HW2

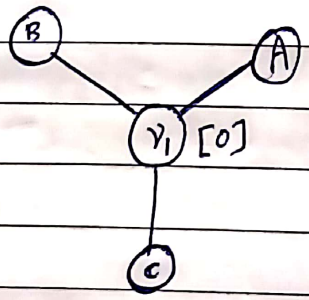
Q1 $h_v^{(0)} = x_v$

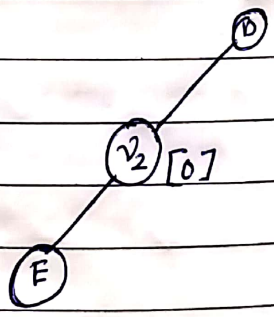
1.1 say $\phi(1) = A$
4 $\phi(5) = B$
& $\phi(6) = C$
& $\phi(7) = G$

then, $\phi(2) = D$
$\phi(3) = H$
&, $\phi(4) = E$
$\phi(8) = F$

one-to-one mapping ϕ

1.2 G_1 - G_2 -





s.t., $h_A^{(0)} = [20]$
$h_B^{(0)} = [5]$
$h_C^{(0)} = [+5]$

$h_D^{(0)} = [20]$
$h_E^{(0)} = [5]$

$(h_{v_1}^{(1)})_{max.\ aggregation} = [20] = (h_{v_2}^{(1)})_{max.\ aggregation}$

$(h_{v_1}^{(1)})_{mean\ aggregation} = \frac{1}{3}[30] = [10] = \frac{1}{2}[20] = (h_{v_2}^{(1)})_{mean\ aggregation}$

but,

$(h_{v_1}^{(1)})_{\text{sum aggregation}} = [30]$

$(h_{v_2}^{(1)})_{\text{sum}} \quad " \quad = [20]$

1.3 suppose, readout $(\{h_v^{(K)}, \forall v \in V_1\}) \neq$ readout $(\{h_v^{(K)}, \forall v \in V_2\})$

but the WL test says that the graphs are isomorphic. Here, aggregate & combine are concatenation & Hashing respectively.

$\because$ readout $(\{h_v^{(K)}, \forall v \in V_1\}) \neq$ readout $(\{h_v^{(K)}, \forall v \in V_2\})$

$\Rightarrow \exists\ v_1, v_2$ s.t. $v_1 \in V_1$ & $v_2 \in V_2$, $h_{v_1}^{(K)} \neq h_{v_2}^{(K)}$.

also $\because$ we use the same aggregate & combine f's,

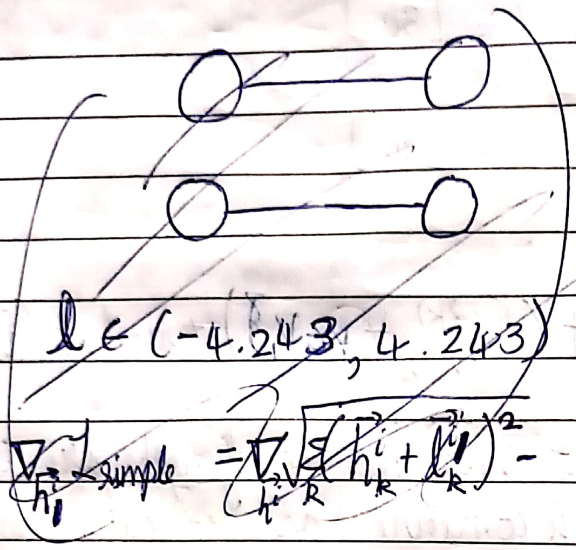
$$l_{v_1}^{(K)} \neq l_{v_2}^{(K)}$$

which contradicts our assumption of that the WL test says that the graphs are isomorphic.

$\Rightarrow$ if (readout ....) then (WL test ....).

(note this can be proved without using proof by contradiction)

Q2 2.1 $\mathcal{L}_{simple} = \sum_{(h,l,t) \in S} d(\vec{h}+\vec{l},\vec{t})$



$l \in (-4.243, 4.243)$

$\nabla_{h_i} \mathcal{L}_{simple} = \nabla_{h^i} \sqrt{\sum_k (h_k^i + l_k^i)^2}$

$\nabla_{\vec{h^i}} \mathcal{L}_{simple} = \nabla_{\vec{h^i}} \sqrt{\sum_k \{(h_k^i + l_k^i)^2 - (t_k^i)^2}$

$= \frac{2}{d(\vec{h^i}+\vec{l^i},\vec{t^i})} \times (\vec{h_i^i} + \vec{l^i})$

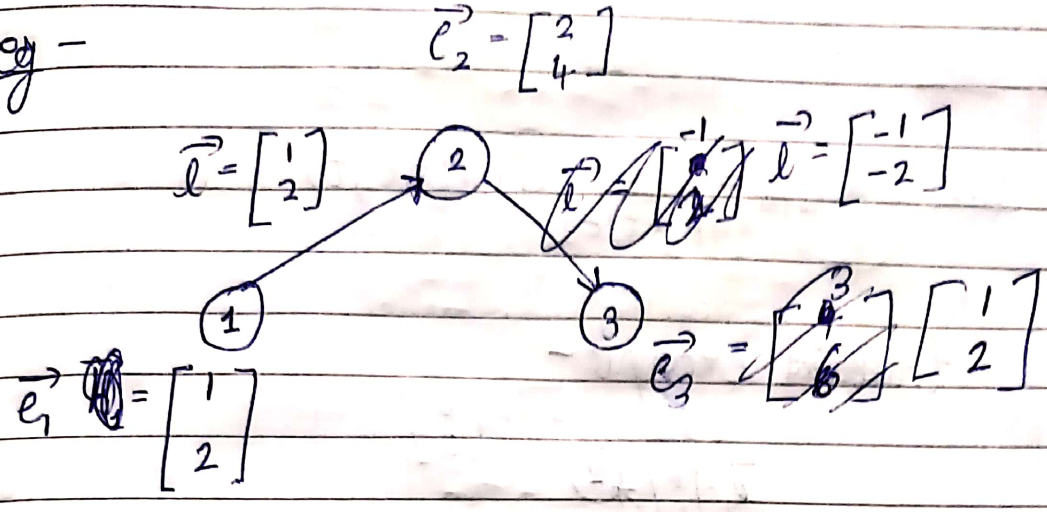
similarly,

$\nabla_{\vec{l^i}} \mathcal{L}_{simple} = \frac{2}{d(\vec{h^i}+\vec{l^i},\vec{t^i})} \times (\vec{h^i} + \vec{l^i})$

& $\nabla_{\vec{t^i}} \mathcal{L}_{simple} = \frac{-2}{d(\vec{h_i^i}+\vec{l^i},\vec{t^i})} \times (\vec{t^i})$

$\because \mathcal{L}_{simple}$ is to be minimized, update step -

eg -



this graph 4 embeddings make $L_{simple} = 0$ BUT these embeddings are useless ∵ all nodes have the same embeddings when normalized, no information is captured about the structure of the graph either.

☆ note that there is no convergence cond^n specified in the pseudocode, so if we loop only till Loss is zero then we might end up normalizing the $\vec{e}$'s but no updates to the embeddings take place & then all nodes have same embeddings.

2.1 $\mathcal{L}_{simple} = \sum_{(h,l,t) \in S} d(h+l, t)$

trivial example -

$\vec{t_1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$\vec{a_1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ (2)

(1) $\vec{h_1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

(3) $\vec{e_3} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

clearly, if we are given all embeddings -

$\vec{e_1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \vec{e_2} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \vec{e_3} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ & $\vec{l_1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

we would not be able to tell where the edge exists, between what nodes. Also, these nodes embeddings do not capture any info. about the graph structure, in fact all nodes do have the same embedding.

but this graph does make the loss 0.

slightly complex example -

$\vec{h_1} = \begin{bmatrix} 2 \\ 4 \end{bmatrix} = \vec{h_2}$

$\vec{l_1} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ (2) $\vec{l_2} = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$

$\vec{h_1} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ (1)

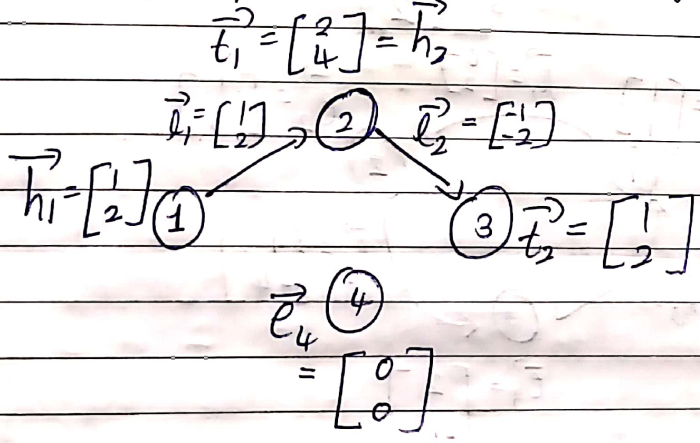
(3) $\vec{t_2} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

(4) $\vec{e_4} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

at $l=\vec{0}$ we will try to make $\vec{h}=\vec{t}$!

same problems as before but instead of all ~~node~~ nodes being 0 all are pointing in the same dir^n which is just as useless. This would also not protect against the corrupted pair (h_2, l_2, e_4).

2.2 same graph as before but diff. embeddings -

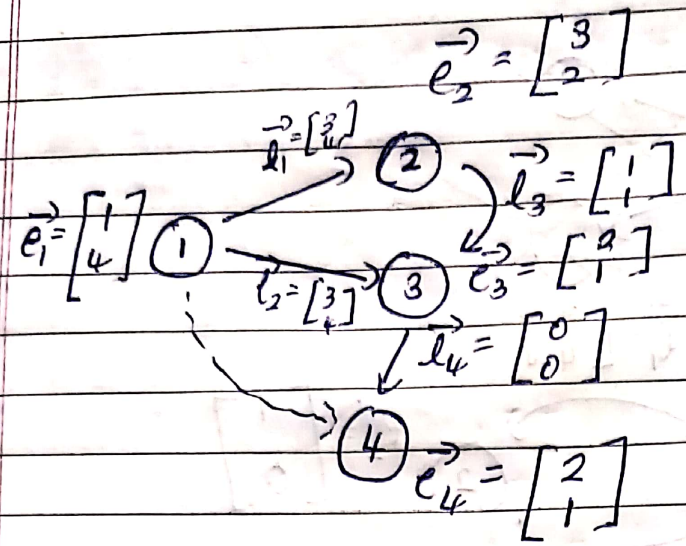


↳ no margin

satisfies all properties & drives the Loss to zero But given the embeddings we would not be able to decide if $\vec{l}_2$ is between ② & ③ or between ③ & ④! (There are many such edges)

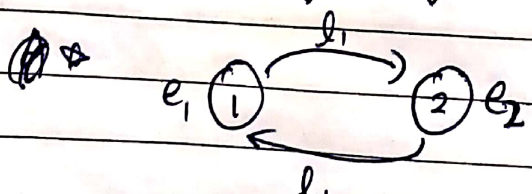
2.3 The algorithm could've increased or decreased h', l OR t' to satisfy the margin threshold; The generated embeddings would be completely out of scale & not at all indicative of the graph structure. We would also lose a ~~sim~~ notion of similarity in the nodes, i.e., two related nodes can have wide diff. in embedding eg. $\vec{h}_1(0.001, 0.001)$ $\vec{l}_1 = (1000, 1000)$ & $\vec{t}_1 = (1000.001, 1000.001)$ $\vec{h}_1 \cdot \vec{t}_1$ would be very small even though they have an edge in between.

2.4



for this graph, if $\vec{l_3} \neq [0\ 0]$, no embedding for edges & nodes can make the Loss 0; but setting $\vec{l_3} = [0\ 0]$ is equivalent to saying that 2 & 3 are the same entities which might not be true ⇒ TransE is bad for 1-to-N relations.

Also note that $\vec{l_4} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ does not imply ③ & ④ are the same entities, eg. ~~En a pro~~ 2 proteins may have very similar structure & functionality but be used by 2 ~~(comp)~~ different organs, in our graph & by TransE ① & ④ will always have an edge regardless.

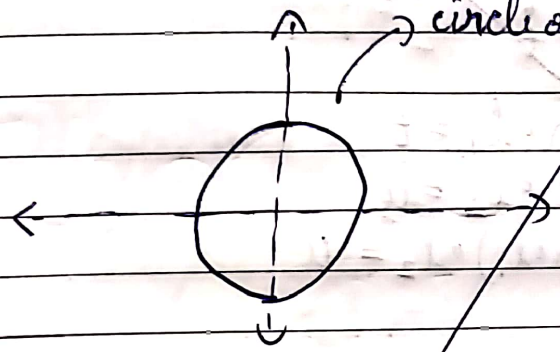


⇒ bad at symmetric relations

⇒ this graph can also not be modelled by Trans-E for any values of e_1, l_1 & e_2.

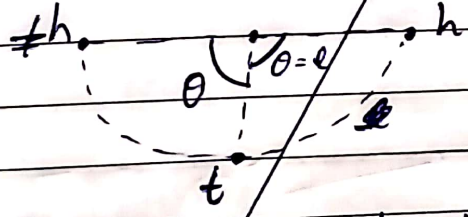
Q3 3.1 Except Symmetry, TransE can model Inverse & Composition relations (in lecture)

3.2 consider $d=1$ for illustration,



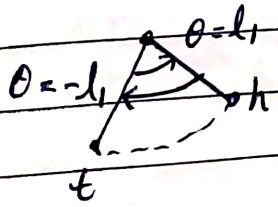
This model has the same problems as TransE :

1) cannot model Symmetric relations.



⇒ if $h \circ l = t$
then $t \circ l \neq h$
when $l \neq 2n\pi \quad n = 0, 1, \ldots$
at $l = 2n\pi$, h & t co-incide which is like TransE with $\vec{l} = 0$; ∴ if angles are a distinguishing factor then this might work but is not perfect.

2) models inverse relations

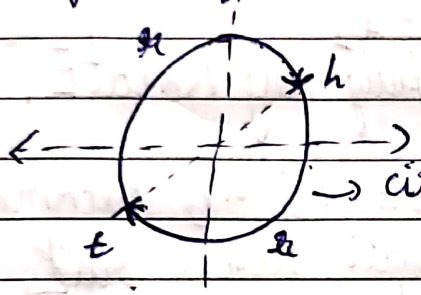


⇒ $h \circ l_1 = t$
& $t \circ l_2 = h \Rightarrow l_1 = -l_2$
can be modelled perfectly.

Q3 3.1 given in lecture slides, TransE can model all relations except symmetry among the 3 given.

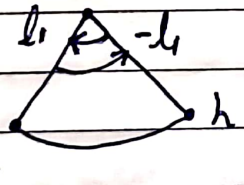
3.2 RotatE can model all 3 types perfectly.

1) Symmetry – (consider d=1 for illustration)



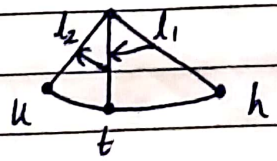
i.e. it will push h & t to $n\pi$ angle between them without $n=0$ & h & t becoming the same entities; the problem in TransE.

2) Inverse –
same as TransE, pick $l_1 = -l_2$



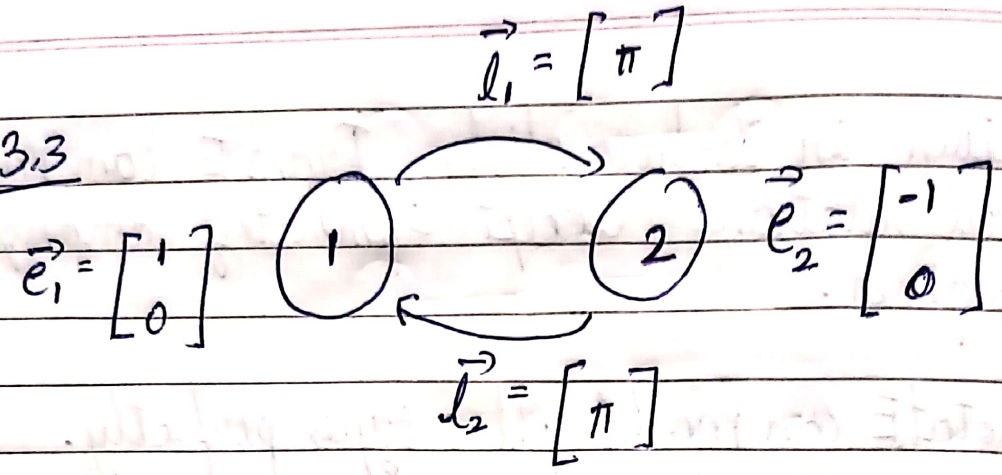
$h \circ l_1 = t$
& $t \circ (-l_1) = h$
$\Rightarrow t \circ (l_2) = h$ s.t. $l_2 = -l_1$

3) Composition –
same as TransE, $l = l_1 + l_2$



$h \circ l_1 = t$
$t \circ l_2 = u$
$\Rightarrow (h \circ l_1) \circ l_2 = u$

3.3



this is a symmetric relation
$\because \vec{l}_1 = \vec{l}_2$
A is modelled fully by RotatE.

but with TransE ($\vec{l}_1$ & $\vec{l}_2$ become 2D)
we cannot model such relations without making $\vec{l}_1 = \vec{l}_2 = \vec{0}$
which is not allowed.