11-741/11-441 – Machine Learning with Graphs **Due: April 25, 11:59PM**

Instructor: Yiming Yang

Homework 5: Knowledge Graph Embedding

In this assignment, you answer questions about several Knowledge Graph Embedding (KGE) algorithms.[1]

1 Honor Code [0 points]

(X) I have read and understood Honor Code before I submitted my work.

I have read and understood Honor Code before I submitted my work.

Collaboration: Write down the names & Andrew ID of students you collaborated with on Homework 2 (None if you didn't).

Name: Blessed Guda

Andrew ID: blessedg

Note: Read our website on our policy about collaboration!

2 Node Embeddings with TransE [20 points]

While many real world systems are effectively modeled as graphs, graphs can be a cumbersome format for certain downstream applications, such as machine learning models. It is often useful to represent each node of a graph as a vector in a continuous low dimensional space. The goal is to preserve information about the structure of the graph in the vectors assigned to each node. For instance, the spectral embedding preserved structure in the sense that nodes connected by an edge were usually close together in the (one-dimensional) embedding x.

Multi-relational graphs are graphs with multiple types of edges. They are incredibly useful for representing structured information, as in knowledge graphs. There may be one node representing "Washington, DC" and another representing "United States", and an edge between them with the type "Is capital of". In order to create an embedding for this type of graph, we need to capture information about not just which edges exist, but what the types of those edges are. In this problem, we will explore a particular algorithm designed to learn node embeddings for multi-relational graphs.

The algorithm we will look at is TransE.[2] We will first introduce some notation used in the paper describing this algorithm. We'll let a multi-relational graph $G = (E, S, L)$ consist of the set of *entities* E (i.e., nodes), a set of edges S, and a set of possible relationships L. The set S consists of triples (h, l, t), where $h \in E$ is the *head* or source-node, $l \in L$ is the relationship, and $t \in E$ is the *tail* or destination-node. As a node embedding, TransE tries to learn embeddings of each entity $e \in E$ into R^k (k-dimensional vectors), which we will notate by $\mathbf{e}$. The main innovation of TransE is that each relationship ℓ is also embedded as a vector $\ell \in \mathrm{R}^k$, such that the difference between the embeddings of entities linked via the relationship ℓ is approximately ℓ. That is, if $(h, \ell, t) \in S$, TransE tries to ensure that $\mathbf{h} + \ell \approx \mathbf{t}$. Simultanesouly, TransE tries to make sure that $\mathbf{h} + \ell \not\approx \mathbf{t}$ if the edge (h, ℓ, t) does not exist.

~~**Note on notation: we will use unbolded**~~ letters e, ℓ, etc. to denote the entities and relationships in the graph, and bold letters $\mathbf{e}, \boldsymbol{\ell}$, etc., to denote their corresponding embeddings. TransE accomplishes

by minimizing the following loss:

$$L = \sum_{(h,\ell,t)\in S} \sum_{(h',\ell,t')\in S'_{(h,\ell,t)}} [\gamma + d(\mathbf{h}+\boldsymbol{\ell},\mathbf{t}) - d(\mathbf{h}'+\boldsymbol{\ell},\mathbf{t}')]_+ \quad (1)$$

Here (h', ℓ, t') are "corrupted" triplets, chosen from the set $S'_{(h,\ell,t)}$ of corruptions of (h, ℓ, t), which are all triples where either h or t (but not both) is replaced by a random entity, and ℓ remains the same as the one in the original triplets.

$$S'_{(h,\ell,t)} = \{(h', \ell, t) \mid h' \in E\} \cup \{(h, \ell, t') \mid t' \in E\}$$

Additionally, $\gamma > 0$ is a fixed scalar called the *margin*, the function $d(\cdot, \cdot)$ is the Euclidean distance, and $[\cdot]_+$ is the positive part function (defined as $\max(0, \cdot)$). Finally, TransE restricts **all the entity embeddings to have length** 1 : $\|\mathbf{e}\|_2 = 1$ **for every** $e \in E$**.**
For reference, here is the TransE algorithm, as described in the original paper on page 3:

Algorithm 1 Learning TransE

input Training set $S = \{(h, \ell, t)\}$, entities and rel. sets E and L, margin γ, embeddings dim. k.

initialize $\boldsymbol{\ell} \leftarrow$ uniform$(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each $\ell \in L$
  $\boldsymbol{\ell} \leftarrow \boldsymbol{\ell} / \|\boldsymbol{\ell}\|$ for each $\ell \in L$
  $\mathbf{e} \leftarrow$ uniform$(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each entity $e \in E$
loop
  $\mathbf{e} \leftarrow \mathbf{e} / \|\mathbf{e}\|$ for each entity $e \in E$
  $S_{batch} \leftarrow$ sample(S, b) // sample a minibatch of size b
  $T_{batch} \leftarrow \emptyset$ // initialize the set of pairs of triplets
  for $(h, \ell, t) \in S_{batch}$ **do**
    $(h', \ell, t') \leftarrow$ sample$(S'_{(h,\ell,t)})$ // sample a corrupted triplet
    $T_{batch} \leftarrow T_{batch} \cup \{((h, \ell, t), (h', \ell, t'))\}$
  end for
  Update embeddings w.r.t. $\sum_{((h,\ell,t),(h',\ell,t'))\in T_{batch}} \nabla[\gamma + d(\boldsymbol{h}+\boldsymbol{\ell},\boldsymbol{t}) - d(\boldsymbol{h'}+\boldsymbol{\ell},\boldsymbol{t'})]_+$
end loop

2.1 Simplified Objective [3 points]

Say we were intent on using a simpler loss function. Our objective function (1) includes a term maximizing the distance between $\mathbf{h}' + \boldsymbol{\ell}$ and $\mathbf{t}'$. If we instead simplified the objective, and just tried to minimize

$$L_{\text{simple}} = \sum_{(h,\ell,t)\in S} d(\mathbf{h}+\boldsymbol{\ell},\mathbf{t}), \quad (2)$$

we would obtain a useless embedding. Give an example of a simple graph and corresponding embeddings which will minimize the new objective function (2) all the way to zero, but still give a completely useless embedding.
Hint: Your graph should be non-trivial, i.e., it should include at least two nodes and at least one edge. Assume the embeddings are in 2 dimensions, i.e., $k = 2$. What happens if $\boldsymbol{\ell} = \mathbf{0}$?

⋆ **Solution** ⋆

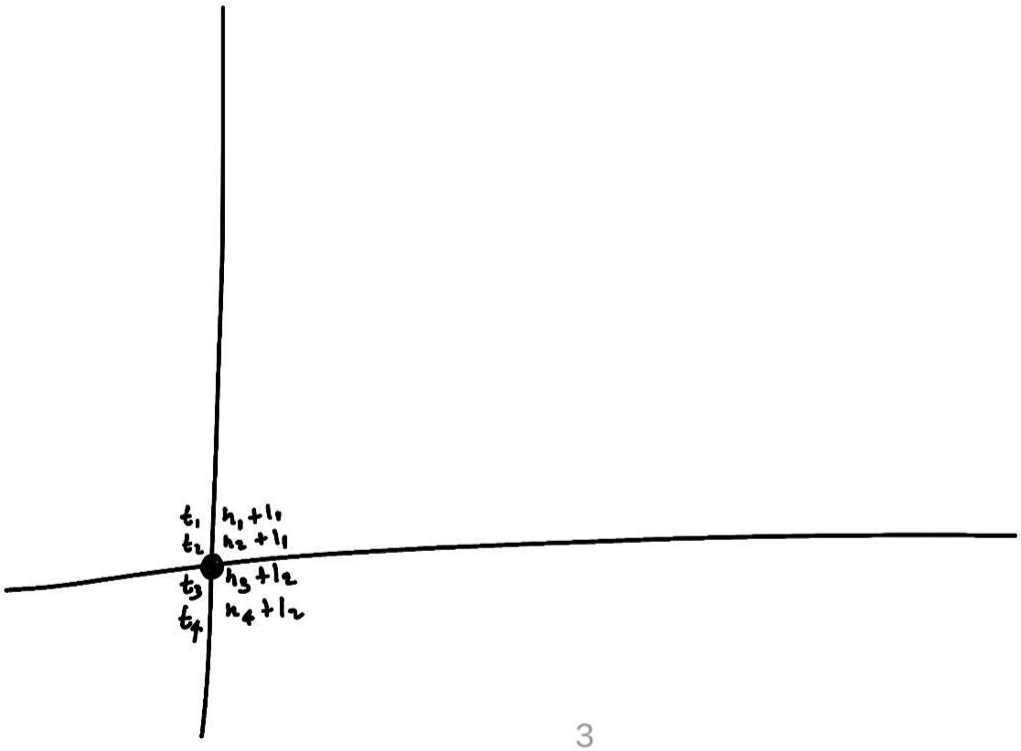
Let us assume the following pairs:

Head	Tail	Relations
h1	t1	l1
h2	t2	l2
h3	t3	l3
h4	t4	l4

If the objective function is defined as

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

As the model attempts to minimize the loss, the embedding values for all pairs of h, l, and t would converge toward zero. Setting the embeddings to zero would artificially result in a zero loss without learning a meaningful representation. One possible solution is:



3

2.2 Utility of γ [5 points]

We are interested in understanding what the margin term γ accomplishes. If we removed the margin term γ from our loss, and instead optimized

$$L_{\text{no margin}} = \sum_{(h,\ell,t)\in S} \sum_{(h',\ell t')\in S'_{(h,\ell,t)}} [d(\mathbf{h}+\ell,\mathbf{t}) - d(\mathbf{h}'+\ell,\mathbf{t}')]_+, \quad (3)$$

it turns out that we would again obtain a useless embedding. Give an example of a simple graph and corresponding embeddings which will minimize the new objective function (3) all the way to zero, but still give a completely useless embedding. By useless, we mean that in your example, you cannot tell just from the embeddings whether two nodes are linked by a particular relation (Note: your graph should be non-trivial, i.e., it should include at least two nodes and at least one edge. Assume the embeddings are in 2 dimensions, i.e., $k = 2$.)

⋆ Solution ⋆

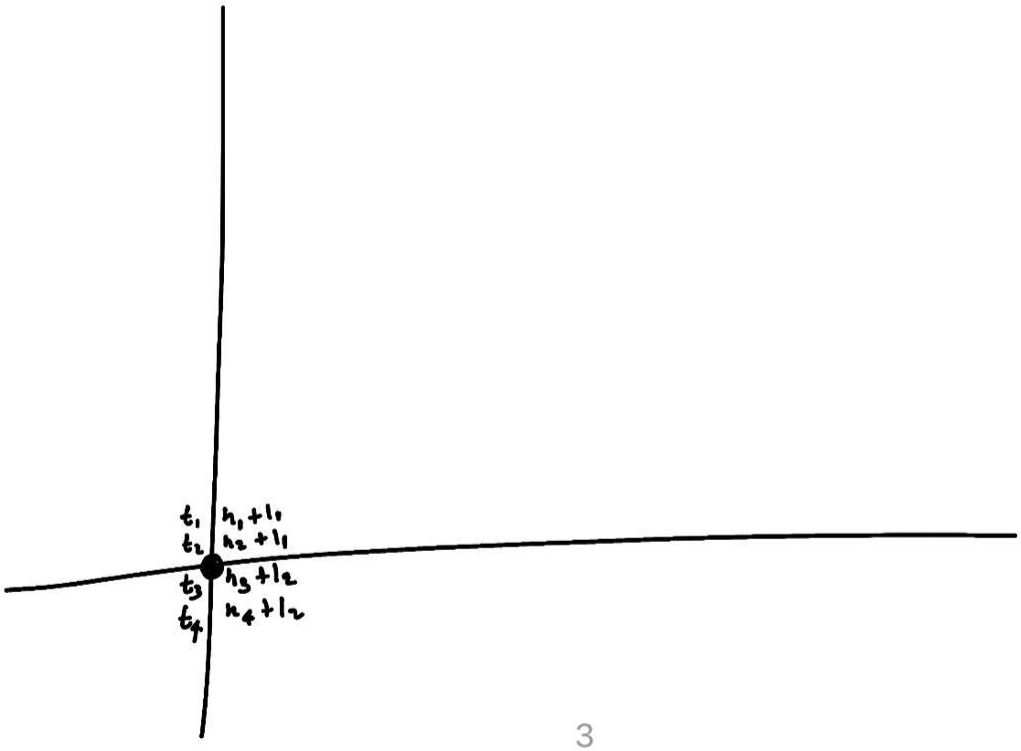
$$\mathcal{L} = \sum_{(h,\ell,t)\in S} \left(\sum_{(h',\ell,t')\in S'_{(h,\ell,t)}} [\gamma + d(\mathbf{h}+\boldsymbol{\ell},\mathbf{t}) - d(\mathbf{h}'+\boldsymbol{\ell},\mathbf{t}')]_+ \right)$$

The original formula as shown here is a hinge loss. We take the maximum of

max(0, γ + (d(h+l,t) – d(h'+l, t')).

In this case, γ represents the margin. It ensures a minimum distance between the positive and negative instances, set to at least γ. Without γ, the distance between positive and negative instances can be zero, yielding a loss of zero without learning a meaningful representation. Consequently, this makes the negative and positive samples inseparable. This scenario could result in a graph similar to the following for the given pairs.

Head	Tail	Relations
h1	t1	l1
h2	t2	l2
h3	t3	l3
h4	t4	l4



3

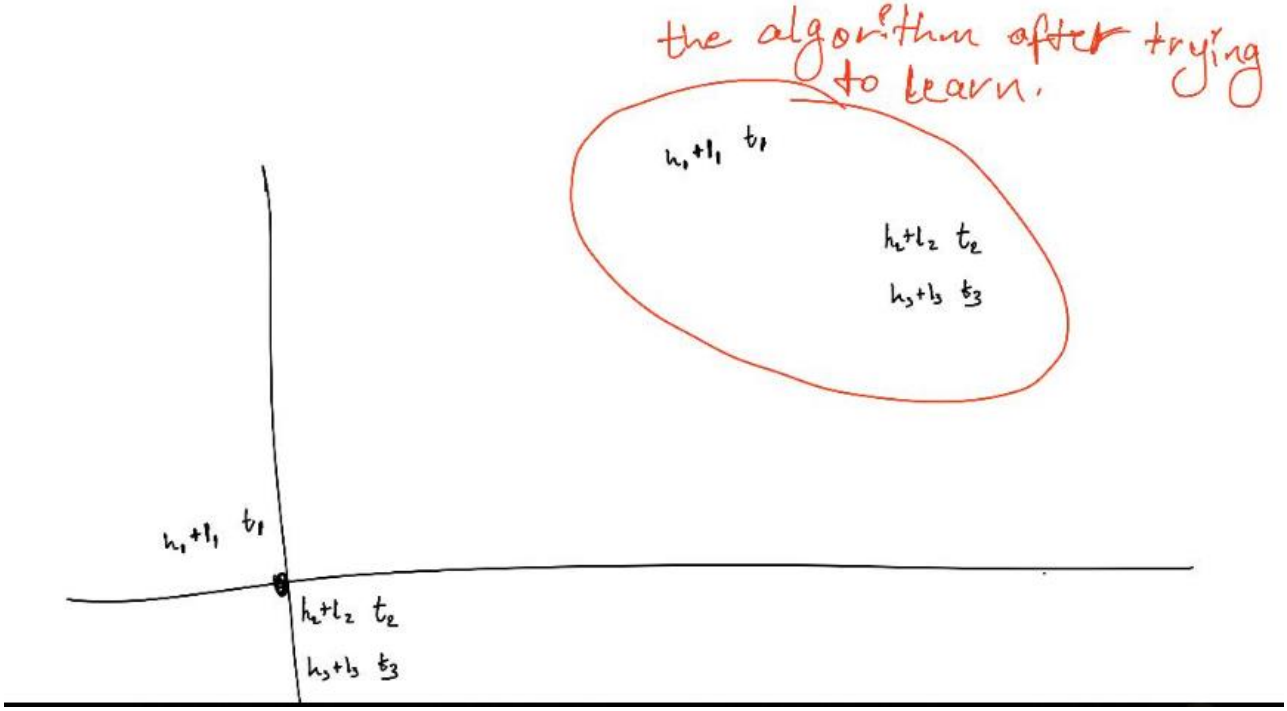
2.3 Normalizing the embeddings [5 points]

Recall that TransE normalizes every entity embedding to have unit length (see line 5 of the algorithm). The quality of our embeddings would be much worse if we did not have this step. To understand why, imagine running the algorithm with line 5 omitted. What could the algorithm do to trivially minimize the loss in this case? What would the embeddings it generates look like?

⋆ Solution ⋆

The answer is stated in the paper itself. The embeddings are normalized to a unit vector to prevent them from minimizing the loss trivially by artificially increasing the norms of the entity embeddings, which is the distance between the points and the origin.

As you can see, the algorithm is increasing the norms of the embeddings, which in this case is not helpful because it just rescales the embeddings instead of trying to learn how to separate them within the restricted distance. Normalization helps to impose this restriction and search for a solution in the confined space.



2.4 Expressiveness of TransE embeddings [7 points]

Give an example of a simple graph for which no perfect embedding exists, i.e., no embedding perfectly satisfies $\mathbf{u} + \ell = \mathbf{v}$ for all $(u, \ell, v) \in S$ and $\mathbf{u} + \ell \neq \mathbf{v}$ for $(u, \ell, v) \notin S$, for any choice of entity embeddings ($\mathbf{e}$ for $e \in E$) and relationship embeddings (ℓ for $\ell \in L$). Explain why this graph has no perfect embedding in this system, and what that means about the expressiveness of TransE embeddings. As before, assume the embeddings are in 2 dimensions ($k = 2$).

Hint: By expressiveness of TransE embeddings, we want you to talk about which type of relationships TransE can/cannot model with an example. (Note that the condition for this question is slightly different from that for Question 2.1 and what we ask you to answer is different as well).

⋆ Solution ⋆

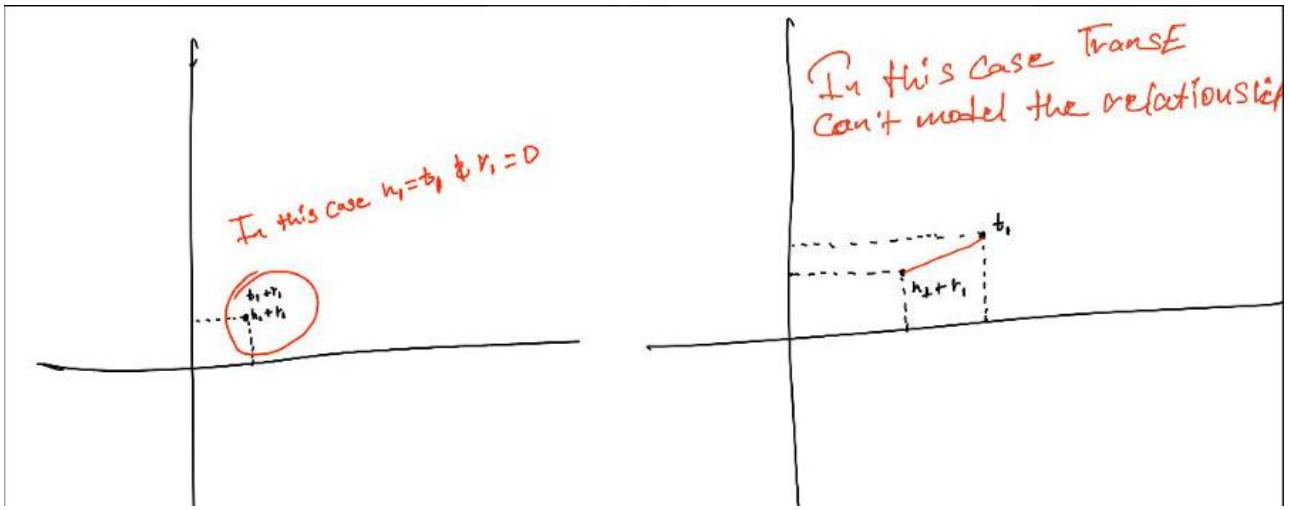
TransE cannot model symmetry relations. Symmetry relations are relations such as:

r(h,t) => r(t, h) for all h and t

For example, the following are symmetry relationships.

- John is married to Rose.
- Rose is married to Joh.

TransE can not model relations such as this because h + r = t and t + r = h cannot be equivalent unless r is 0 and h is equal to t.



3 Expressive Power of Knowledge Graph Embeddings [10 points]

TransE is a common method for learning representations of entities and relations in a knowledge graph. Given a triplet (h, ℓ, t), where entities embedded as h and t are related by a relation embedded as ℓ, TransE trains entity and relation embeddings to make $h + \ell$ close to t. There are some common patterns that relations form:

- Symmetry: A is married to B, and B is married to A.
- Inverse: A is teacher of B, and B is student of A. Note that teacher and student are 2 different relations and have their own embeddings.
- Composition: A is son of B; C is sister of B, then C is aunt of A. Again note that son, sister, and aunt are 3 different relations and have their own embeddings.

3.1 TransE Modeling [3 points]

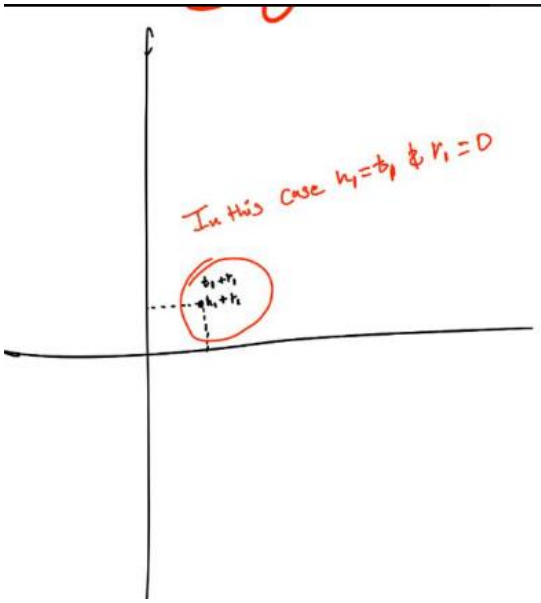
For each of the above relational patterns, can TransE model it perfectly, such that $h + \ell = t$ for all relations? Explain why or why not. Note that here $\mathbf{0}$ embeddings for relation are undesirable since that means two entities related by that relation are identical and not distinguishable.

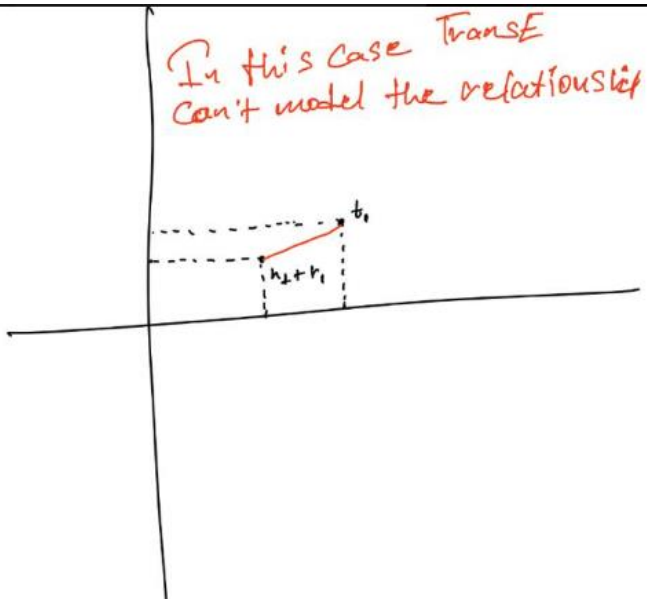
⋆ Solution ⋆

1. Symmetry

As also discussed in the previous answer, TransE can model inverse and composition relations. However, it cannot model symmetric relations.

- Symmetry relations are relation such as
  - r(h,t) => r(t,h) for all h and t pairs.
- For example
  - John is married to Rose.
  - Rose is married to John.
- TransE cannot model relations such as this because:
  - t + r = t and t + r = h cannot be equal unless r equal to zero and h equals to t.





2. Inverse

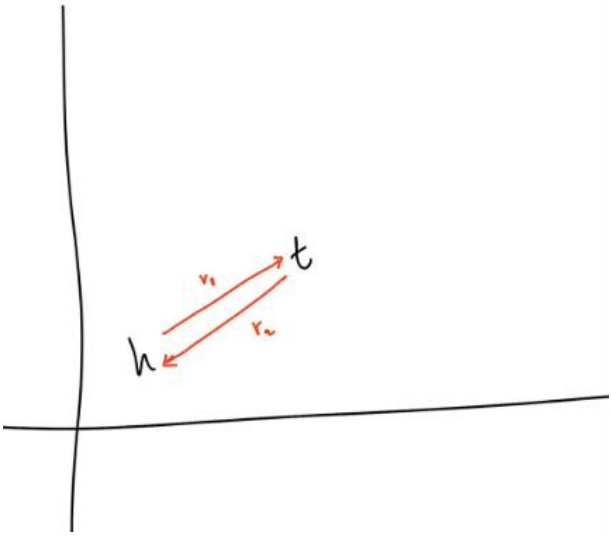
TransE can model inverse relations such as

r1(h,t) => r2(t,h)

For example

- John is the brother of Rose.
- Rose is the sister of John.

h+r1 = t and t +r2 = h if r1 = -r2



3. Composition

TransE can also model composition relationships such as

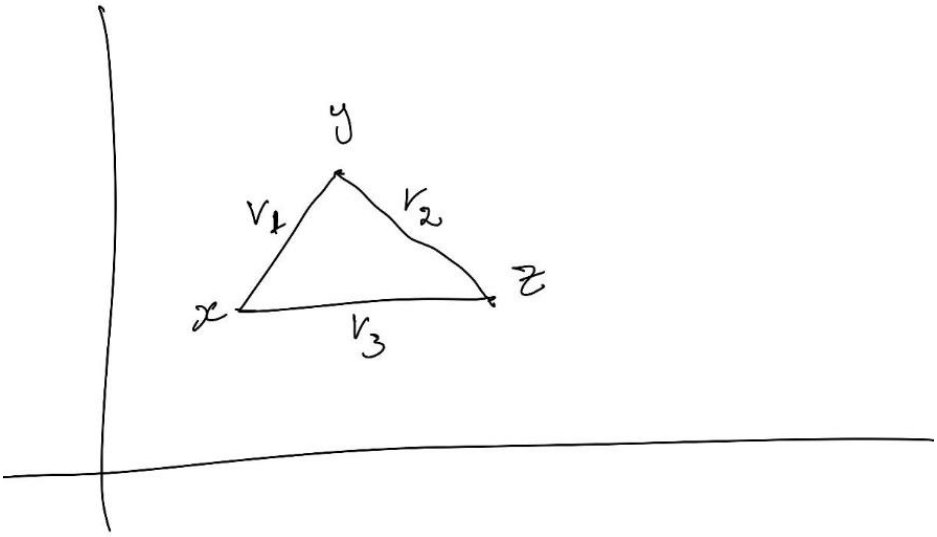
r1(x,y) ∧ r2(y,z) = r3 (x,z) for all x,y,z

An example of such a relationship is

- My father's children are my siblings.

This can be represented as

r1 + r2 = r3 or



3.2 RotatE Modeling [3 points]

Consider a new model, RotatE. Instead of training embeddings such that $h+\ell \approx t$, we train embeddings such that $h \circ \ell \approx t$. Here $\circ$ means rotation. You can think of h as a vector of dimension $2d$, representing d 2D points. ℓ is a d-dimensional vector specifying rotation angles. When applying $\circ$, For all $i \in 0 \ldots d-1$, (h_{2i}, h_{2i+1}) is rotated clockwise by l_i. Similar to TransE, the entity embeddings are also normalized to L2 norm 1. Can RotatE model the above 3 relation patterns perfectly? Why or why not?

⋆ Solution ⋆

- Yes, rotate e can model symmetry, inverse, and composition relationships.

1. Symmetry

- Symmetry relations are relation such as
  - r(h,t) => r(t,h) for all h and t pairs.
- For example
  - John is married to Rose.
  - Rose is married to John.

RotatE basically applies a rotation of the entities by r on the condition that $\mathbf{r_i^2 = 1}$ **or** $r = \pm 1$. This means h = r ◦ t = r ◦ r ◦ h. Which means r ◦ r = 1. In this case one we can model the relationship because it allows us to rotate from head to tail and tail to head. Let us consider the following:

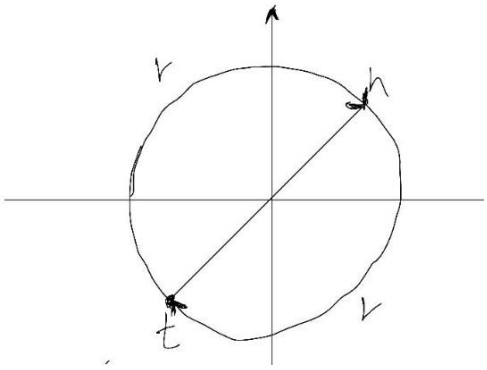
h=(1,2) t =(-1,-2)

RotatE can learn a rotation matrix in real space.

r = [[-1, 0],

[0, -1]]

This matrix can model the relationship.



2. Inverse

Similarly, RotatE can model inverse relations.

Inverse relations such as

$r_1(h,t) \Rightarrow r_2(t,h)$

For example

- John is the brother of Rose.

- Rose is the sister of John.

Two relations r_1 and r_2 are considered inverses if and only $\mathbf{r1} \circ \boldsymbol{r}\mathbf{2} = \mathbf{1}$ or $\mathbf{r2} = \overline{r1}$. Assuming this x = r1 ◦ y = r1 ◦ r2 ◦ x, which means this equality r1 ◦ r2 = 1 or $\mathbf{r2} = \overline{r1}$ must hold.

3. Composition

RotetE can also model composition relationships such as

r1(y, x) ◦ r2(y, z) = r3 (x, z)

An example of such a relationship is

- My father's children are my siblings.

Let x = y ◦ r1, z = y ◦ r2, and x = z ◦ r3 then

x = y ◦ r2 ◦ r3 and replacing x with y ◦ r1

y ◦ r1 = y ◦ r2 ◦ r3

Finally,

r1 = r2 ◦ r3

3.3 Failure Cases [4 points]

Give an example of a graph that RotatE cannot model perfectly. Can TransE model this graph perfectly? Assume that relation embeddings cannot be $\mathbf{0}$ in either model.

⋆ Solution ⋆

RotatE and TransE cannot model non-commutative relationships. An example of a non-commutative relationship is

- Your husband's sister is not the same thing as your sister's husband.

Let r1 = husband & r2 = sister

Your husband's sister = $x \circ r1 \circ r2$ and Your sister's husband = $x \circ r2 \circ r1$

RotatE assumes that this Your husband's sister is equivalent to Your sister's husband. However,
- $\boldsymbol{r1} \circ \boldsymbol{r2} \neq \boldsymbol{r2} \circ \boldsymbol{r1}$

Here the multiplication is not commutative.