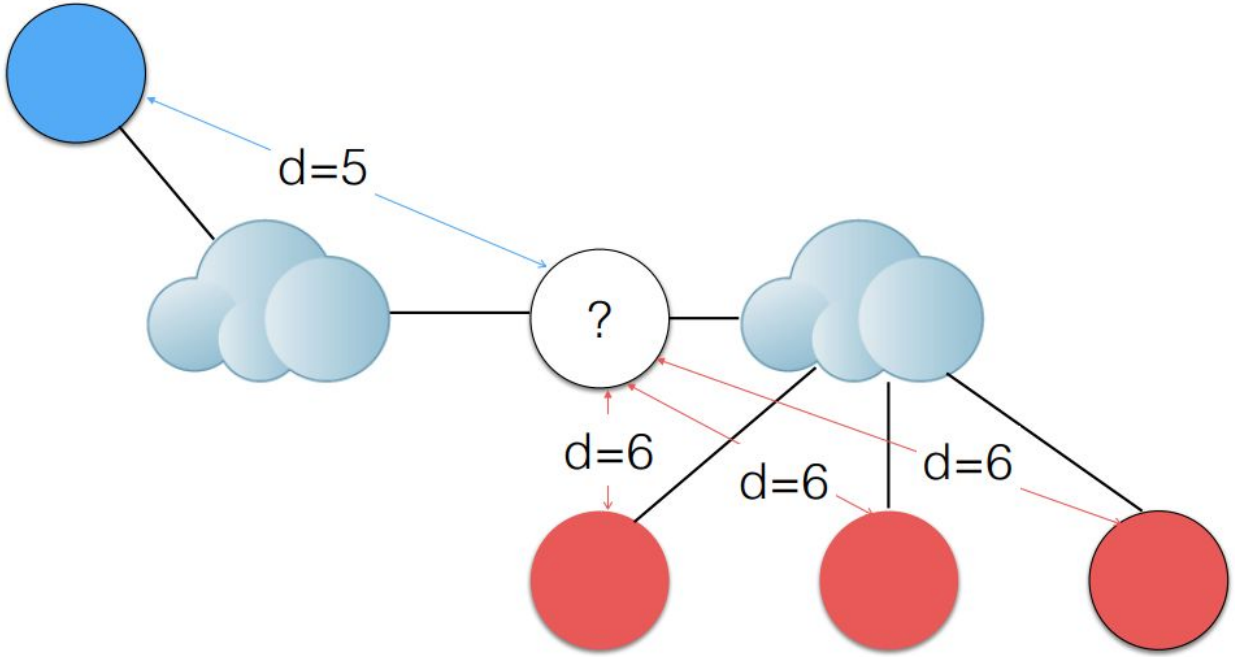
Adapted from CS598, Jian Peng

Shortest path



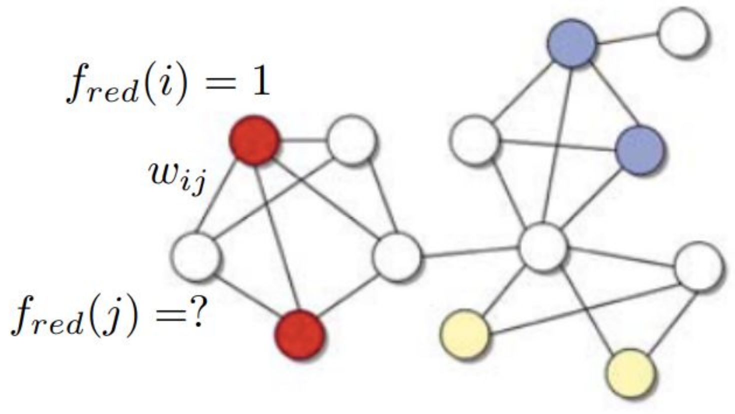
Floyd-Warshall algorithm: all pairwise distances
Computational Complexity: $O(n^3)$

Is shortest path a good metric?



Adapted from CS598, Jian Peng

Label propagation algorithm



Connected nodes tend to have similar function (color).

$$\min_{f_{red}} \sum_{(i,j)\in E} w_{ij}(f_{red}(i) - f_{red}(j))^2$$

$$\forall i \in RED, f_{red}(i) = 1$$

How to solve this problem?

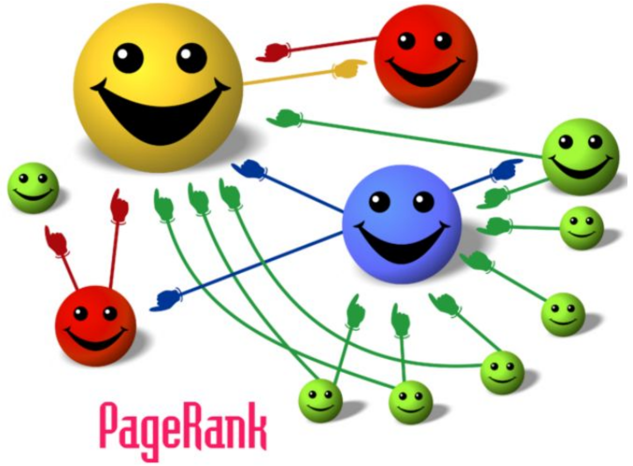
Diffusion component analysis (DCA)

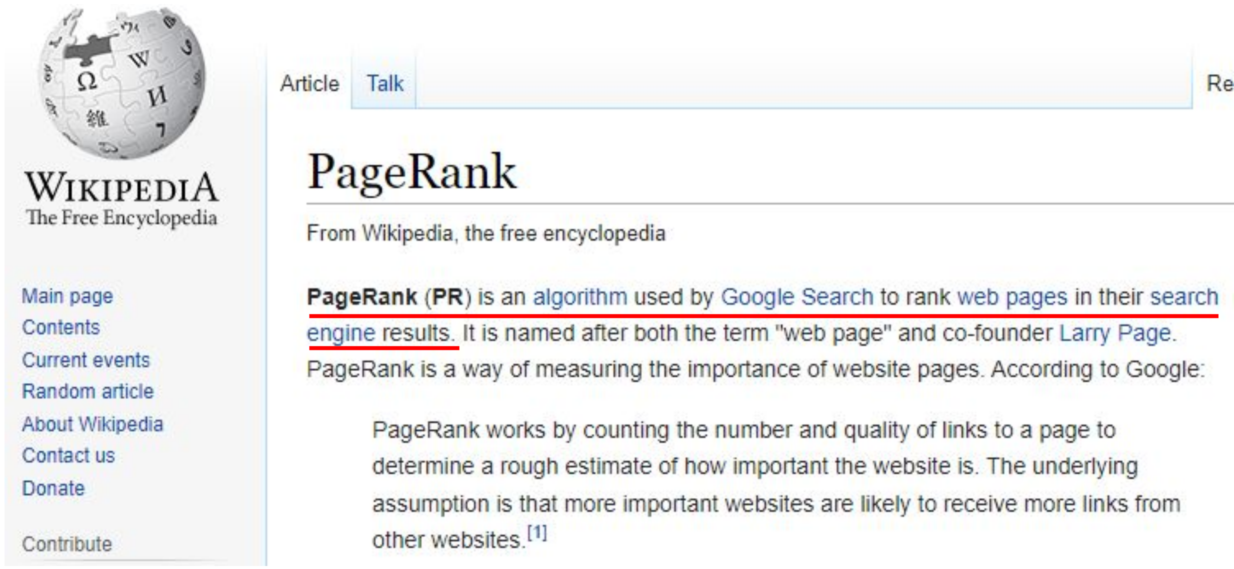
[1] Cho, Hyunghoon, Bonnie Berger, and Jian Peng. "Diffusion component analysis: unraveling functional topology in biological networks." *International Conference on Research in Computational Molecular Biology*. Springer, Cham, 2015.

[2] Cho, Hyunghoon, Bonnie Berger, and Jian Peng. "Compact integration of multi-network topology for functional analysis of genes." *Cell systems* 3.6 (2016): 540-548.

Slides adapted from UIUC CS598, Jian Peng

Random walk and Pagerank





PageRank

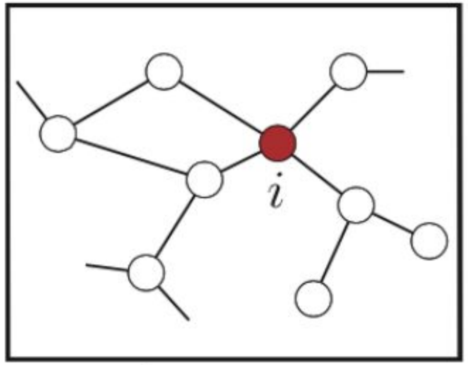
From Wikipedia, the free encyclopedia

PageRank (**PR**) is an algorithm used by Google Search to rank web pages in their search engine results. It is named after both the term "web page" and co-founder Larry Page. PageRank is a way of measuring the importance of website pages. According to Google:

> PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.[1]

Random walk with restart

Start from node i

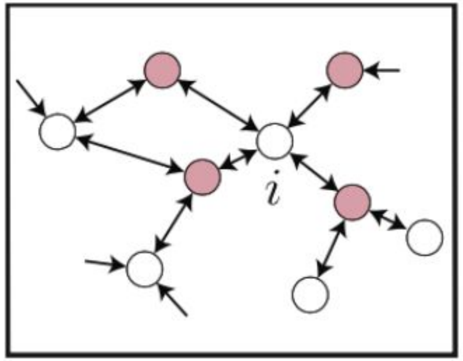


Initialization

$$s_i = (0, 0, ..., 1, ..., 0)$$

Random walk with restart

Distribute to neighbors



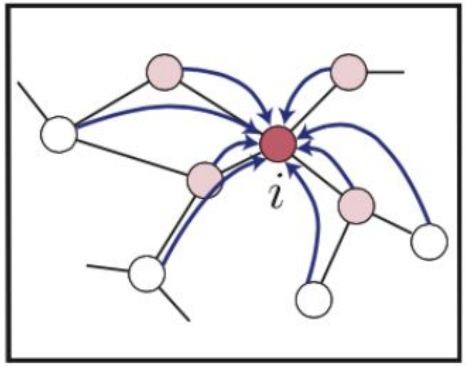
Propagation

four neighbors

$$s_i = (0, ..., 0.25, ..., 0.25, ..., 0.25, ..., 0.25, ..., 0)$$

Random walk with restart

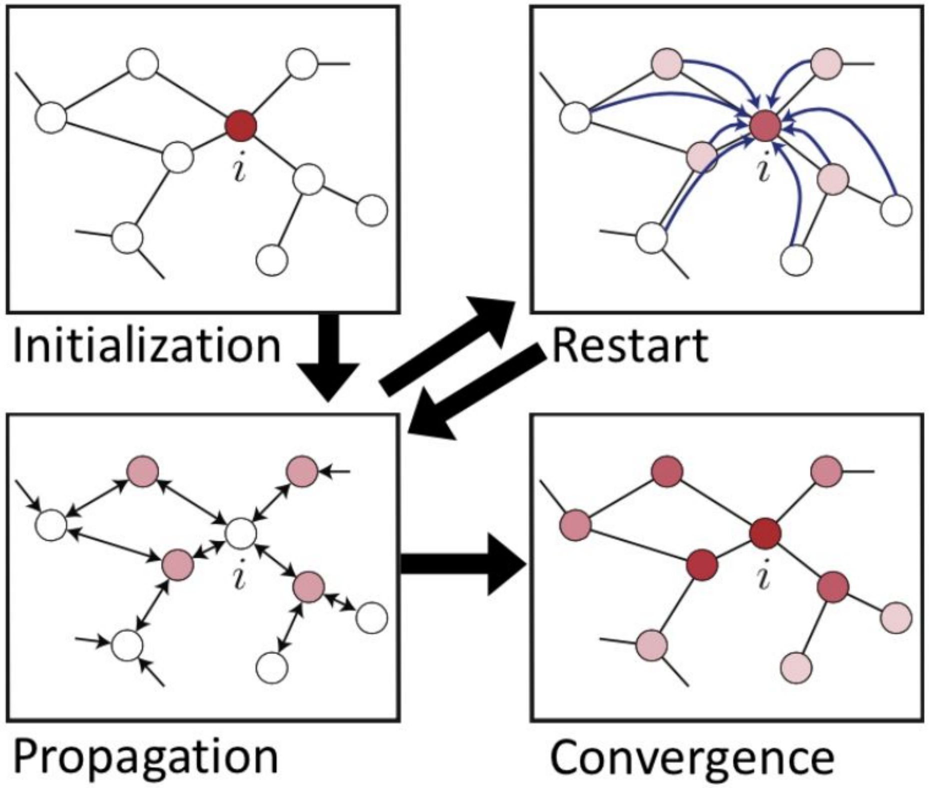
Shinkage and restart from node i



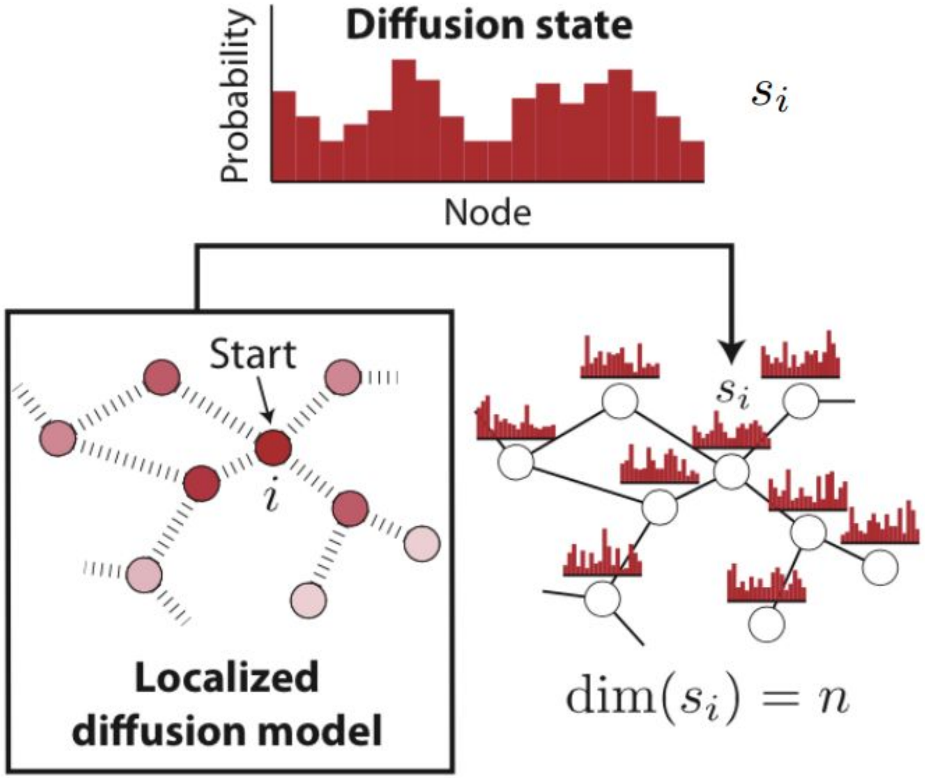
Restart

$$s_i = (0, ..., 0.125, ..., 0.125, ..., 0.5, ..., 0.125, ..., 0.125, ..., 0)$$

Random walk with restart



Random walk with restart



Random walk with restart

Adjacency matrix	$A : A_{ij}$
Transition matrix	$B \in R^{n \times n} : B_{ij} = A_{ij} / \sum_k A_{ik}$
Restart probability	p

$\boxed{\text{Algorithm: Repeat}}$

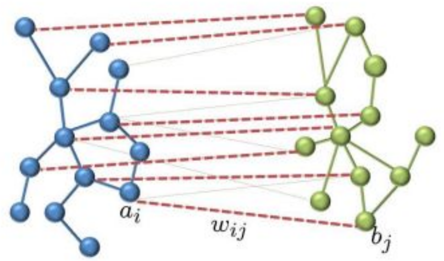
$$\forall i, j \qquad s_i(j) = (1-p) \sum_k s_i(k) B_{kj} + p\delta(i = j)$$

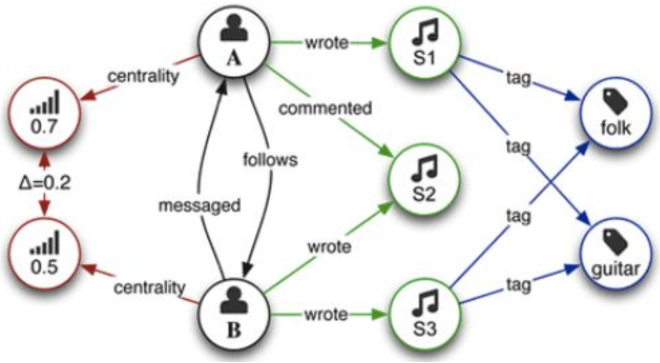
$\boxed{\text{Matrix operation}}$

$$S^{new} = (1-p) S^{old} B + p I_n$$

Random walk with restart

- **Comments**:
  - Very simple implementation
  - Capture long-range relationship in the graph
  - Robust to missing edges
  - Many applications in social network analysis, web data analysis, and bioinformatics





Weighted Voting

Assign color vector:

$$\forall i \in RED, f_{red}(i) = 1$$

$$\forall i \notin RED, f_{red}(i) = 0$$

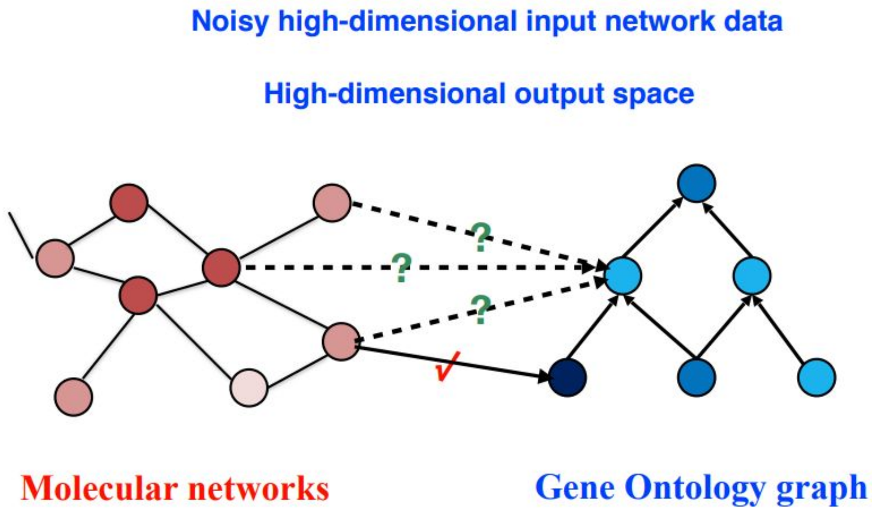
Majority voting $$p_{red}(i) = \sum_j Sim(i,j) f_{red}(j)$$

Some good similarity metrics:

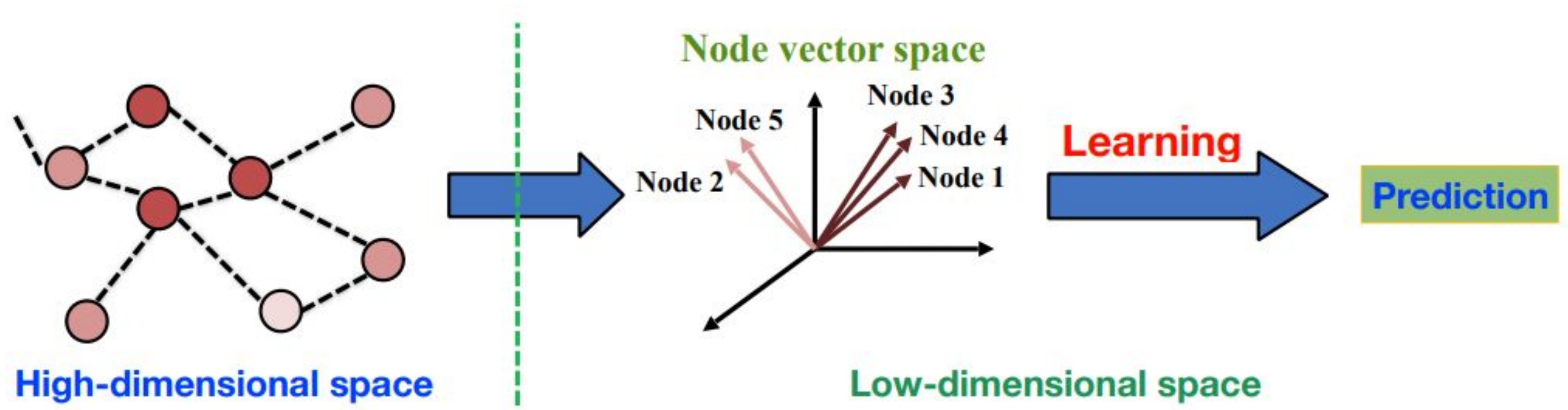
1. reciprocal of shorted distance	$1/d(i,j)$
2. random walk similarity	$s_i(j) + s_j(i)$
3. diffusion state similarity	$1/\|s_i - s_j\|_1$

Supervised learning: protein function prediction

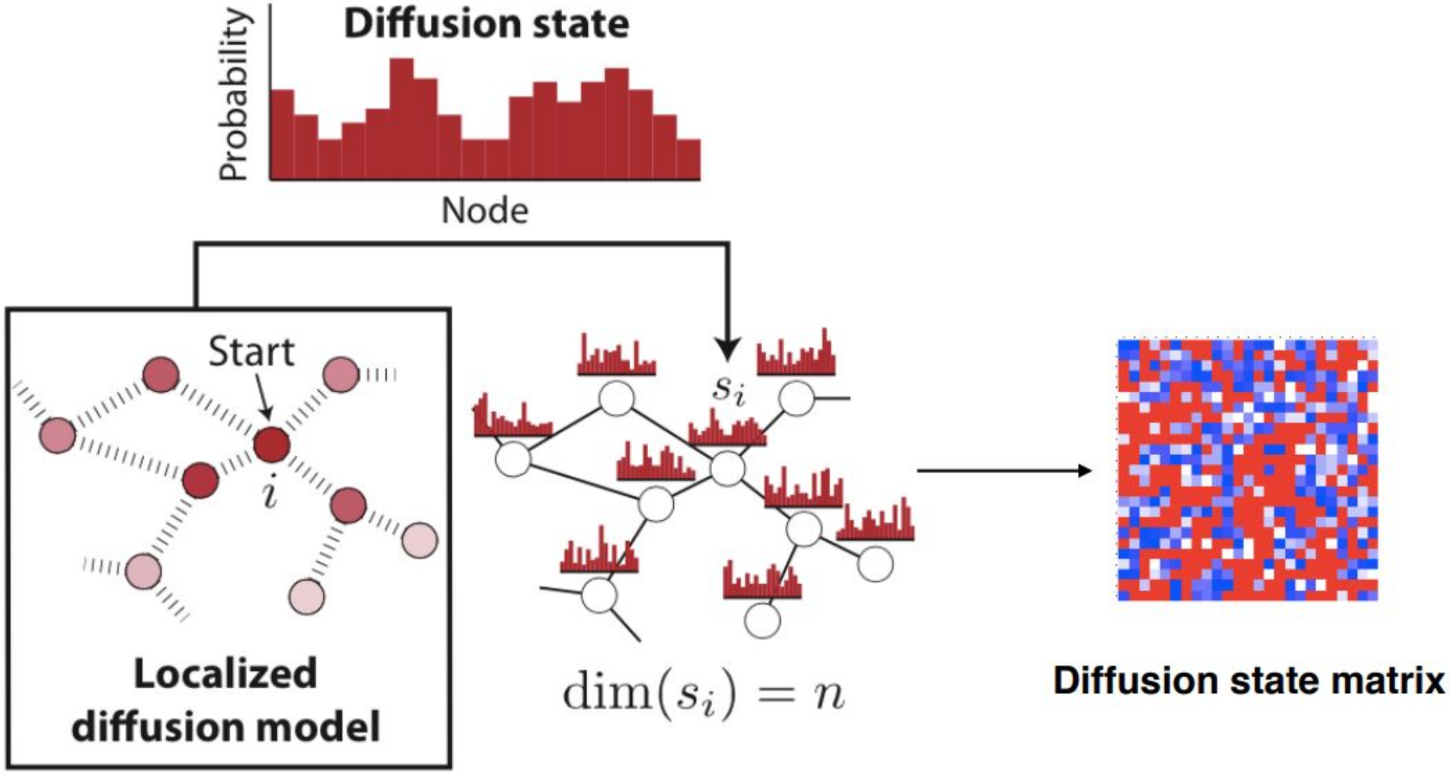
- More than 16K proteins in human protein-protein interaction network
- There may be many fake and missing edges in the network
- Nearly half of the functions only have very few annotations
- Not enough training samples for these functions
- A simple classifier will be overfitting


Noisy high-dimensional input network data
High-dimensional output space
?
?
?
✓
Molecular networks
Gene Ontology graph

Dimensionality reduction

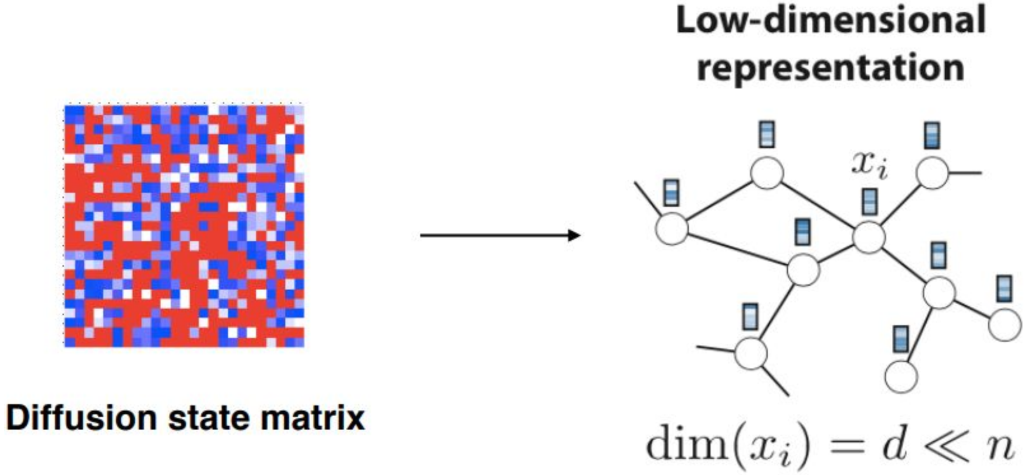


After we run random walk with restart



We can run SVD or NMF on *S*

- Let us assume that we don't know how this diffusion state matrix was generated



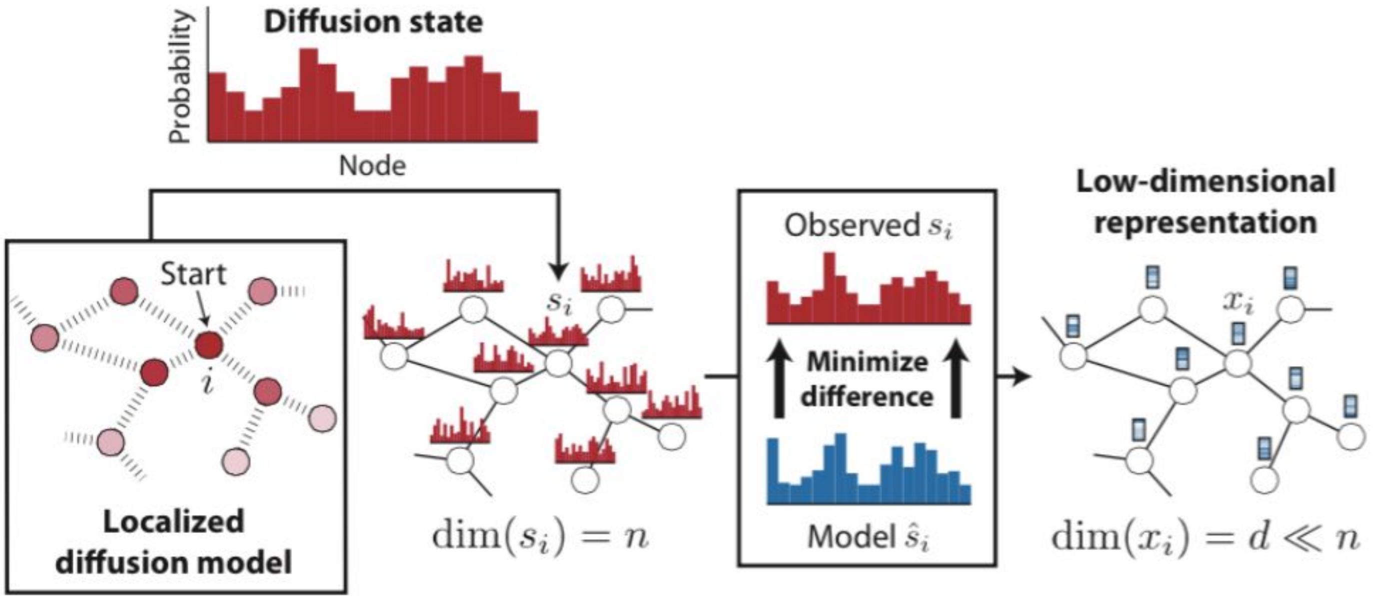
Any issues with these dimensionality reduction algorithms?

$$\min_{X,W} \|S - XW\|_2^2$$

$$X \in R^{n\times d}, W \in R^{d\times n}$$

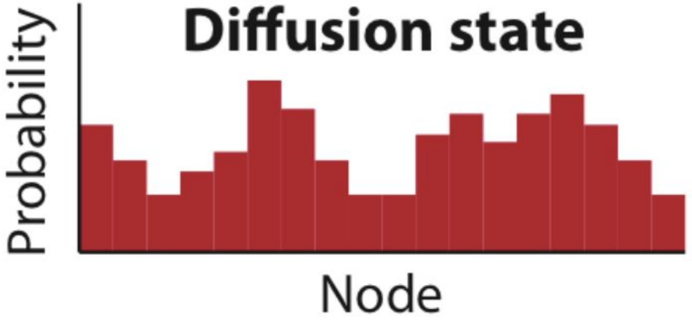
Diffusion component analysis (DCA)

- **Idea**: we hope to construct a model to approximate observed diffusion states



After random walk (diffusion)

- $s_i(j)$ gives the probability of reaching node j after the random walk from node i

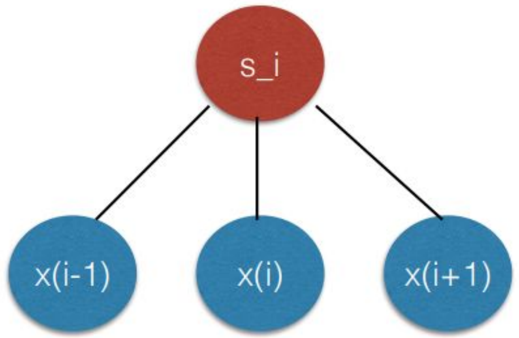


$$\sum_j s_i(j) = 1$$

$$\forall i, j \qquad s_i(j) \geq 0$$

What would be a good way to model diffusion states?

Logistic regression



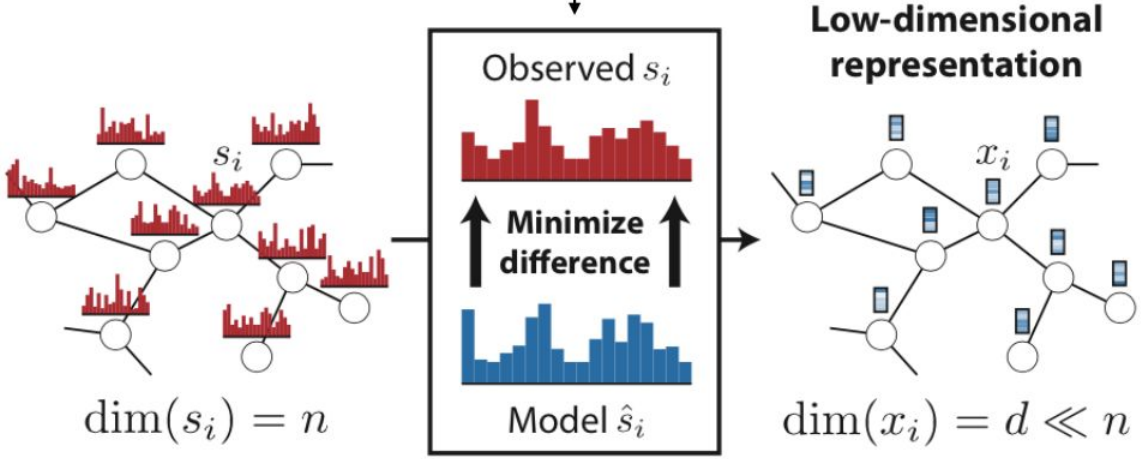
$$\hat{s}_i(j) = \frac{\exp(w_i^T x_j)}{\sum_k \exp(w_i^T x_k)}$$

Is it better than SVD/NMF?

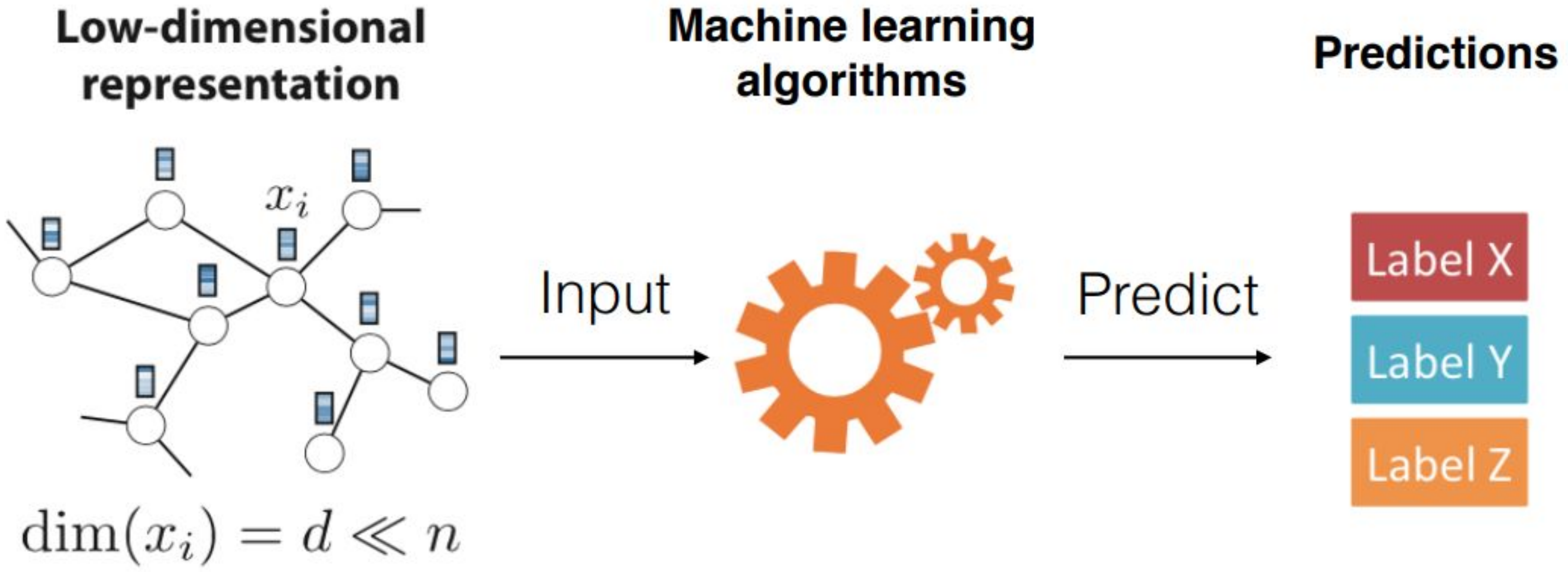
Matching observed and model data

$$\min_{w,x} \sum_i KL(s_i \| \hat{s}_i)$$

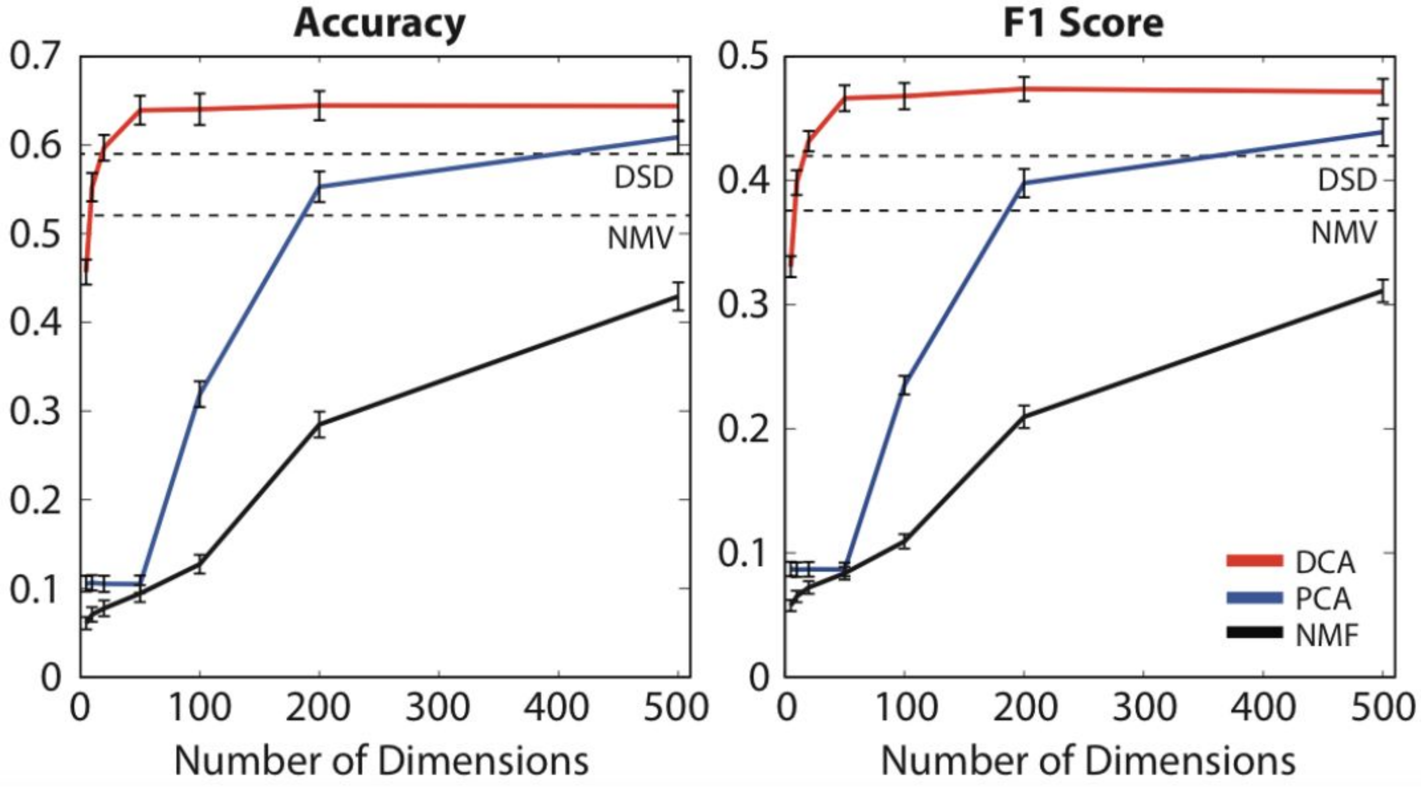
with $$\hat{s}_i(j) = \frac{\exp(w_i^T x_j)}{\sum_k \exp(w_i^T x_k)}$$



Machine learning with network data



Comparisons



Analogy to PCA

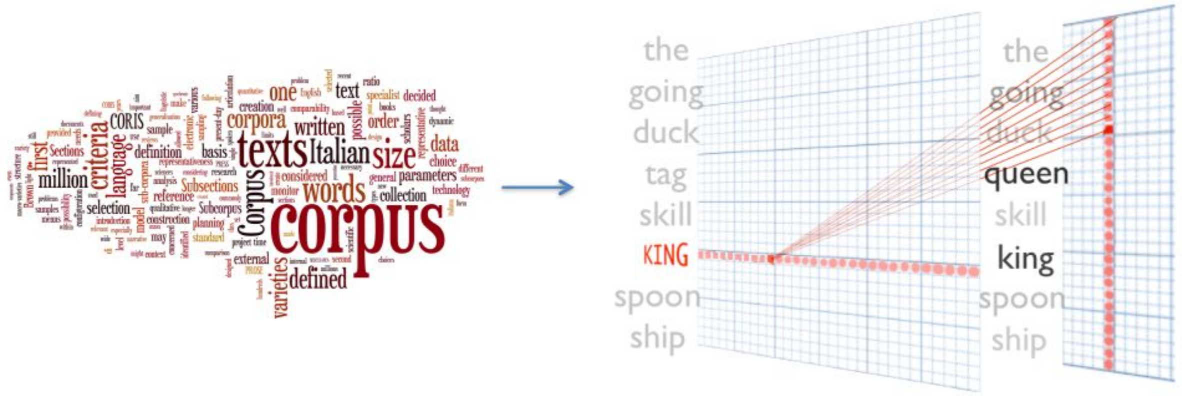
Principal component analysis (PCA)

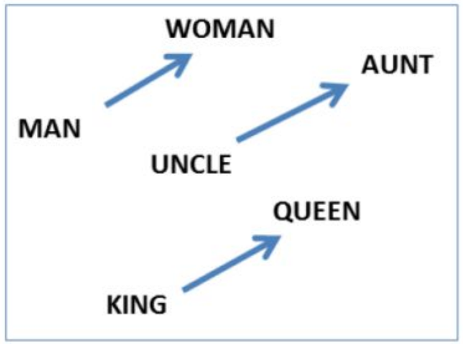
- Input: **matrix** data
- Goal: find low-rank approximation that best explain the **variance** of the matrix input

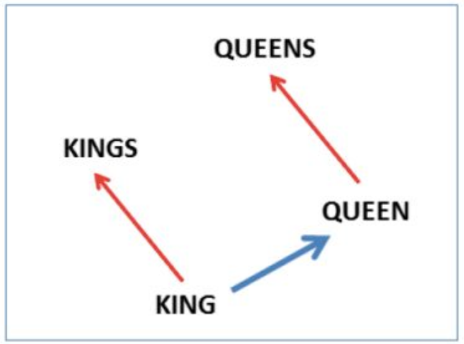
Diffusion component analysis (DCA)

- Input: **network** data
- Goal: find low-dim representations that best explain the **topological pattern** of the network input

Similarity to word2vec



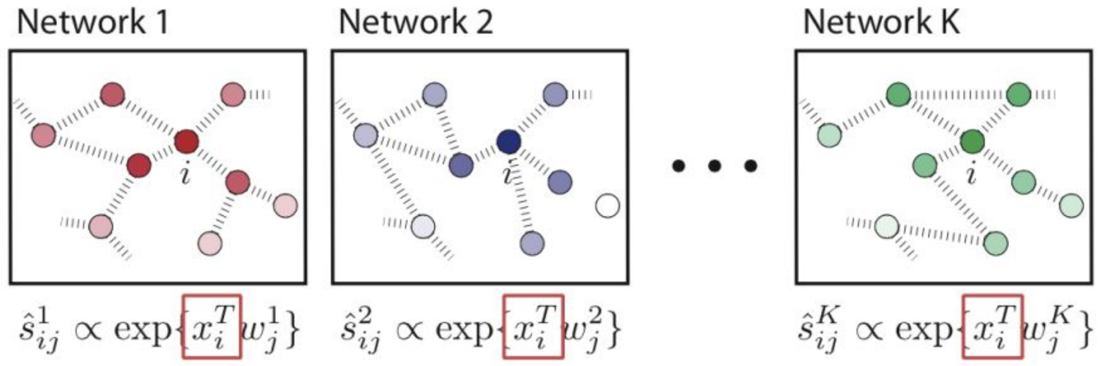




If we have many networks

Step 1: run diffusion within each network

Step 2: jointly optimize vectors over all networks


Network 1
Network 2
Network K
i
i
i
$\hat{s}^1_{ij} \propto \exp\{\boxed{x_i^T} w_j^1\}$
$\hat{s}^2_{ij} \propto \exp\{\boxed{x_i^T} w_j^2\}$
$\hat{s}^K_{ij} \propto \exp\{\boxed{x_i^T} w_j^K\}$

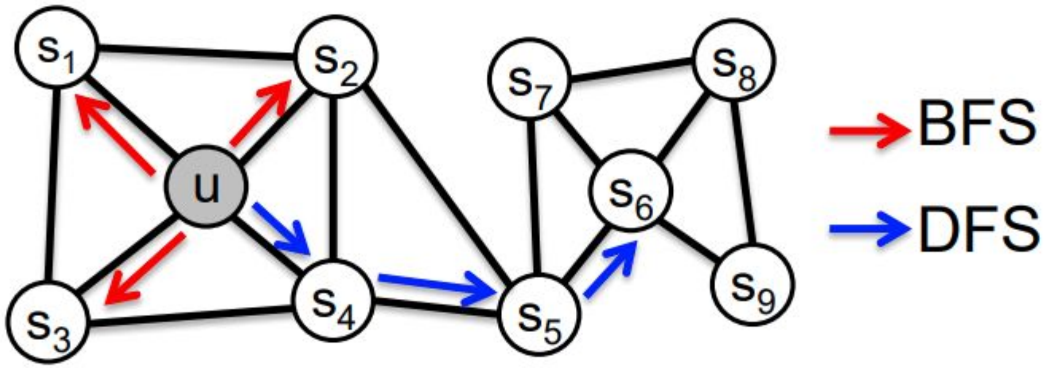
Shared vectors capture global patterns

node2vec

Grover et al. “node2vec: Scalable Feature Learning for Networks”. *KDD*, 2016

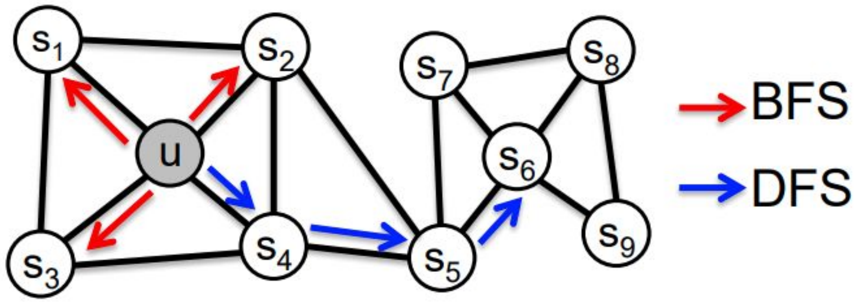
node2vec

- **Goal**: Embed nodes with similar network neighborhoods close in the feature space
- **Idea**: Use flexible, **biased random walks** that can trade off between **local** and **global** views of the network



Biased walks

Two classic strategies to define a neighborhood ***N(u)*** of a given node ***u***



Walk of length 3 (***N(u)*** of size 3):

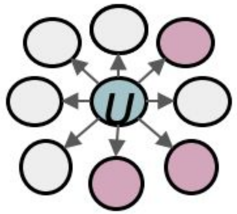
$$N_{BFS}(u) = \{ s_1, s_2, s_3 \}$$

Local microscopic view

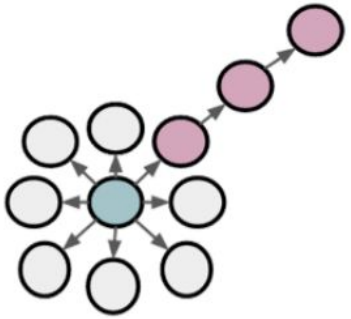
$$N_{DFS}(u) = \{ s_4, s_5, s_6 \}$$

Global microscopic view

BFS & DFS



BFS:
Micro-view of neighbourhood



DFS:
Macro-view of neighbourhood

How to interpolate BFS & DFS?

Biased fixed-length random walk ***R*** that, given a node ***u***, generates neighborhood $N_R(u)$

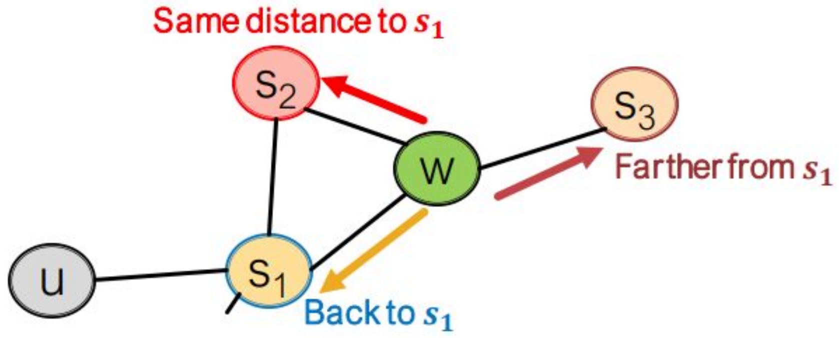
Two parameters:

- Return parameter ***p***:
  - Return back to the previous node
- In-out parameter ***q***:
  - Moving outwards (DFS) vs. inwards (BFS)
  - Intuitively, ***q*** is the “ratio” of BFS vs. DFS

Biased random walk

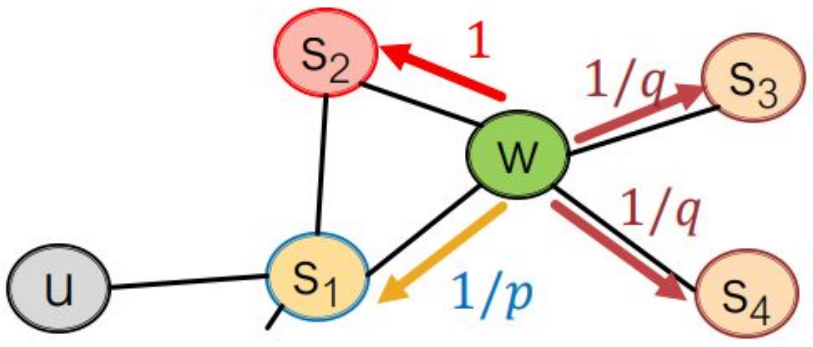
Biased 2nd-order random walks explore network neighborhoods

- Random walk just traversed edge (***s_1***, ***w***) and is now at ***w***
- Observation: Neighbors of ***w*** can only be



Biased random walk

Walker came over edge (s_1, w) and is now at w. Where to go next?



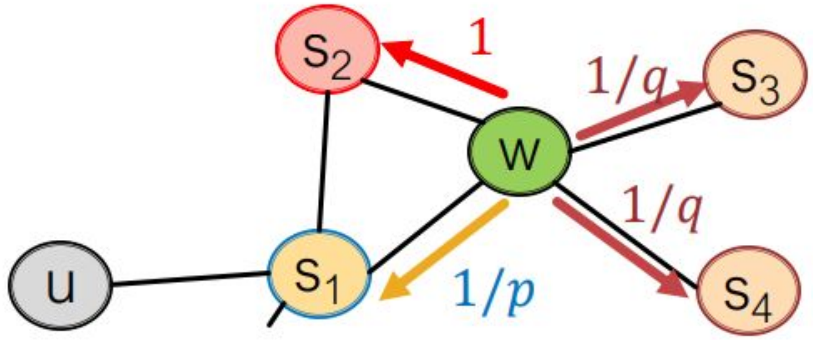
1/p, 1/q, and 1 are **unnormalized** probabilities

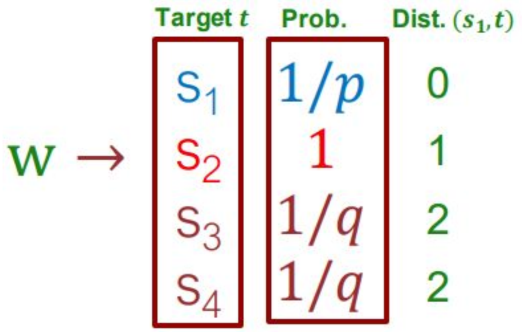
p, ***q*** model transition probabilities

- ***p***: return parameter
- ***q***: "walk away" parameter

Biased random walk

Walker came over edge (***s_1***, ***w***) and is now at ***w***. Where to go next?





$w \rightarrow$

Target t	Prob.	Dist. (s_1, t)
s_1	$1/p$	0
s_2	1	1
s_3	$1/q$	2
s_4	$1/q$	2

1/p, 1/q, and 1 are **unnormalized** probabilities

- **DFS-like walk**: High value of ***1/p***
- **BFS-like walk**: High value of ***1/q***
- **Random walk $N_R(u)$**: nodes visited by the biased walk

Representation learning framework

- Given ***G* = (*V*, *E*)**
- Goal: learn a mapping ***f***: ***u*** -> $\boldsymbol{R^d}$: $\boldsymbol{f(u)=z_u}$
- Intuition: learn representations such that given the representation $\boldsymbol{z_u}$ of node ***u***, we can predict what its neighbors $\boldsymbol{N_R(u)}$ are
- **Log-likelihood objective**

$$\max_f \sum_{u \in V} \log \mathrm{P}(N_{\mathrm{R}}(u) | \mathbf{z}_u) \longleftarrow \text{Maximum likelihood objective}$$

$\boldsymbol{N_R(u)}$ is the neighborhood of node ***u*** by strategy ***R***

Optimization

- **Log-likelihood objective**

$$\max_{f} \sum_{u \in V} \log \mathrm{P}(N_{\mathrm{R}}(u) | \mathbf{z}_u)$$

- Equivalently

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log(P(v|\mathbf{z}_u))$$

- **Intuition**: maximize the similarity between ***u*** and other nodes in the walk $\boldsymbol{N_R(u)}$

- Parameterize $\boldsymbol{P(v|z_u)}$ using softmax

$$P(v|\mathbf{z}_u) = \frac{\exp(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_n)}$$

Optimization

- Putting it all together

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log\left(\frac{\exp(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_n)}\right)$$

- Finding representations $\boldsymbol{z_u}$ that minimize $\boldsymbol{L}$

Optimization

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log\left(\frac{\exp(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_v)}{\boxed{\sum_{n \in V} \exp(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_n)}}\right) \longrightarrow$$

Expensive to compute.
Need approximation.

- **Solution**: Negative sampling

$$\log\left(\frac{\exp(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_n)}\right)$$

$$\approx \log\left(\sigma(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_v)\right) - \sum_{i=1}^{k} \log\left(\sigma(\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_{n_i})\right),\ n_i \sim P_V$$

Sigmoid function

(Makes each term a "probability" in [0, 1])

Random distribution voer nodes

- **Idea**: Instead of normalizing w.r.t all nodes, just normalize against ***k*** random negative samples
- Sample ***k*** negative nodes each with prob. proportional to its degree

node2vec algorithm

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

- Linear-time complexity
- All 3 steps are individually parallelizable

Other random walk-based methods

- Different kinds of biased random walks:
  - Based on node attributes (Dong et al., 2017)
  - Based on learned weights (Abu-El-Haija et al., 2017)
- Alternative optimization schemes:
  - Directly optimize based on 1-hop and 2-hop random walk probabilities (e.g., LINE from Tang et al. 2015)
- Network preprocessing techniques:
  - Run random walks on modified versions of the original network (e.g., Ribeiro et al. 2017's struct2vec, Chen et al. 2016's HARP)

Summary of today

- Representations learning in graphs
    - Learning embeddings that capture structure/topological similarity between network nodes
- Unsupervised representation learning framework
    - Random-walk based
- Two algorithms
    - Diffusion Component Analysis (DCA)
    - Node2vec