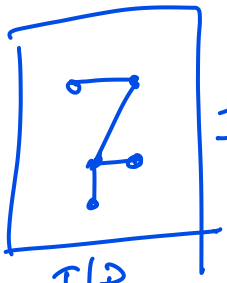
Graph is general language which describes/analyze relation among entities.

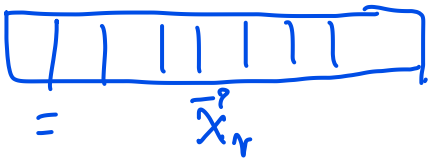
How to explore the relational structure for better prediction?

ML Pipeline



$\Rightarrow f: v \to \mathbb{R}^d$

→ similar node needs to have similar embedding.



$\bar{x}_v$

$y = Mx + C$

I/P G

f → Manually / Automatically

Nodes. Link. Graph. 3 kind of feature generation

Nodes → set of node

Link → set of link.

Feature generation is two step procedure.

① Design the feature

② Extraction of feature

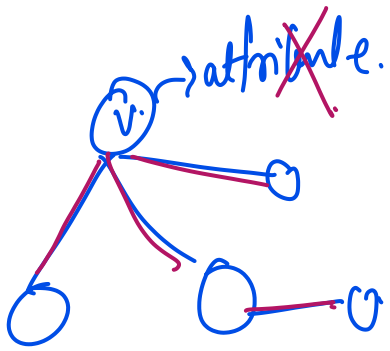
This can be done in two ways:

① Manually → Traditional ML of Graph.

② Automatically → Graph feature representation learning

Lets Manual part-

Node feature generation:



How to encode relational or Interaction / structure into feature set

Need to design a function $f: V \to R^d$ (z_{v_1}, z_{v_2})

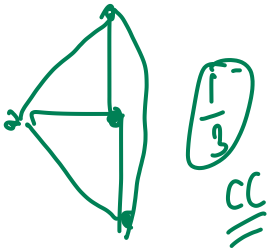
condition similarity $(v_1, v_2) = z_{v_1} \cdot z_{v_2}$

Design the function 'f':

Node feature design

- Importance based
  - Node degree
    - $f: v \to deg(v)$
  - centrality
    - Eigen C
    - Between C
    - closeness Cent.
      - Bringing two node closer

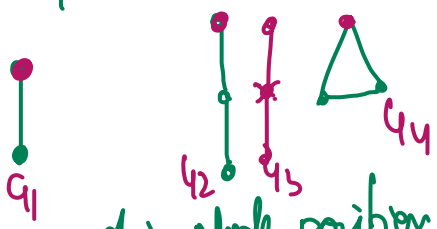
- structure base
  - deg.
  - clustering coeff.
  - Graphlets.



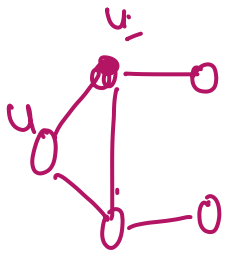
NCC: How well your neighbours are well connected

Graph local topological information is well explained by graphlet:
Graphlet for node and Graphlet for Graph is different.

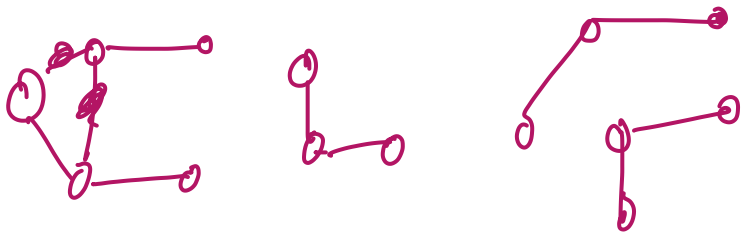
Node Graphlet:



Non-isomorphic Node position with 3 node GDV.



	G_1	G_2	G_3	G_4
X_u^d =	2	2	0	1
	3	1	2	1

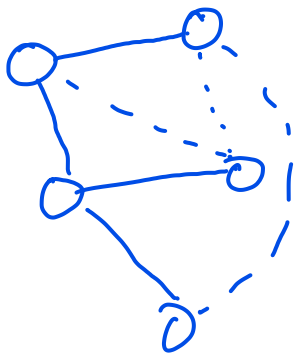


Node feature set Generation Summary:

① Importance base

② Structure based → Graphlet

Link prediction : we need a model which can predict link between nodes which is currently not existing in graph.



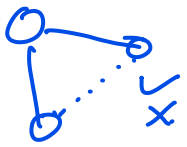
x_u x_v $M(x_u, x_v) = k_1$

x_u

Top k value will be taken where we can say link can exist.

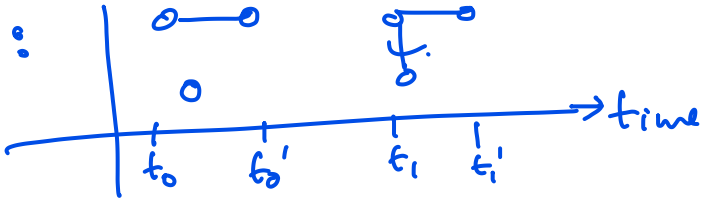
Two situation in which we need to predict link:

① static graph:



② Time evolving graph:

Time period

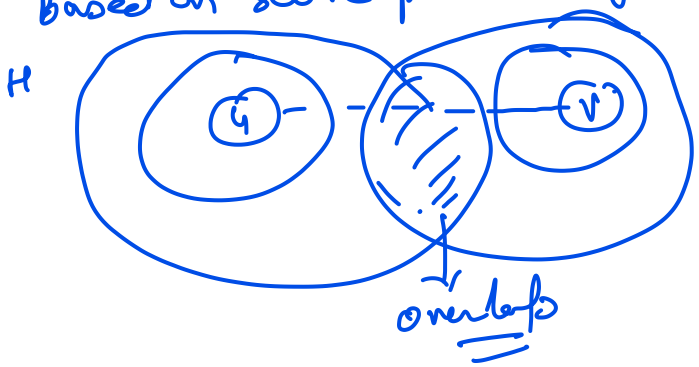


observe Predict.

General notion of link prediction: c can be neighbours overlaps.

Design heuristic function $c(x, y) = \text{score}$

Based on score predict if link will exist or not.



Link features design

① Distance based

② Local neighbourhood overlaps

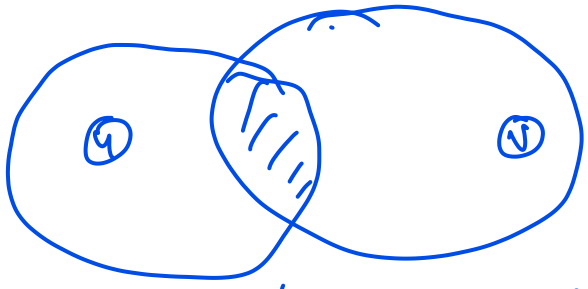
③ Global neighbourhood overlaps.

① Distance based: we take pair of node

$S(u, v)$ = k-hop distance

This way we encode the feature for link prediction but major disadvantage. This do not consider overlapping neighbours for their strength of connection.

② Local neighbourhood overlaps.



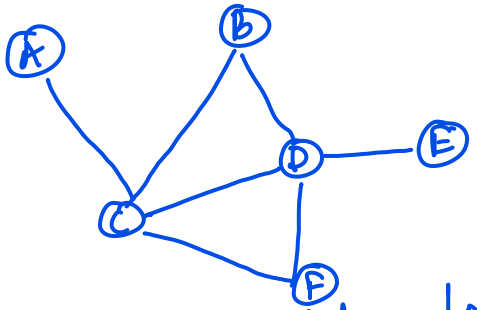
CN = $N(u) \cap N(v)$ ⇒ common neighbours.

Jaccard's coefficient = $\frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$

Adamic-Adar Index = $\sum_{c \in N(u) \cap N(v)} \frac{1}{\log(k_c)}$

k_c is degree of node

Example:



⇒ $N(A) = \{C\}$

$N(B) = \{C, D\}$

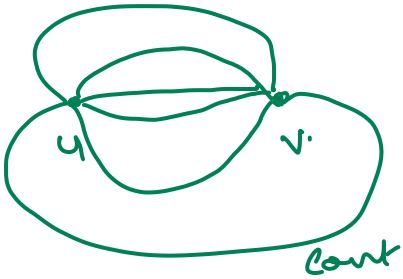
① Common neighbours overlaps = $|N(A) \cap N(B)| = |\{C\}| = 1$

② Jaccard coefficient = $\frac{|N(A) \cap N(B)|}{|N(A) \cup N(B)|} = \frac{1}{2}$

③ Adamic-Adar-Index = $\sum_{c \in N(A) \cap N(B)} \frac{1}{\log(k_c)} = \frac{1}{\log(4)}$

What is the issue in above: More than 2-hop away will have zero overlapping neighbours but strong chances that link will exist.

Global Network Neighbourhood overlaps:



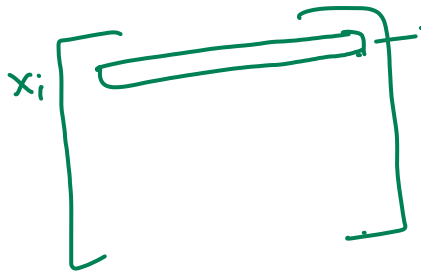
Many path in graph through which we can walk between ($u \neq v$)

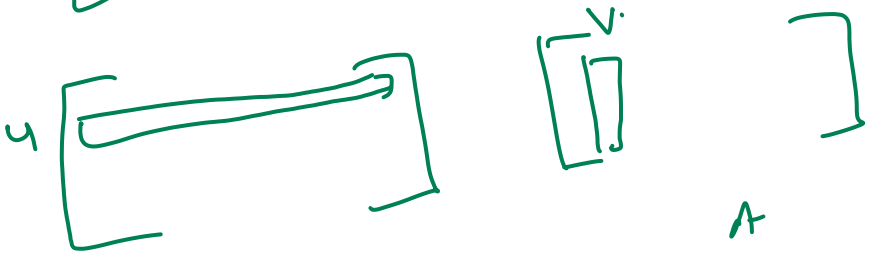
$u \rightarrow v$.

Count: All possible walk length between pair of nodes. ⇒ Kautz index.

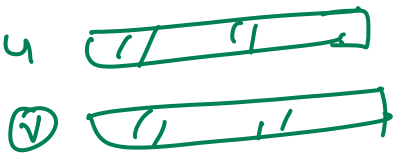
$$\sum_{l=0}^{\infty} K_l(u,v)$$

This can be computed using adjacency matrix.





Overlapping neighbour



Only overlapping neighbours will be added

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

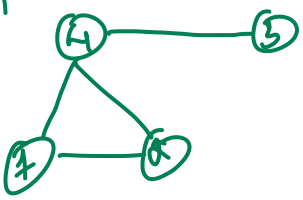
I is unit matrix.

$A \times A = A^2$

A^3

$A^4 \dots A^K$

Example:



$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

A^2_{22}

$$\begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

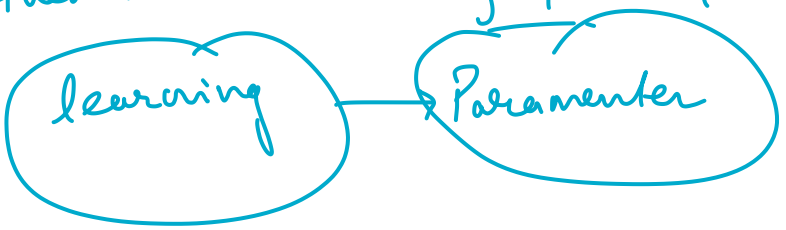
2

2-walk length

Compute 2-length walks between (1,2)

$$A^2_{12} = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = 1$$

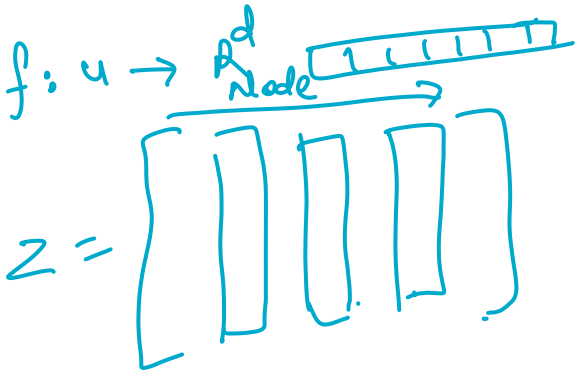
Automatic Embedding : when we learn embedding automatically then that is called graph representation learning.



→ Based on some objective function.

$$\text{Similarity}(x, y) \approx z_x \cdot z_y \Rightarrow$$

learning based on structural / Relation.
Not on attribute of node / link



Node embedding is lookup in Matrix Z so this is called shallow encoding.

How to learn parameter?

Framework:

① Encoder : we define function $f: \mathbb{R}^d$.

② Similarity function : similarity (u, v)

Criterion :

① Nodes are linked

② Overlapping neighbours.

③ have similar structural role

etc.

③ Decoder : Maps similarity score to embedding space

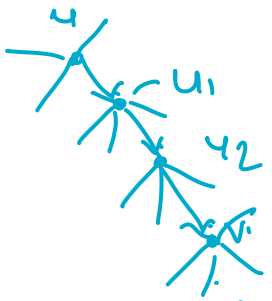
④ Optimization : $\text{similarity}(u, v) \approx z_u \cdot z_v$

Two method

① Random Walk method.

② node2vec

Random walk : Random walk can be many kind like uniform.



① In this we will initialize some R^d for node of our interest.
let Z_u = Initialize embading vector of size d.

② Let uniform distribution strategy adopted then. we can say.

$$P(v/Z_u) = P(u/Z_u) \times P(u_1/Z_u) \times P(u_2/Z_{u_1}) \times P(v/Z_{u_2})$$

we will take log likelihood

$$\log P(v/Z_u) = \max_f \sum_{u \in V} \log P(N_R(u)/Z_u)$$

$f: u \to R^d$

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log\left(P(N_R(v)/Z_u\right)$$

③ $Z_u^T \cdot Z_v$ is cosine similarity which will be maximum if u & v are co-occuring frequently in random walk.

④ $Z_u^T \cdot Z_v \Rightarrow$ Take softmax fun $= \dfrac{\exp(Z_u^T \cdot Z_v)}{\sum_{n \in V} (Z_u^T Z_n)} = P_R(v/Z_u)$

$$\mathcal{L} = -\sum_{u \in V} \sum_{v \in N_R(u)} \log\left[\frac{\exp(Z_u^T \cdot Z_v)}{\sum_{n \in V} \exp(Z_u^T Z_n)}\right]$$

Random walk optimization eqn.

V = set of all vertices
v = target vertex when started from u

$$P(v/z_u) = P(u/z_u) \times P(x/z_u) \times P(y/z_x) \times P(v/z_y)$$

log likelihood $\log\left[P(v/z_u)\right] = \sum_{u \in V}\left[\log P\left(N_R(u)/z_u\right)\right]$

$$\theta = \max_{f} \sum_{u \in V} \sum_{v \in N_R(u)} \log P(v/z_u)$$

$$\log\left(P(v/z_u)\right) = \log\left(\frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)}\right)$$

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log \frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)}$$

(Minimize objective funch.)

→ sigmoid (σ)
↘ K-sample

Negative sampling approach:

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log \frac{\sigma(z_u^T z_v)}{\sum_{n \in K} \sigma(z_u^T z_n)}$$

$\left(\min_{\theta} \mathcal{L}\right)$ → objective.
→ same as deeplearning. where we can adopt optimization strategy

$\frac{\partial L}{\partial \theta_1}, \frac{\partial L}{\partial \theta_2} \cdots \frac{\partial L}{\partial \theta_n}$

$z_u = z_u - \eta \frac{\partial L}{\partial z_u}$ η is learning rate

Random walks R.

Probability of reaching node v from u will be high if they are near. Nearby nodes will be similar.

$$\log\left(P(v/z_u)\right) = \sum_{u \in V} \log P\left(N_R(u)/z_u\right)$$

$z_u^T z_v \approx P(v/z_u)$

$$= \sum_{u \in V} \sum_{v \in N_R(u)} \log P(v/z_u)$$

$$\text{Softmax}(z_u \cdot z_v) = \frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)} = P(v/z_u)$$

$$\mathcal{L} = -\sum_{u \in V} \sum_{v \in N_R(u)} \log \frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)}$$

$\exp \to \sigma$

$n \in V \to n \in K.$

$$= -\sum_{u \in V} \sum_{v \in N_R(u)} \log \frac{(\sigma(z_u^T z_v))}{\sum_{n \in K} \sigma(z_u^T z_k)}$$

$\min_{z_u} \mathcal{L} \rightarrow$ gradient desc. $\quad K = 5 - 10$

$$z_u - \eta \frac{\partial L}{\partial z_u}$$

$-$ ↘

Node2Vec : Till now we have seen one random strategy R = uniform distribution.

$P(v/z_u)$ = Probability of reaching node v from u.

Assumption : similar node will have higher probability

$$z_u^T \cdot z_v \approx P(v/z_u)$$

log likelihood : $\log(P(v/z_u)) = \sum_{u \in V} \log P(N_R(u)/z_u)$

$$\max_f \left(\sum_{u \in V} \sum_{v \in N_R(u)} \log P(v/z_u) \right) \rightarrow \text{objective fun.}$$

$$\text{Softmax}(z_u^T z_v) = \frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)} = P(v/z_u)$$

Minimization objective fun.

$$-\sum_{u \in V} \sum_{v \in N_R(u)} \log \frac{\sigma(z_u^T z_v)}{\sum_{n \in K} \sigma(z_u^T z_n)} \qquad K = 5-10$$

high.

How we select R defines the performance of embedding.

In node2vec R will change.

Macroscopic Vs Microscopic View of Graphs

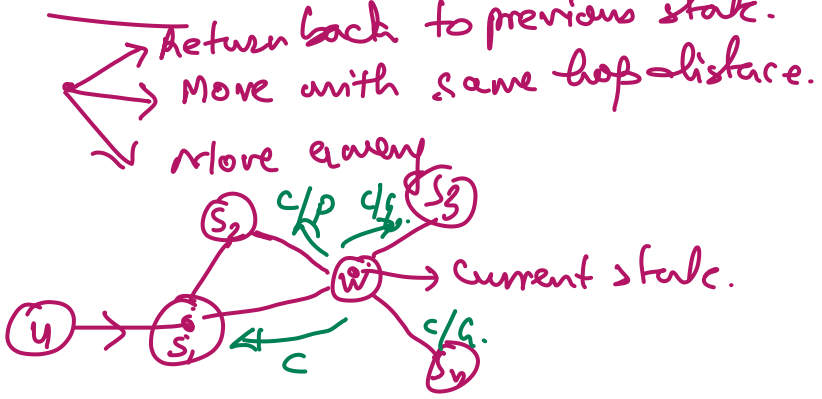
↳ Micr.

BFS will always look the nodes near to target node.

DFS will go away - Microscopic Macroscopic.

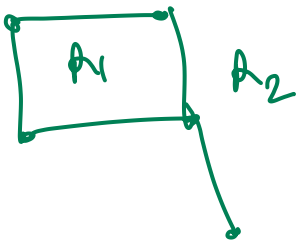
for any node we will create $N_R(u)$ based on R.

what is new R.



$N_R(u) = \{ \quad \}$ for deg 1

Graphs Region.



$2e \geq 3r \Rightarrow 3r \leq 2e$

$r = e - v + 2$

$r \leq \frac{2}{3}e$

$e - v + 2 \leq \frac{2}{3}e$

$e - \frac{2}{3}e \leq v - 2$

$\frac{3e - 2e}{3} \leq v - 2$

$e \leq 3v - 6$ ✓

$e \geq \frac{3}{2}r$

$e = r + v - 2$

$r + v - 2 \geq \frac{3}{2}r$

$v - 2 \geq \frac{3r - 2r}{2} \Rightarrow 2v - 4 \geq r$

Proper subset / subset -

$S = \{a, b, c\}$

$S \subseteq S$

$\{a, b\} \subseteq S$

$\{a, c\}$

$\{a\}$