Node 2 Vec:

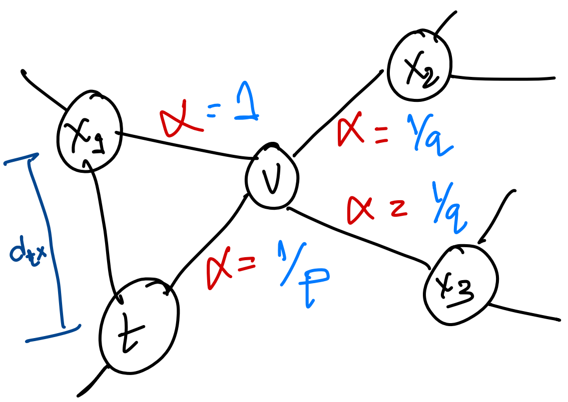
- Based on paper Word2Vec & uses the Skip-gram architecture described in the paper of Word2Vec

Why Node2Vec?

Traditional search strategies like BFS or DFS can explore neart neighbours or neighbour at depth. But if we have a hybrid algorithm to explore neighbours & also vertices at depths it reduces the computational cost & development time

$$f: G(v,e,w) \longrightarrow R^n$$

In node2Vec we generate Corpus of acyclic graph & feed it to Skip-gram model to obtain the results for down-stream tasks.



α = Search Bias

⇒ This helps to choose the next node to visit.

$P \rightarrow$ Prob. of walking back to <t> after visiting <v>

$q \rightarrow$ Prob to visit the next undiscovered parts.

For all source, $U \in V$ there exist $N_S(U) \subset V$

$N_S(U) \rightarrow$ Neighbourhood of 'U'

Log probability is chosen because adding is simpler than multiplication

$\Rightarrow$ We maximize the log probability of observing a network neighbourhood N_S for node 'u' on feature rep. 'f'

$$\boxed{\max_f \sum \log Pr(N_S(u) \mid f(u)) \quad - \ (1)}$$

For optimization we make the assumption of

a) Conditional independence

$$Pr(N_S(u) \mid f(u)) = \prod_{n_i \in N_S(u)} Pr(n_i \mid f(u)) \quad - \ (2)$$

(Markov assumption)

b) Symmetry in feature space

Using Softmax function:

$$Pr(n_i \mid f(u)) = \frac{e^{f(n_i).f(u)}}{\sum_{v \in V} e^{f(v).f(u)}} \quad - \ (3)$$

So using (2) & (3) Equ. (1) simplifies to

$$\max_f \sum_{u \in V} \log \left[\prod \frac{e^{f(n_i).f(u)}}{\sum_{v \in V} e^{f(v).f(u)}} \right]$$

$$\Rightarrow \max_f \sum \left[-\log \sum_{v \in V} e^{f(v).f(u)} + \sum_{n_i \in N_S(u)} f(n_i).f(u) \right]$$

from the above we obtained the maximization of features

Random Walks (again from Markov's concepts)

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

π_{vx} = transitional probability (unnormalized)

= how we move from one node to other.

$= \alpha_{pq}(t, x) \cdot w_{vx}$.

$$\begin{bmatrix} \pi_{v1} \\ \vdots \\ \pi_{vx} \end{bmatrix} = \begin{bmatrix} \alpha_{p1} \\ \vdots \\ \alpha_{pq} \end{bmatrix} \begin{bmatrix} w_{v1} \\ \vdots \\ w_{vx} \end{bmatrix} \rightarrow \text{these can be pre computed}$$

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

Negative Sampling

Soft max fn. in skip gram has the eq.

$$P = \frac{e^{f(n_i) \cdot f(u)}}{\sum_{i=1}^{V} e^{f(v) \cdot f(u)}} \in R^n$$

the problem here is this fn. is computationally expensive as we need to scan through all values of n_i

and the normilization factor in denominator.
requires V iterations to converge.
time Complexity $\approx O(V)$

So to overcome this we use negative Sampling
with SGD

learning rate

$$\theta^{(new)} = \theta^{(old)} - \eta \cdot \nabla_{J(\theta)}$$

→ Cost function

↳ Grad. of weight matrix

$$J(\theta; n_i) = - \sum_{-c \le j \le c, j \ne 0} \log p(n_{i+j} | n_i ; \theta)$$

So the parameter update equation of SGD
becomes
$$\theta^{(new)} = \theta^{(old)} - \eta \nabla_{J}(\theta, w^{(t)})$$