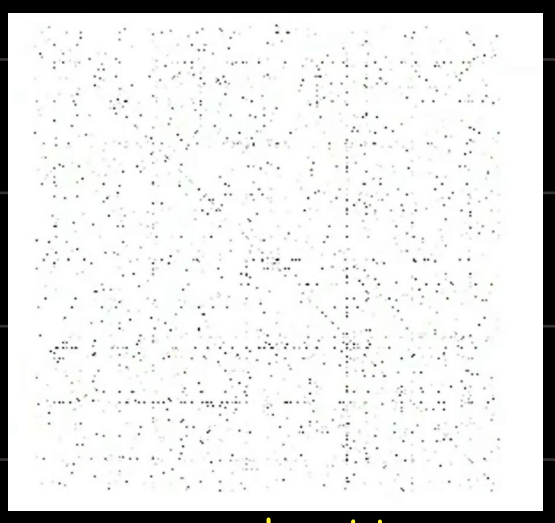
Graph Representation

Def Adjacency Matrix

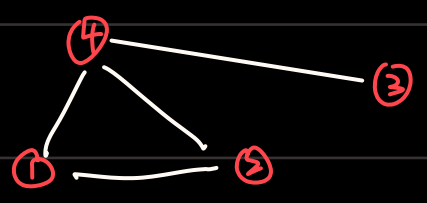
$A \in M_{N \times N}$ s.t $\begin{cases} A_{ij} = 1 & \text{undirect } i \leftrightarrow j \text{ / direct } i \to j \\ A_{ij} = 0 & \text{O.W} \end{cases}$

Property → A very sparse matrix →



Social Network (Big Data)

① Undirected Graph



① : 12, 14
② : 21, 24
③ : 34
④ : 41, 42, 43

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$ Node ②

$A_{ij} = A_{ji}$

$A_{ii} = 0$

Node Degree

$$k_i = \sum_j^N A_{ij} = \sum_j^4 A_{ij}$$

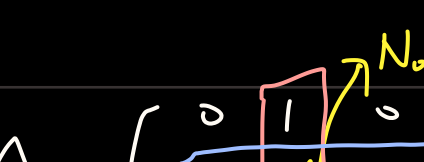
$$k_j = \sum_i^N A_{ij} = \sum_i^4 A_{ij}$$

Avg nd

$$\bar{k} = \frac{2E}{N} = \frac{2 \cdot 4}{4}$$

$$L = \frac{1}{2}\sum_i^N k_i - \frac{1}{2}\sum_{i,j}^N A_{ij}$$

② directed Graph



① : 14
② : 21
③ : no
④ : 42, 43

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$ out, In, Node ④

$A_{ij} \neq A_{ji}$

$A_{ii} = 0$

Node Degree

$$k_i^{out} = \sum_j^N A_{ij} = \sum_j^4 A_{ij}$$

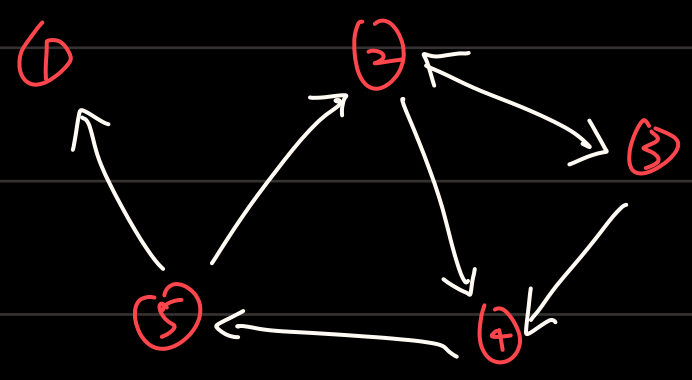
$$k_j^{in} = \sum_i^N A_{ij} = \sum_i^4 A_{ij}$$

Avg nd

$$\bar{k} = \frac{E}{N} = \frac{4}{4}$$

$$L = \sum_i^N k_i = \sum_j^N k_j^{out} = \sum_{ij}^N A_{ij}$$

Def Adjacency List



Out list

1 : no

2 : 3, 4

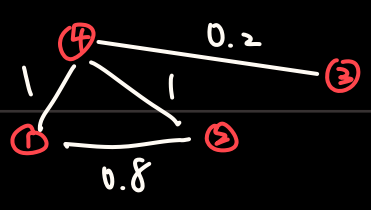
3 : 2, 4

4 : 5

5 : 1. 2

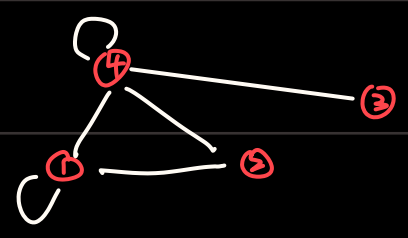
More Types

Weighted



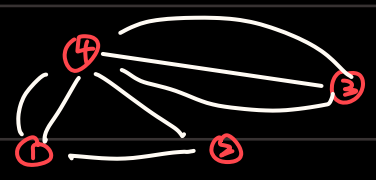
$$A = \begin{pmatrix} 0 & 0.8 & 0 & 1 \\ 0.8 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0.2 \\ 1 & 1 & 0.2 & 0 \end{pmatrix}$$

self-edges



$$A = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

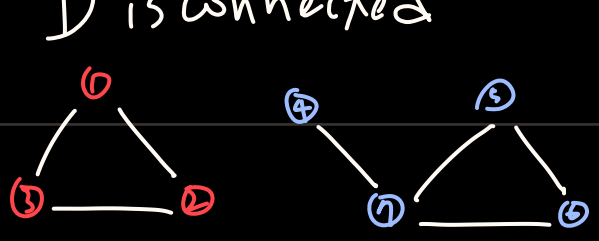
Multigraph



$$A = \begin{pmatrix} 0 & 1 & 0 & 2 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 3 \\ 2 & 1 & 3 & 0 \end{pmatrix}$$

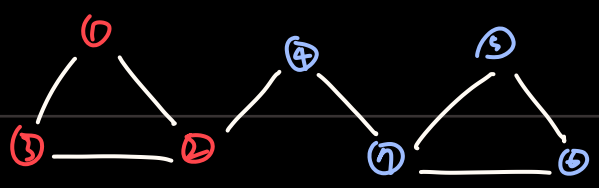
Connectivity

Disconnected



$$A = \left(\begin{array}{ccc|ccc} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right)$$

Connected



$$A = \left(\begin{array}{ccc|ccc} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{array}\right)$$

Types of Connectivity

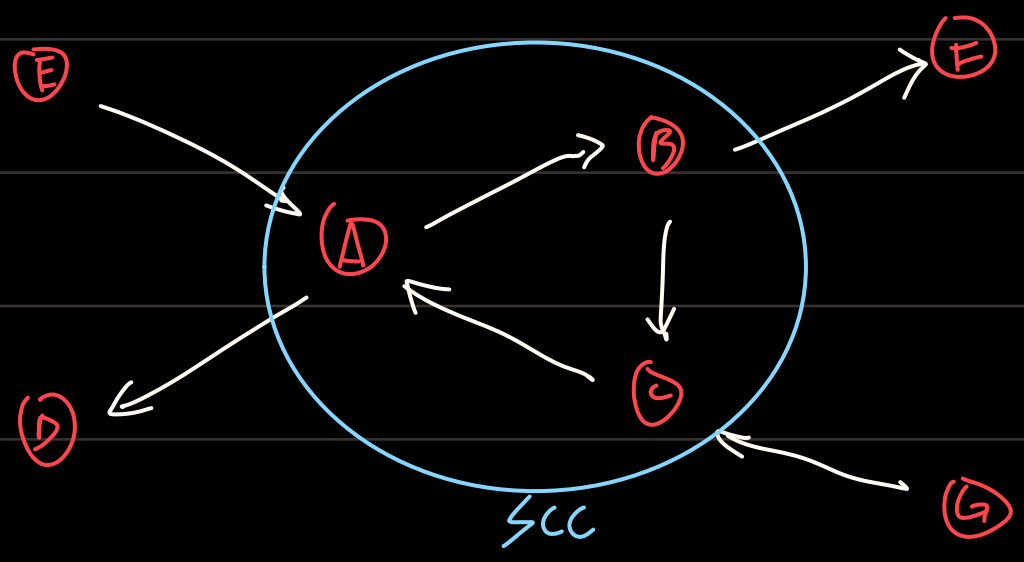
① Strongly connected directed graph

$A \longleftrightarrow B$

② Weakly connected directed graph

$A \longrightarrow B$

Strongly Connected Components (SCCs)



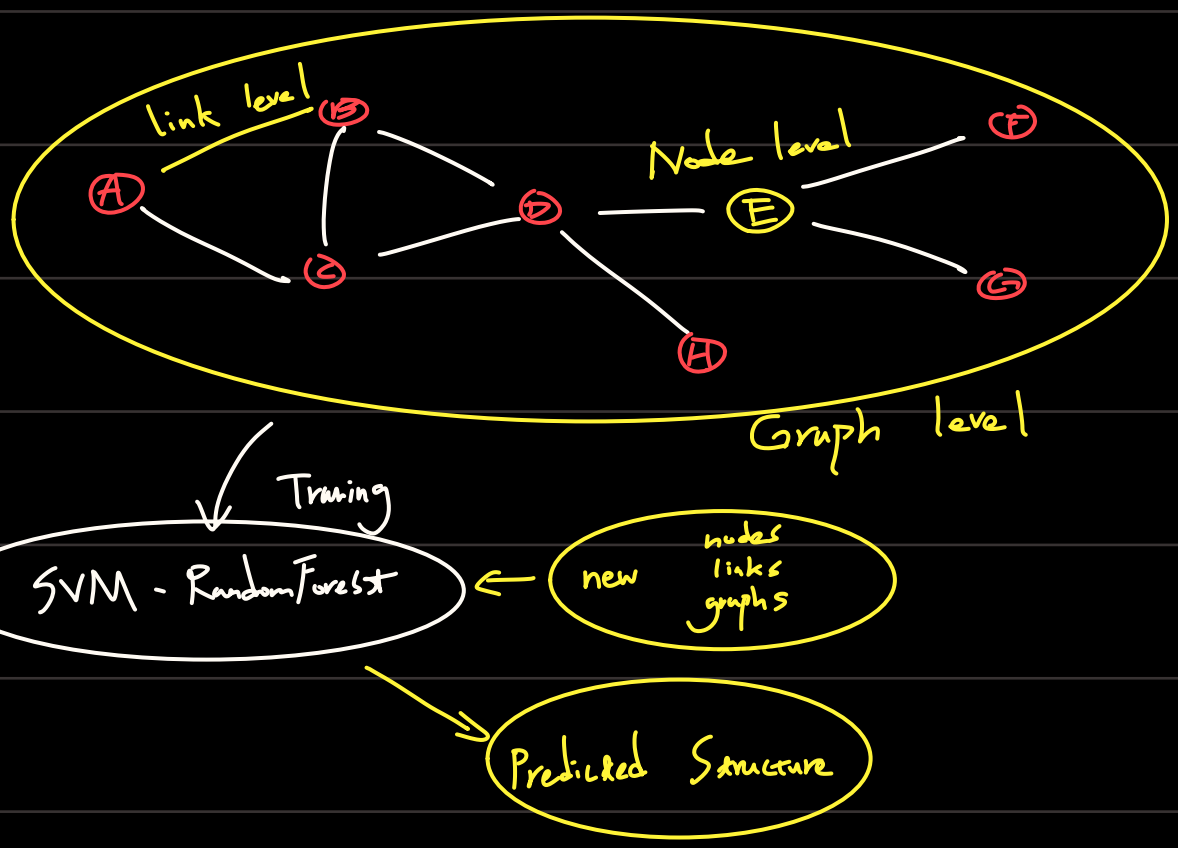
In - components : A . B . C

Out - components : E . D . F . G

Feature Based

Traditional Method For Machine Learning In Graphs

Structured Features : Node - Link - Graph



ML Basic Settings

What Model do? → Make a prediction for a set of objects

Features : $\in R^d$. Rrep for node / link / graph

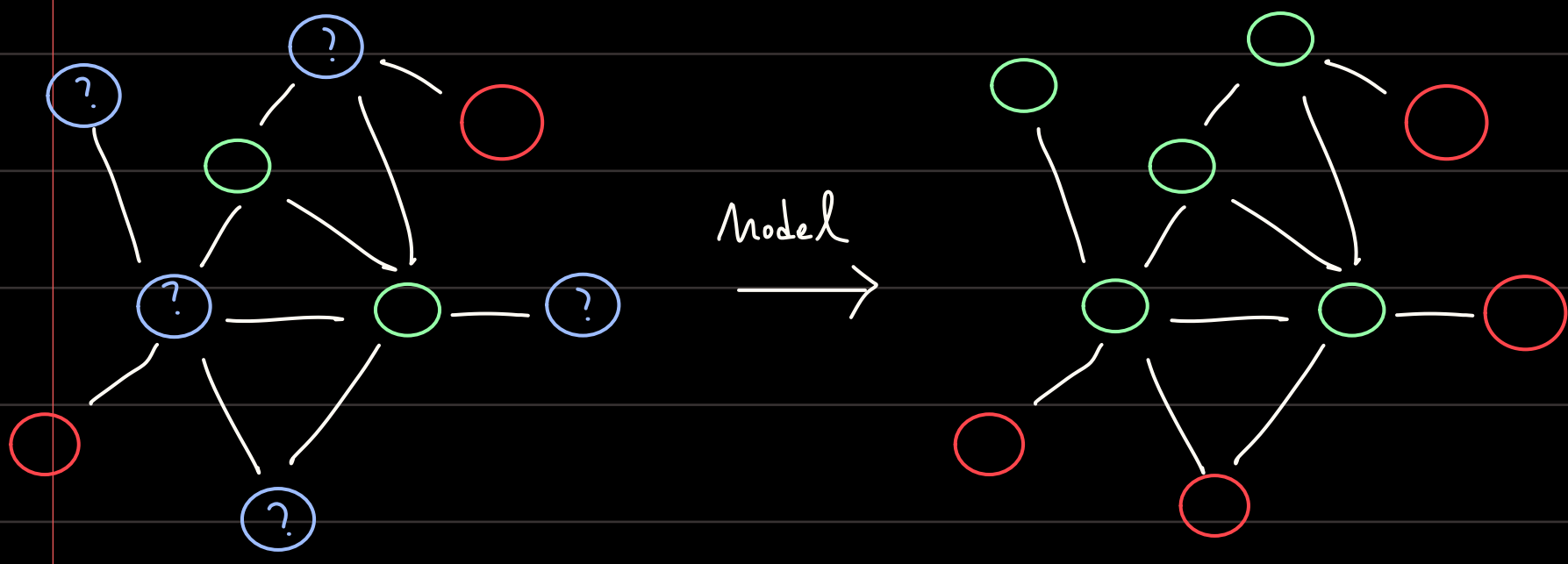
Object: Nodes - Edges · Set of nodes · Graphs

Objective : task specific

Given Graph $G: (V, E)$ → Build a function $f: V \to R$

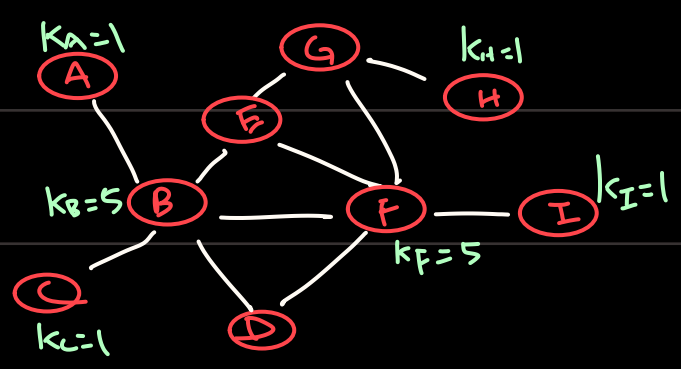
Node-Level Features

Node Classification → Condisider 2 labels : {○, ○}



Characterize Node Features : Node Degree · Node Centrality
Clustering Coefficient · Graphlets

① Node Degree



Weak : $k_A = k_C = k_H = k_I$, $k_B = k_F$ ⇒ model can't distinguish

② Node Centrality → How Important the Node is

Def Eigenvector Centrality

Node v is Important if Surrounding by important nodes $u \in N(v)$

$c_v = \frac{1}{\lambda} \sum_{u \in N(v)} c_u$, $\lambda > 0$ iff $\lambda c = Ac$, where A: Adjacency Matrix, c: Centrality Vector

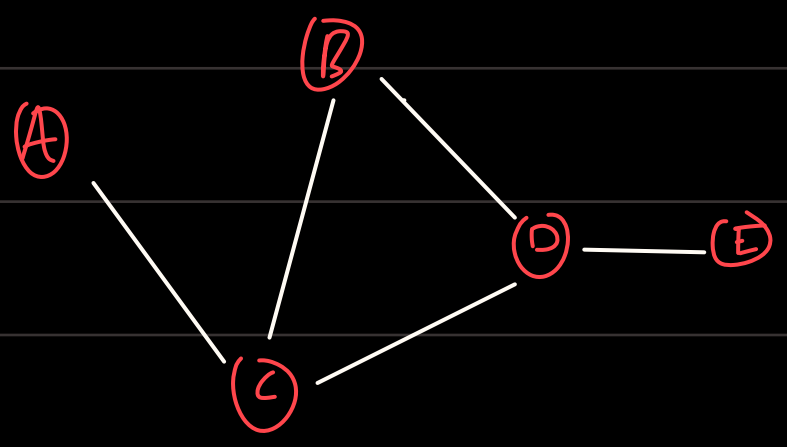
By Perron - Frobenius Thm → $\lambda_{max} > 0$ and unique

→ C_{max} is used for Centrality

(Def) Betweeness Centrality

Node v is important if lies on many shortest path between other nodes

$$C_V = \sum_{s \neq V \neq t} \frac{\text{Count: shortest paths between } s \text{ and } t \text{ that contains } V}{\text{Count: shortest paths between } s \text{ and } t}$$



$C_A = C_B = C_E = 0$

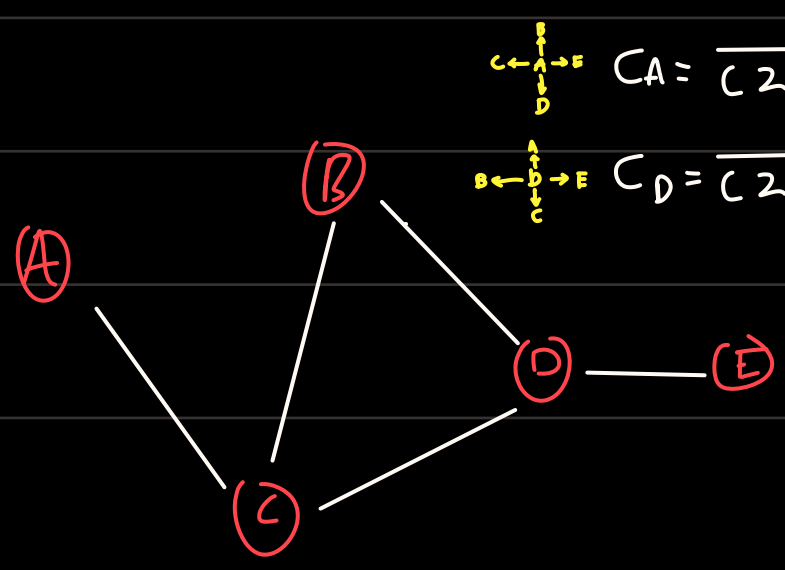
$C_C = 3 \rightarrow$ ACB (A→B), ACD (A→D), ACDE (A→E)

$C_D = 3 \rightarrow$ BDE (B→E), CDE (C→E), ACDE (A→E)

(Def) Closeness Centrality

Node v is important if it has small shortest path length to all other nodes

$$C_V = \frac{1}{\sum_{M \neq V} \text{shortest path length between } M \text{ and } V}$$



$C_A = \frac{1}{(2+1+2+3)} = \frac{1}{8} \rightarrow$ (ACB、AC、ACD、ACDE)

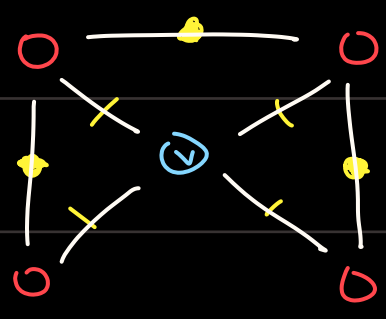
$C_D = \frac{1}{(2+1+1+1)} = \frac{1}{5} \rightarrow$ (DCA、DB、DC、DE)

Importance: $C_D > C_A$

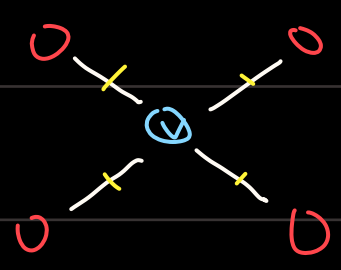
Def Clustering Coefficient

Measures how connected v's neighboring nodes

$$e_v = \frac{\text{Count: edges among neighboring nodes}}{\binom{k_v}{2}} \in [0, 1]$$

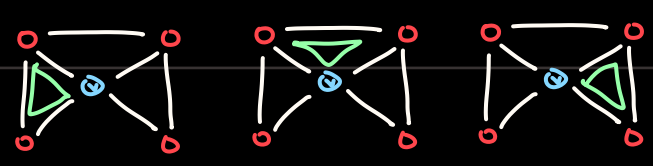


$e_v = \frac{1}{\binom{4}{2}} \cdot 3 = \frac{1}{2}$



$e_v = \frac{1}{\binom{4}{2}} \cdot 0 = 0$

friend of friend is not my friend

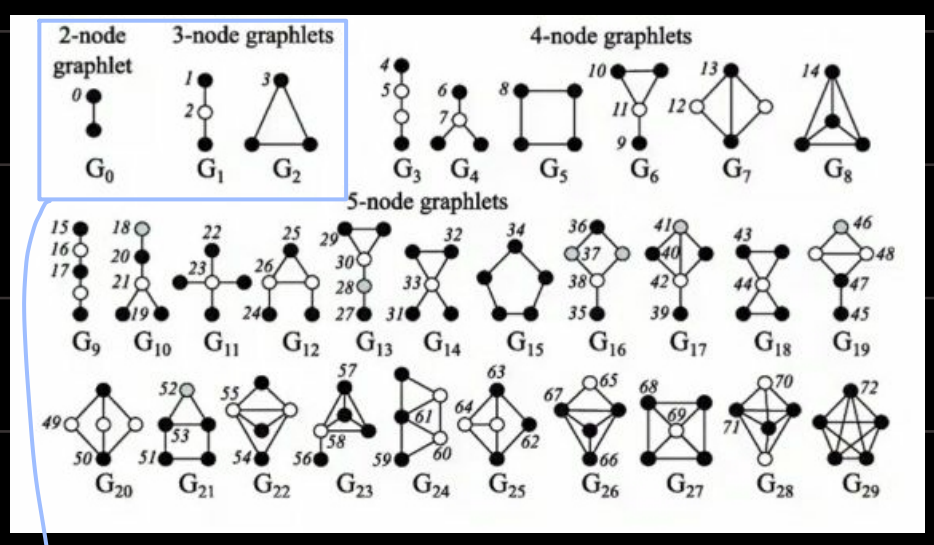


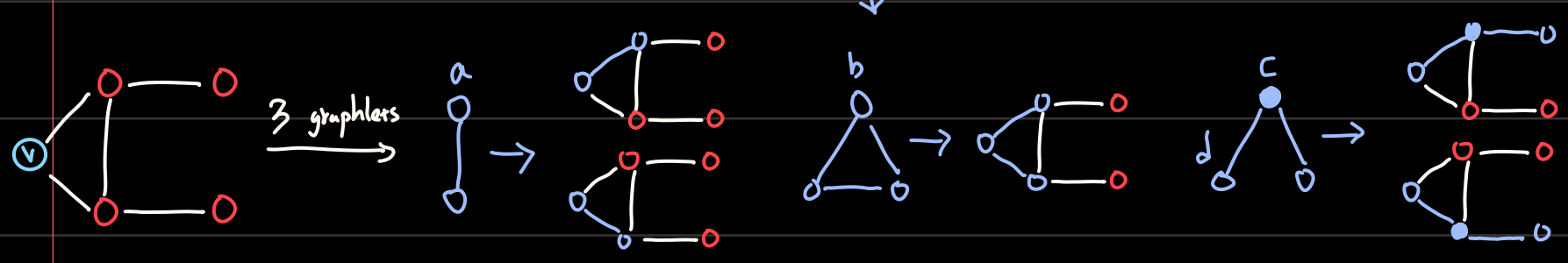
Social Network → friend of friend also my friend

Def Graphlets → Rooted Connected None-Isomorphic Subgraphs

Graphlet Degree Vector (GDV): Graphlet-base feature for node
(A count vector of graphlets rooted at a given node)

- Degree counts that a Node touches Edges
- Clustering counts that a Node touches Triangles
- GDV counts that a Node touches Graphlets





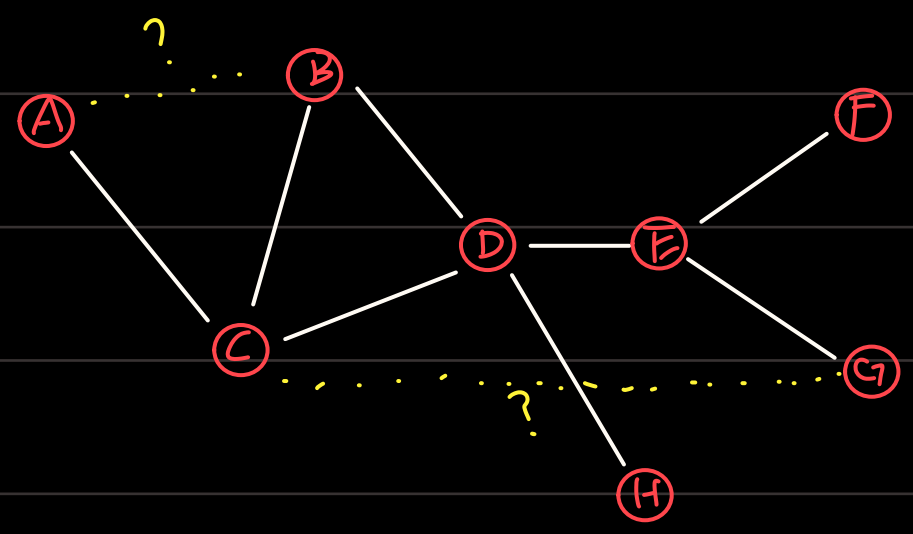
GDV of Node v on 2 to 3 nodes: (a, b, c, d) → (2, 1, 0, 2)

If on 2 to 5 → $GDV \in V^{1\times 13}$

GDV provide a measure of a Node's Local Network Topology

Comparing GDV of 2 Nodes provides a more detailed measure of local topological similarity than Node Degrees or Clustering Coefficient

Link-Level Features

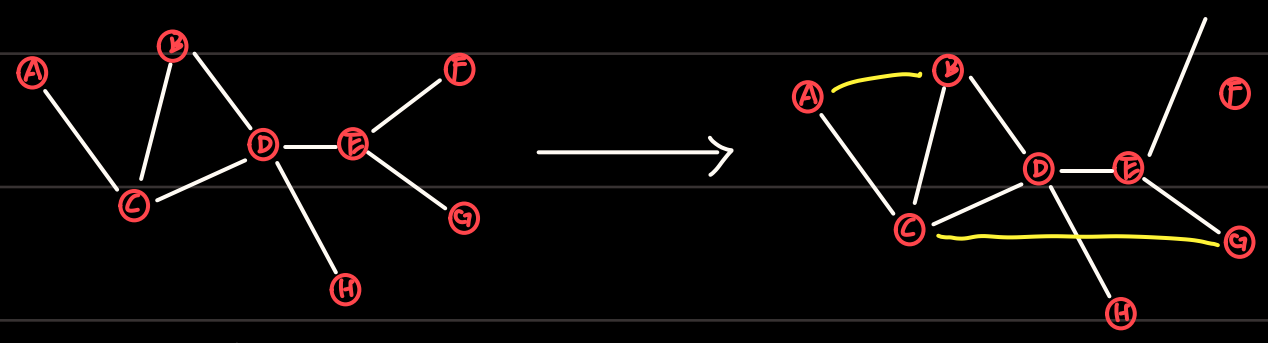


Key: Design features for pairs of nodes

Formulations for link prediction:

① Random missing links: Remove links randomly and try to predict them

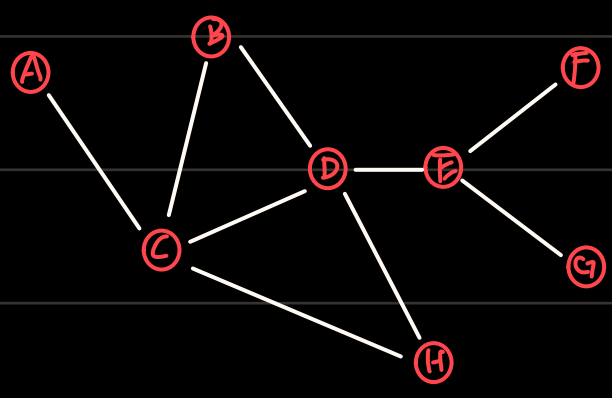
② links over time:



$G[t_0, t_0']$ $G[t_1, t_1']$

① Distance-Based Feature

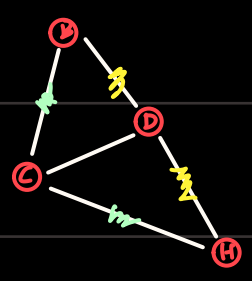
shortest distance from Node1 to Node2



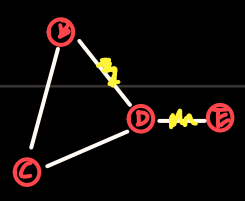
$S_{BH} = S_{BE} = S_{AB} = 2$

$S_{BG} = S_{BF} = 3$

Weak: not capture the degree of neighborhood overlap



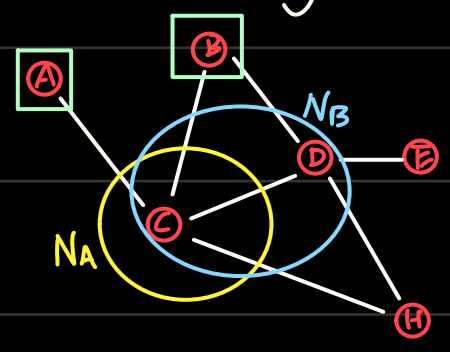
S_{BH}: 2 paths



S_{BE}: 1 paths

② Neighborhood Overlaps Feature

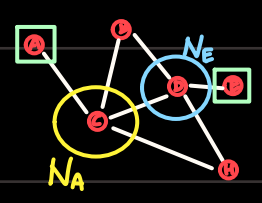
Def Local Neighborhood Feature



Capture num of neighboring nodes shared between nodes v_1 and v_2

- Common Neighbors: $|N(A) \cap N(B)| = |\{C\}| = 1$
- Jaccard's coeffcient: $\frac{|N(A) \cap N(B)|}{|N(A) \cup N(B)|} = \frac{|\{C\}|}{|\{C,D\}|} = \frac{1}{2}$
- Adamic-Adar Index: $\sum_{M \in N(A) \cap N(B)} \frac{1}{\log(k_M)} = \frac{1}{\log(k_C)} = \frac{1}{\log 4}$

limitation →



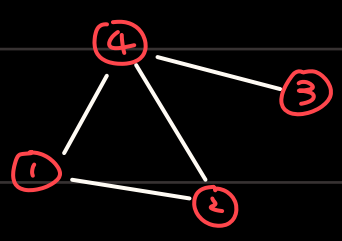
$N_A \cap N_E = \emptyset$

$|N_A \cap N_E| = 0$

Def Global Neighborhood Feature (katz Index)

katz Index: Count the num of the path of all length between a given pair of nodes

(Use Power Of Adjucency Matrix to Compute num of paths between 2 nodes)



$P_{12}^{(1)} = A_{12}$

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

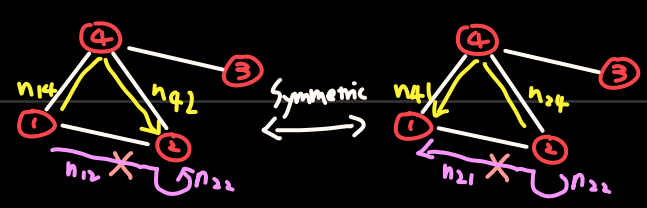
$A_{uv} = 1$ if $u \in N(v)$

Let $P_{uv}^{(k)}$ denote num of path length is k between node u and v

Property: $P^{(k)} = A^k$

$$A^2 = A \cdot A^T = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 3 \end{pmatrix}$$

$P_{12}^{(2)} = A_{12}^2$



katz Index between node v_1 and v_2:

$$S_{v_1 v_2} = \sum_{l=1}^{\infty} \beta^l A_{v_1 v_2}^l, \quad \beta = \text{discount factor} \in (0, 1)$$

↓ closed form

$$S = (I - \beta A)^{-1} - I$$

Proof $\sum_{l=1}^{\infty} \beta^l A^l = (I - \beta A)^{-1} - I$

$S = \beta A + \beta^2 A^2 + \beta^3 A^3 \cdots$

$(I - \beta A)(I + S)$

$= (I - \beta A)(I + \beta A + \beta^2 A^2 + \beta^3 A^3 + \cdots)$

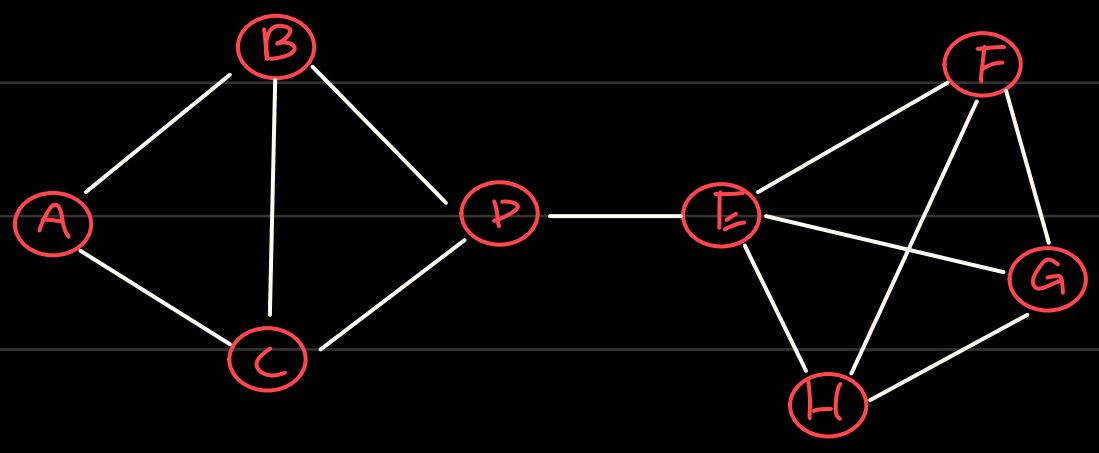
$= (I + \beta A + \beta^2 A^2 + \ldots) - (\beta A + \beta^2 A^2 + \cdots)$

$= I$

$\rightarrow I + S = (I - \beta A)^{-1}$

$\rightarrow S = (I - \beta A)^{-1} - I$

Graph Level Feature



How do we describe entire Graph Feature

Def Kernels (e.g. kernel SVM · kernel PCA)

(from low dimension to approximate high dimension similarity)

$x \in \mathbb{R}^{n \times d}$, $\phi(x) \in \mathbb{R}^{n \times n}$, $n > d$, $\phi(x)^T \phi(x) \approx x'x = \sum_i \sum_j x_i \bar{x}_j = \sum_i \sum_j k(x_i, x_j)$

G : Graph

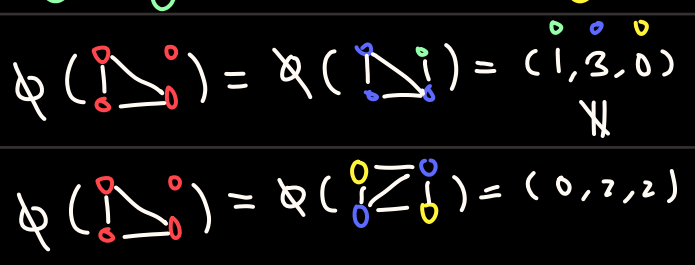
Kernel : $K(G, G') \in \mathbb{R}$ measure similarity of 2 graphs

Kernel Matrix : $K = (k(G, G'))_{G,G'}$, pos-semi define : ① pos eigenvalue ② symmetric

$\exists \phi$ (feature Rep) s.t $K(G, G') = \phi(G)^T \phi(G')$

Key Design : Bag of words ⟶ Bag of node degrees

O : 1 Degree O : 2 Degree O : 3 Degree

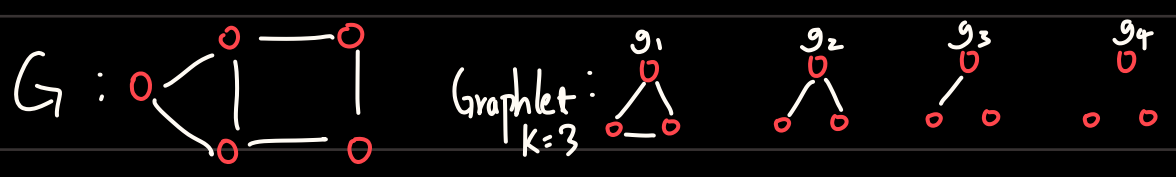


Def Graphlet Feature (Not Node-Level Graphlet Feature)

$g_k = (g_1, \ldots, g_{n_k})$: a list of graphlets of size k

Given a graph G, graphlet list g_k, define the graphlet count $f_G \in \mathbb{R}^{n_k}$

$(f_G)_i = \text{count}(g_i \subseteq G)$ $\forall i = 1, \ldots, n_k$



$f_G = (1, 3, 6, 0)$

Graphlet Kernel: $k(G, G') = f_G^T f_{G'}$

prevent skew → normalize: $h_G = \frac{f_G}{\sum f_G^{(i)}}$, $k(G, G') = h_G^T h_{G'}$

limitation: Computation Expensive (NP-Hard, $O(nd^{k-1})$ graph's node degrees bdd by d)

Def Weifeiler-Lehman Kernel

Color Refinement Algorithm: use neighborhood structure to iteratively enrich node vocabulary

Given a graph G with a set of nodes V

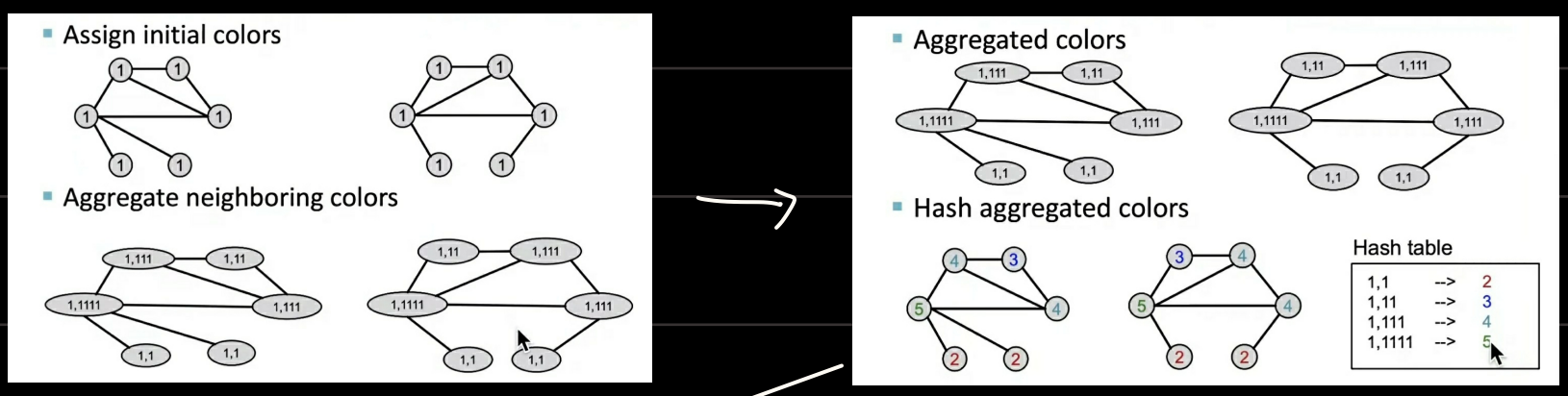
Init color $c^{(0)}(v)$ to each node v in V

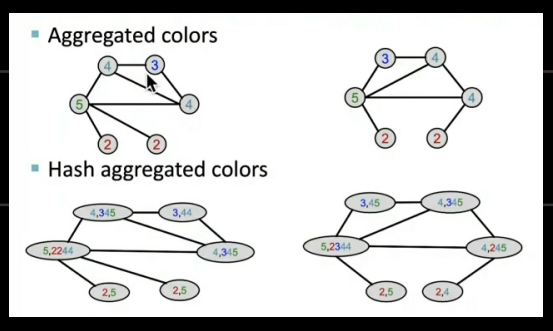
Iterate: $c^{(k+1)}(v) = \text{HASH}(\{c^{(k)}(v), \{c^{(k)}(u)\}_{u \in N(v)}\})$

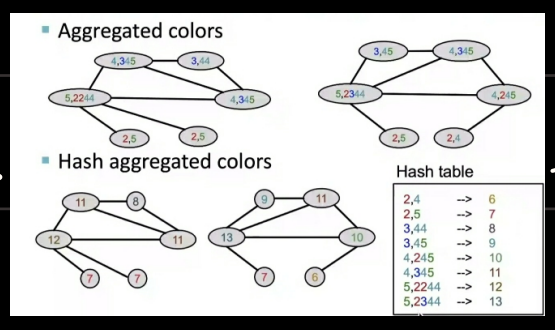
HASH: Map: V → color; $c^{(k)}(v)$: node v's color; $\{c^{(k)}(u)\}$: v's neighbors' colors

K steps → $c^{(K)}(v)$ summerizes the structure of K-hop neighborhood

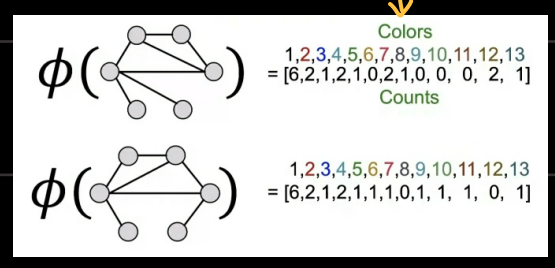
Init

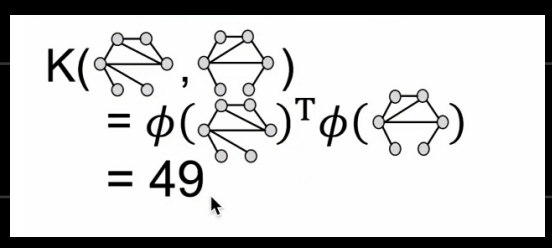






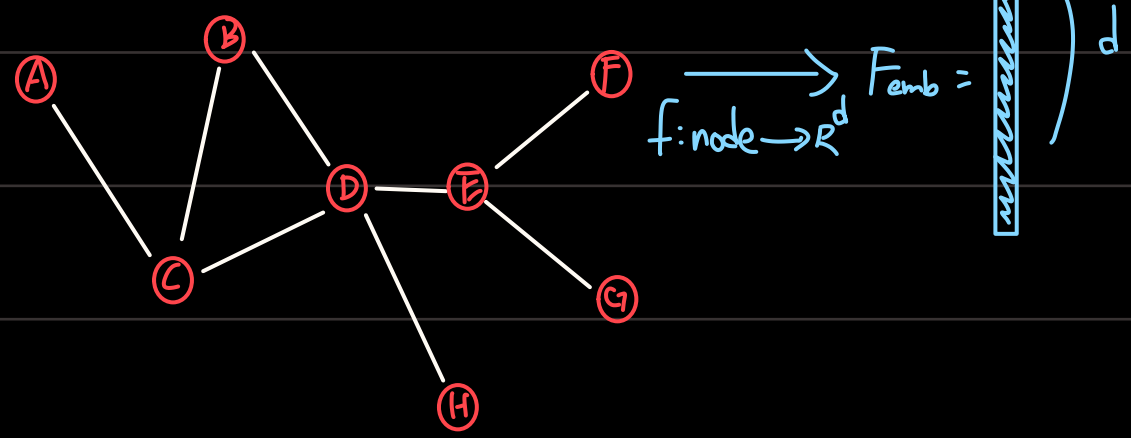
the count of of the color appeared in all iter step



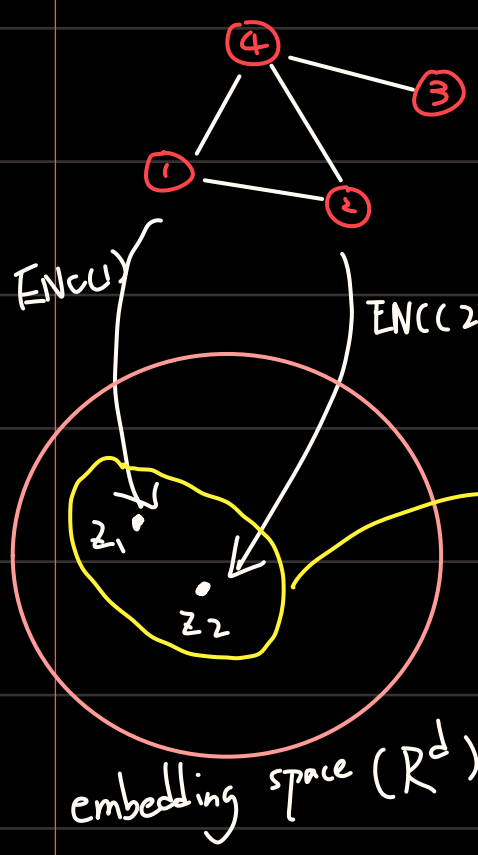


Node Embedding

Motivation



Key Point



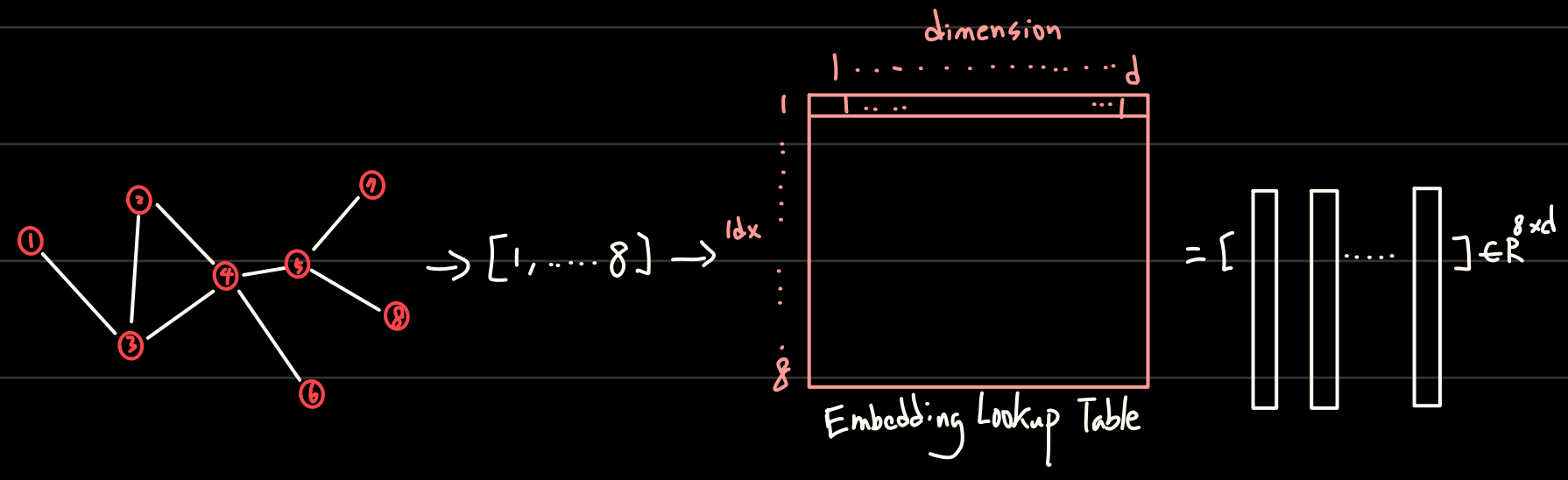
$V = \{1, 2, 3, 4\}$

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

the similarity in embedding space $\approx$ the similarity in the graph
(e.g. cosine similarity) (e.g. link distance)

$\text{Sim}(①, ②) \approx z_1^T z_2$

(Def) Shallow Encoding



How to learn embedding lookup table → DeepWalk, Node2Vec

Def Random Walk for Node Embeddings

Notation:

Node u → table → emb z_u

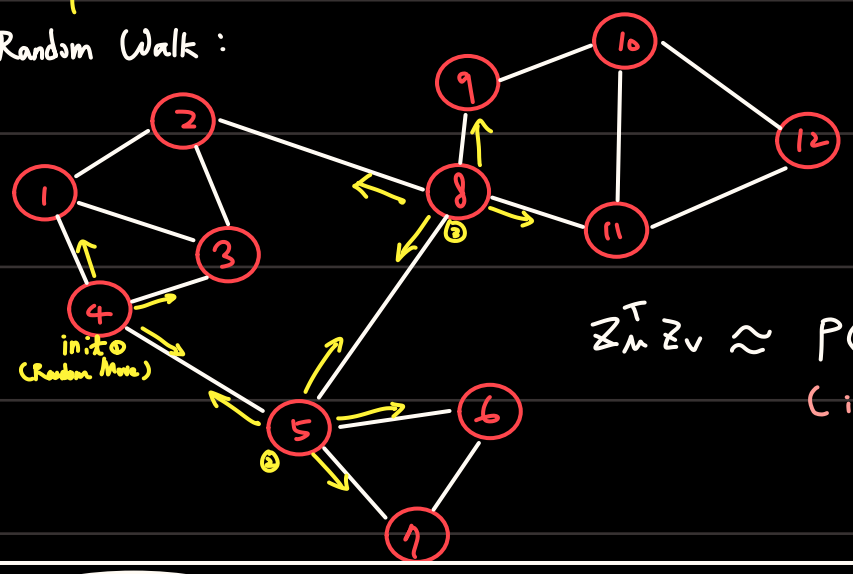
$P(v|z_u)$ = Prob of node u visit node v by Random Walk

① softmax: $z = \begin{bmatrix} \\ \\ \end{bmatrix} k$, $\sigma(z)_i = \frac{e^{z_i}}{\sum_j^k \exp(z_j)}$, $\sigma(z) = \begin{bmatrix} \\ \\ \end{bmatrix} k \in [0,1]_{1 \times K}$

② Sigmoid: $S(x) = \frac{1}{1+e^{-x}}$

$N_R(u)$: Given node u, neighborhood of u obtained by some Random Walk Strategy R (Sequence of nodes)

Random Walk:



$z_u^T z_v \approx$ P(u and v co-occur on a random walk over the graph)

(if u, v close: they have high frequency visiting each others)

Optimization

Given graph $G: G(V, E)$ (Vertex: All Nodes, # links), learn $f: u \to R^d$ denoted by $f(u) = z_u$

Optimize z_v, $\theta \to \cos(\theta) \propto P_R(v|u)$, z_u from Random Walk Statistics where R is Random Walk Strategy

objective: $\max_f \sum_{u \in V} \log P(N_R(u) | z_u)$

Algorithm:

① Starting from each node u in graph $N_R(u)$: ⇒ [n1, n1, n2, n1, ...]

② $\forall N_R(u)$: optimize embeddings from → Given u, predict its neighbors $N_R(u)$

and $\max_f \sum_{u \in V} \log P(N_R(u) | z_u) \iff \min \sum_{u \in V} \sum_{v \in N_R(u)} -\log(P(v|z_u))$, where $P(v|z_u) = \frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)}$

(minimize cross entropy: $\sum_{u \in V} E_{v \in V} [-\log P(v|z_u)]$)

$\sum_V \sum_{N_R(u)} \frac{\cdots}{\sum_V}$ too expensive: Instead of normalizing w.r.t all nodes, normalize against k Negative Sample n_i

$\log \frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)} \approx \log(\sigma(z_u^T z_v)) - \sum_{i=1}^k \log(\sigma(z_u^T z_{n_i}))$, $n_i \sim P_V$ (Random Distribution over nodes)

Negative Sampling for n_i: Sample k negative nodes each with prob proportional to its degree (k = 5~20)

minimize Obj by SGD ∇_z Loss

Part Of Recommend System

Graph Matrix · PageRank · RandomWalks · Embeddings

Google Algorithm : PageRank (Directed Graph)

Web → Graph

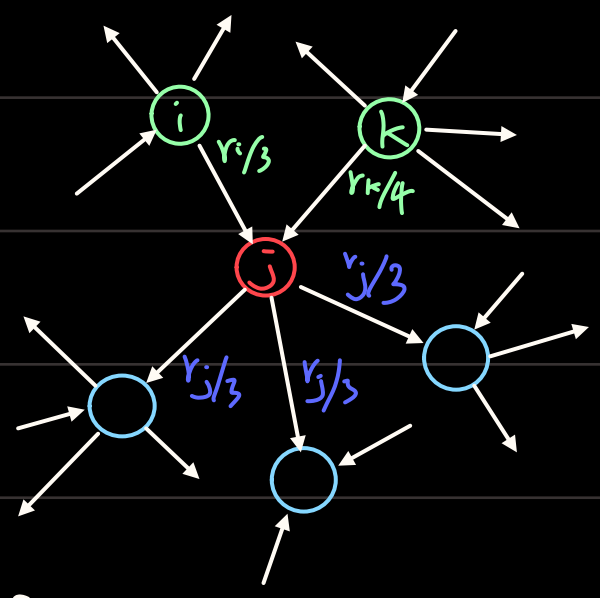
Pages → Nodes

Links(Edges) → Hyperlinks

Why Rank? → Web Pages are not equally important

Important pages have more links ↔ A vote from important page is worth more

Link As Node : Flow Model



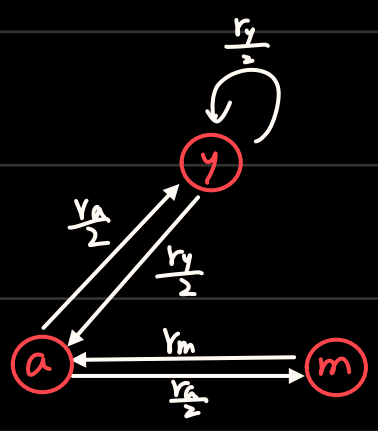
If page i with importance r_i has $d_i = 3$ out-links, each link get $\frac{r_i}{d_i}$ votes

Page j's own importance r_j is the sum of the votes on its in-links : $r_j = \frac{r_i}{3} + \frac{r_k}{4}$

Out-links → votes and In-links → importance

Def rank r_j for node j

$$r_j = \sum_{\substack{i \to j \\ \text{in}}} \frac{r_i}{d_i}$$, where d_i is out-degree of node i

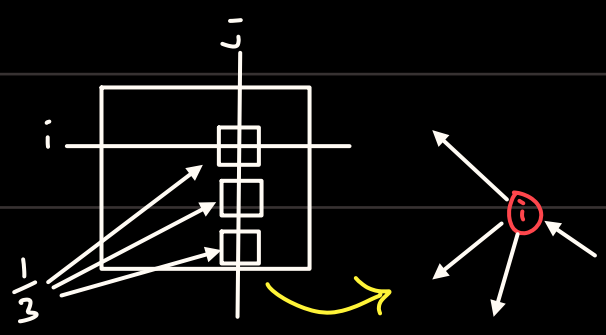


$$r_a = r_m + \frac{r_y}{2}$$
$$r_y = \frac{r_a}{2} + \frac{r_y}{2}$$
$$r_m = \frac{r_a}{2}$$

(Don't use Gaussian elimination to solve this system)

Stochastic Adjacency Matrix M (similar to transition matrix)

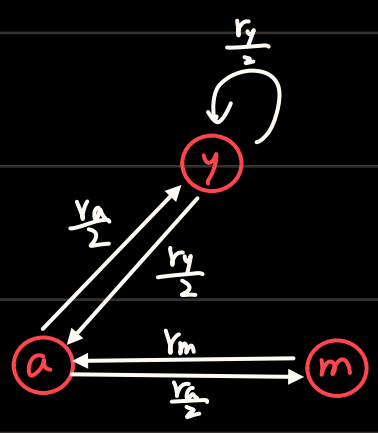
if $j \to i$ then $M_{ij} = \frac{1}{d_j}$, $\sum_i M_{ij} = 1$



→ only link-out index non-zero

Rank vector r: An entry per page, r_i is rank of page i where $\sum_i r_i = 1$

Flow Equation: $r = M \cdot r \longleftrightarrow r_j = \sum_{i \to j} \frac{r_i}{d_i}$



$$r_a = r_m + \frac{r_y}{2}$$
$$r_y = \frac{r_a}{2} + \frac{r_y}{2}$$
$$r_m = \frac{r_a}{2}$$

$$M = \begin{array}{c} \\ r_y \\ r_a \\ r_m \end{array} \begin{array}{c} \begin{array}{ccc} r_y & r_a & r_m \end{array} \\ \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 1 \\ 0 & \frac{1}{2} & 0 \end{bmatrix} \end{array}$$

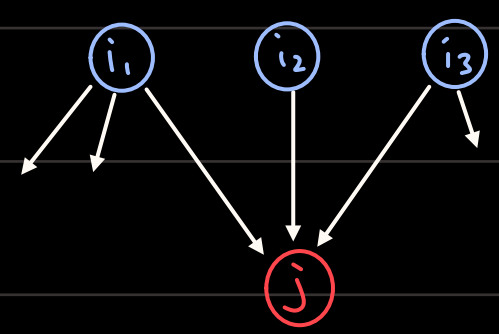
$$r = M \cdot r \to \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 1 \\ 0 & \frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix}$$

Def Connect to Random Walk: Random Web Surfer

{At any time t, surfer is on some page i

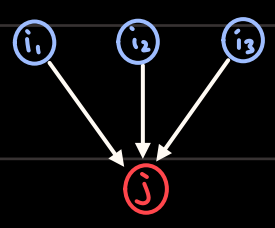
time $t+1$: the surfer follows an out-link from i uniformly at random

Ends up on some page j linked from i} Process repeats indefinitely



$p(t)$: a vector whose i^{th} coordinate is the prob that the surfer is at page i at time t

→ A probability distribution over pages



The Stationary distribution:

Follow a link uniformly at random: $p(t+1) = M\,p(t)$

Random Walk ⇒ If $p(t+1) = M \cdot p(t) = p(t)$ → $p(t)$ is a stationary distribution of a random walk

r satisfies $r = M \cdot r$ → r is stationary distribution for the random walk

(Eigenvalue Centrality: $\lambda c = Ac$, where A is Adjacency matrix)

$$1 \cdot r = M \cdot r \rightarrow 1 \cdot \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 1 \\ 0 & \frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix}$$

Stochastic Adjacency Matrix M Property

① $\sum_i M_{ij} = 1$ ② $\sum_j M_{ij} = 1$ ③ $\sum_i M_{ij} = \sum_j M_{ij} = 1$

$\max_\lambda |\lambda| = 1 \quad \forall \lambda \in \{\lambda \mid \lambda r = Mr\}$

Eigenvalue of M : 1

Eigenvector of M : $\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix}$

long-term distribution of surfer : $r = MM \ldots M u \Rightarrow$ satisfies $1 \cdot r = M \cdot r$

→ PageRank = Limiting diseribution = principle eigenvector of M with eigenvalue(=1)

(asympotic) = the Stationary distribution of a ndom walk over the graph

Solve PageRank

Given $G(V, E)$, with n nodes

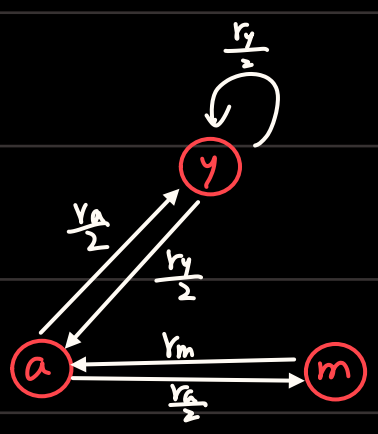
Init each node an initial pagerank

Repeat until achieve Converge condition($\sum_i |r_i^{t+1} - r_i^t| < \varepsilon$) $\{ r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i} \quad \forall_j \}$

Def Power Iteration

Init $r^{(0)} = \begin{pmatrix} \frac{1}{N} \\ \vdots \\ \frac{1}{N} \end{pmatrix}$

Repeat until $|r^{(t+1)} - r^{(t)}|_1 < \varepsilon$ $\{ r^{(t+1)} = M r^{(t)} \}$



$$r_a = r_m + \frac{r_y}{2}$$
$$r_y = \frac{r_a}{2} + \frac{r_y}{2}$$
$$r_m = \frac{r_a}{2}$$

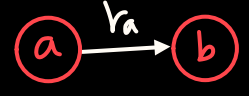
$$M = \begin{array}{c} \\ r_y \\ r_a \\ r_m \end{array} \begin{array}{c} \begin{array}{ccc} r_y & r_a & r_m \end{array} \\ \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 1 \\ 0 & \frac{1}{2} & 0 \end{bmatrix} \end{array}$$

$r^0 \xrightarrow{r' = Mr^0} r^1 \quad \ldots \quad r^T$

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}, \begin{bmatrix} \frac{1}{3} \\ \frac{3}{6} \\ \frac{1}{6} \end{bmatrix} \ldots \begin{bmatrix} \frac{6}{15} \\ \frac{6}{15} \\ \frac{3}{15} \end{bmatrix}$$

Problems:

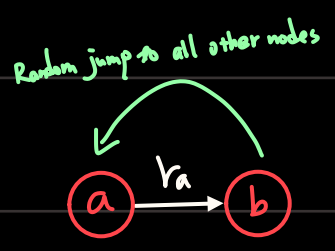
① some pages are dead ends (no out-links)



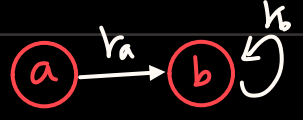
$r_a = 0 \quad r_b = r_a \quad M = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$

$\begin{bmatrix} r_a \\ r_b \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ (Random Walk Stop but $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$)

Solution →



② spider traps (all out-links are within the group)

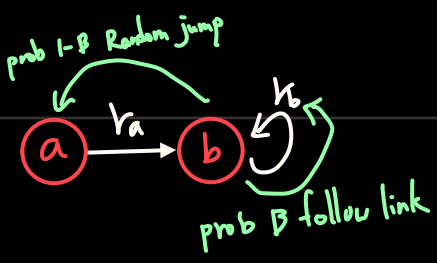


$r_a = 0 \quad r_b = r_a + r_b \quad M = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$

$\begin{bmatrix} r_a \\ r_b \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \ldots \begin{bmatrix} 0 \\ 1 \end{bmatrix} \ldots \infty$

(Random Walk keep running on the same node can't escape)

Solution →



PageRank Equation:

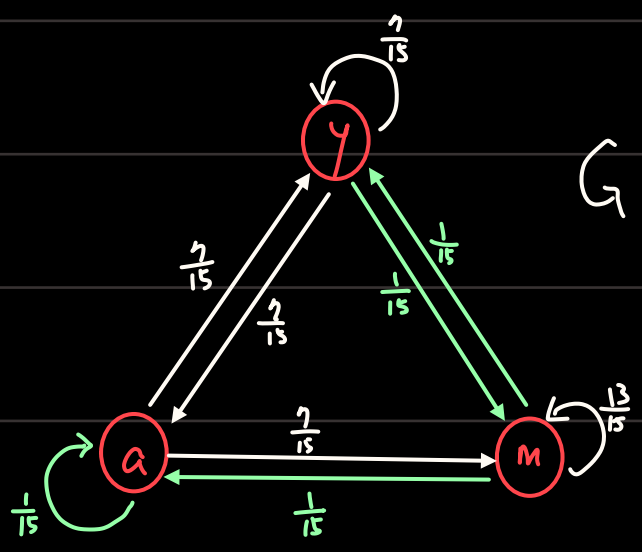
At each step, random surfer has 2 options:

① with prob β follow a link at random

② with prob $1-\beta$, jump to some random page

$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N} \quad \forall i, j$, where N is num of nodes in the graph (Expectation of link-in score)

$\Rightarrow 1 \cdot r = G \cdot r$, where $G = \beta M + (1-\beta)\left[\frac{1}{N}\right]_{N \times N}$ (β usually: 0.8, 0.9)



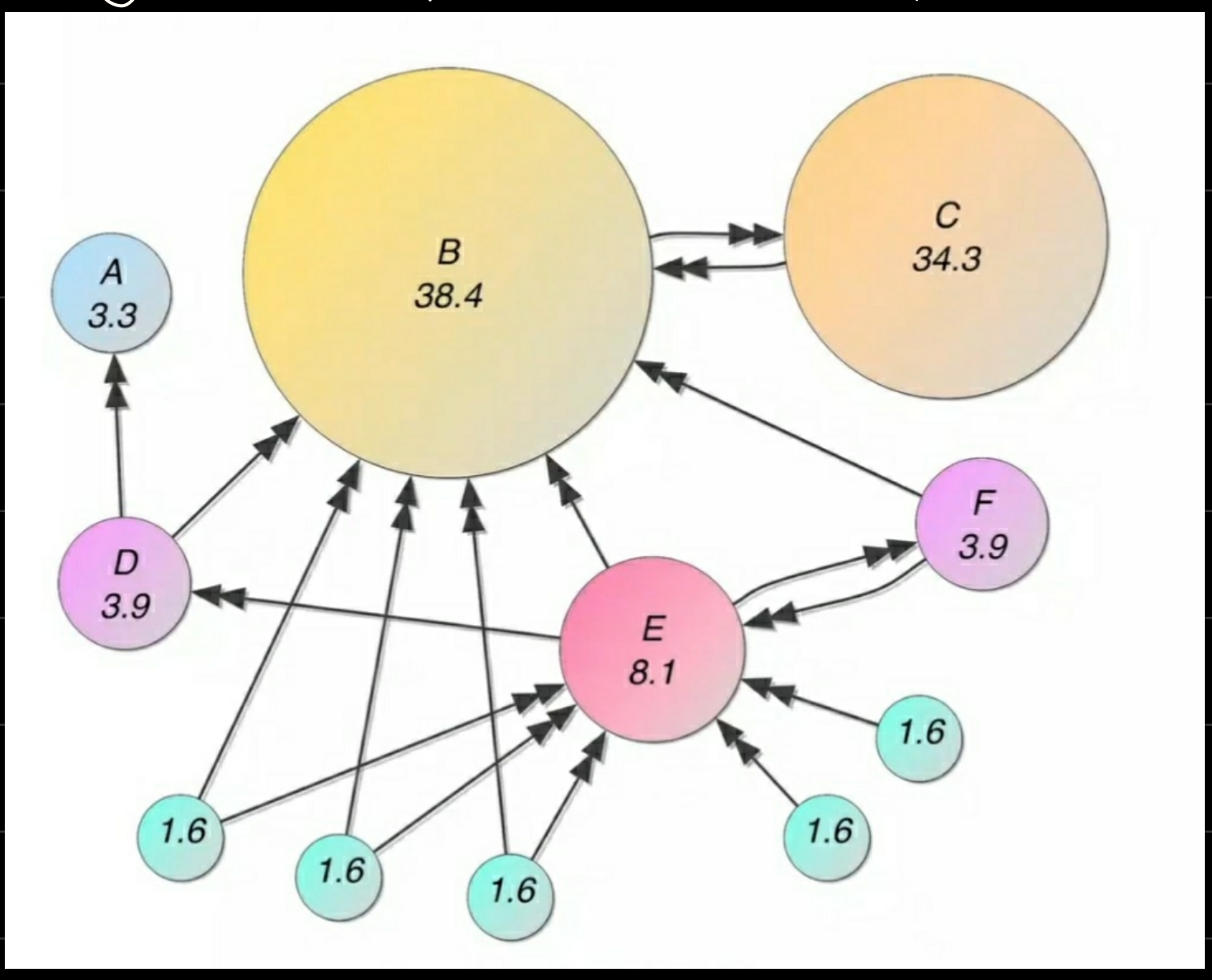
(Origin transition) (Multi-Arm Bandit Trick) (new transition)

$$G = 0.8\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 1 \end{bmatrix} + 0.2\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} = \begin{bmatrix} \frac{7}{15} & \frac{7}{15} & \frac{1}{15} \\ \frac{7}{15} & \frac{1}{15} & \frac{1}{15} \\ \frac{1}{15} & \frac{1}{15} & \frac{13}{15} \end{bmatrix}$$

$$r = \begin{bmatrix} y \\ a \\ m \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}, \begin{bmatrix} 0.33 \\ 0.2 \\ 0.46 \end{bmatrix} \cdots \cdots \begin{bmatrix} \frac{7}{33} \\ \frac{5}{33} \\ \frac{21}{33} \end{bmatrix}$$

$\rightarrow$ $r = G \cdot r$

Google : Agar.io and play $\rightarrow$ You'll better understand this topic



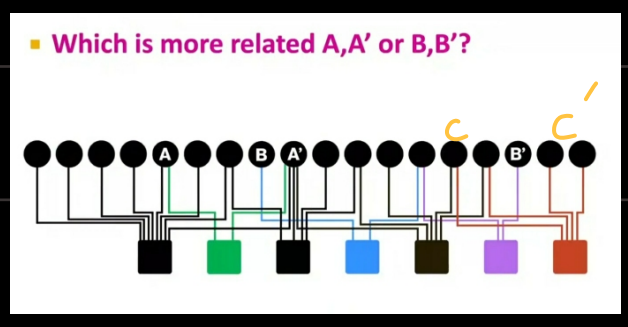
Random Walk with Restarts and Personalized PageRank

Intuition:

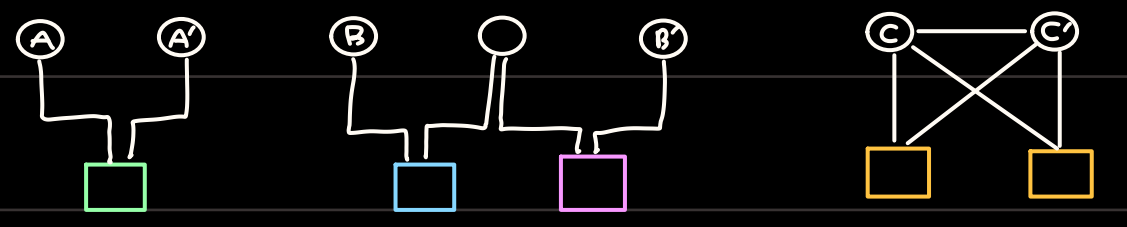
What items should we recommend to a user who interacts with Item Q ?

Proximity → If items Q and P are interacted by similar users, recommend P when user interacts with Q

Measurement (Bipartide):



Similarity? How to recommend?



shortest path

Common Neighbors

WTF which one ??

Def Proximity on graphs

Page Rank: ① Rank nodes by importance ② jump to any node with same prob

↓

Personalized PageRank: Ranks Proximity of nodes to jump nodes S
(Topic specific PageRank)

↓ What is most related to item Q?

Random Walks with Restarts: with prob β jump back to the starting node S={Q}
(Topic specific PageRank for 1 query)

Random Walk Simulation:

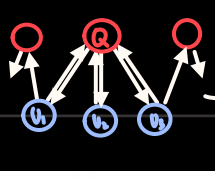
Every node has some importance → importance gets evenly split among all links and pushed to the neighbors

Given {query nodes} simulate a RandomWalk: $Q_1, Q_2 \ldots$ ① multi queries: personal ② single query: restart

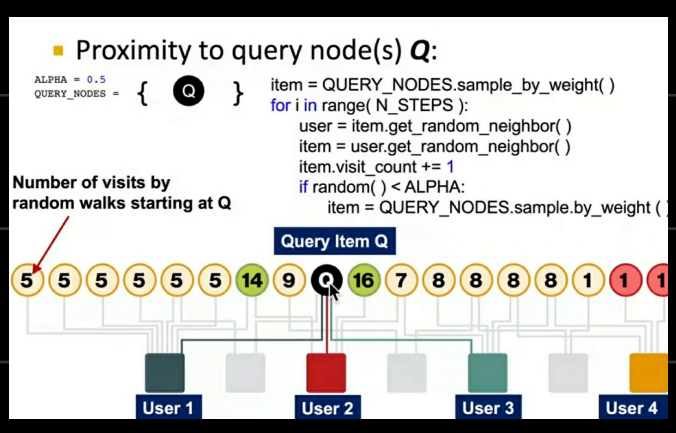
① Make a step to random neighbor and record the visit and visit count

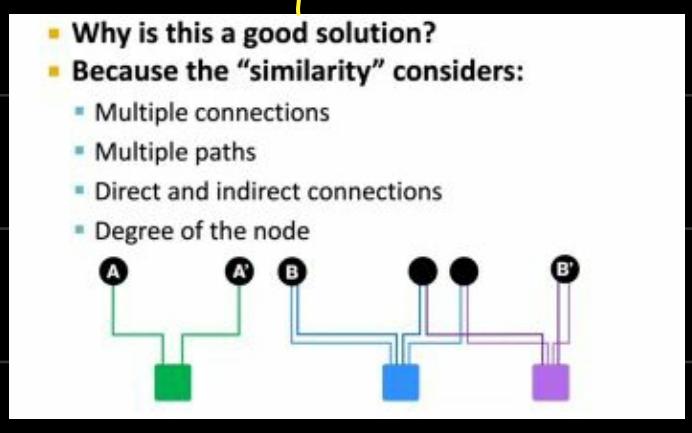
② with prob α → restart the walk at one of the query nodes

③ the nodes with the highest visit count have highest proximity to the queryd nodes

After Walking → 

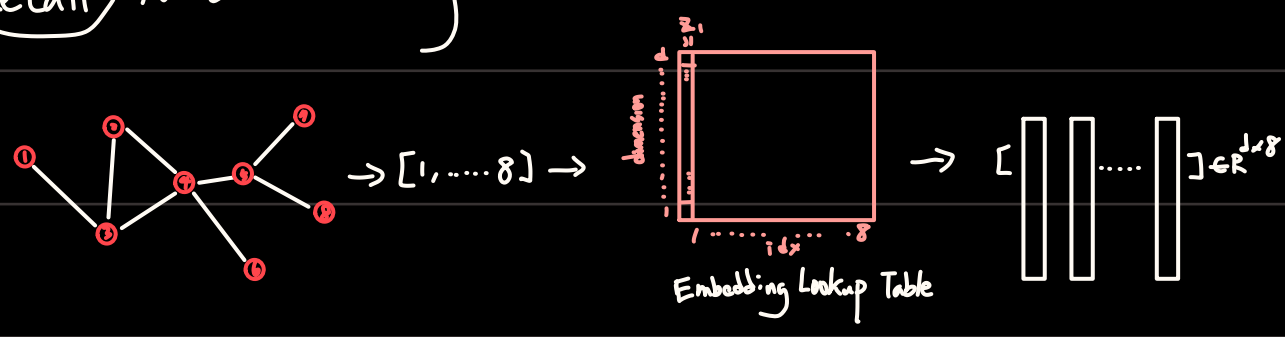
→ PageRank + Run Power Iteration → Pick highest rank item and recommand to user below Q (or Topk)





Matrix Factorization : Node Embedding

Recall Node embedding



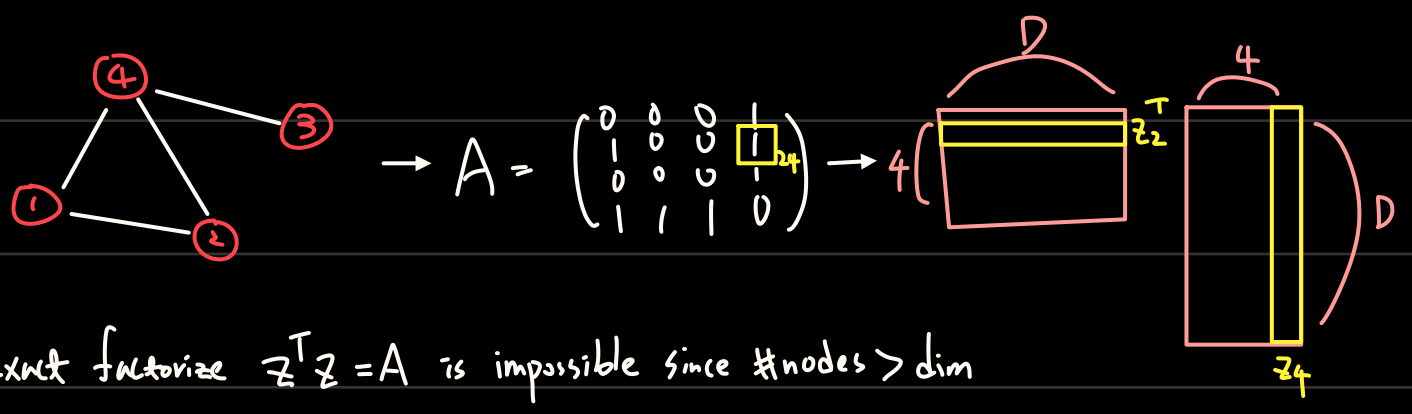
Objective: $\max z_v^T z_u$ for node pair (u, v) that are similar

Previous similar: node pairs appear many times when doing Randomwalk $\Rightarrow \max z_u^T z_v$

What if : u-v connected $z_v^T z_u = A_{uv} = 1 \longrightarrow Z^T Z = A$

Def Matrix Factorization

Key : edge connectivity $\approx$ Embedding dot-product similarity $\Rightarrow$ if connected : $z_u^T z_v \approx 1$
ow : $z_u^T z_v \approx 0$



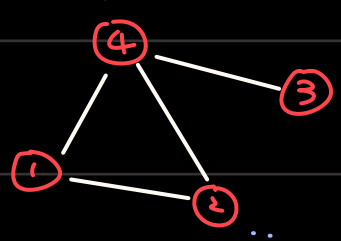
Exact factorize $Z^T Z = A$ is impossible since #nodes > dim

So we approach : $Z^T Z \approx A$ by optimization

Objective : $\min_Z \|A - Z^T Z\|_F^2$

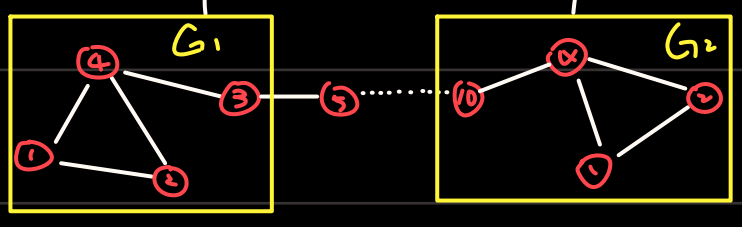
limitations

① can't compute new node embedding $\Rightarrow$ need to recompute whole embedding



(5) (new user in social network)

② cannot capture structural similarity (nodes in G_1, G_2 should be close in embedding space)



③ Can't incorperate node/link/graph level feature

Solution : Deep Representation Learning and Graph Neural Network