CS224W Homework 2

Due: November 4, 2024

1 Node Embeddings with TransE [21 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.[1] 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 $\mathbb{R}^k$ (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 $\ell \in \mathbb{R}^k$, 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 $\mathbf{h} + \boldsymbol{\ell} \approx \mathbf{t}$. Simultanesouly, TransE tries to make sure that $\mathbf{h} + \boldsymbol{\ell} \not\approx \mathbf{t}$ if the edge (h, ℓ, t) does not exist.

[1]See the 2013 NeurIPS paper by Bordes et al: https://papers.nips.cc/paper/5071-translating-embeddings-formodeling-multi-relational-data.pdf

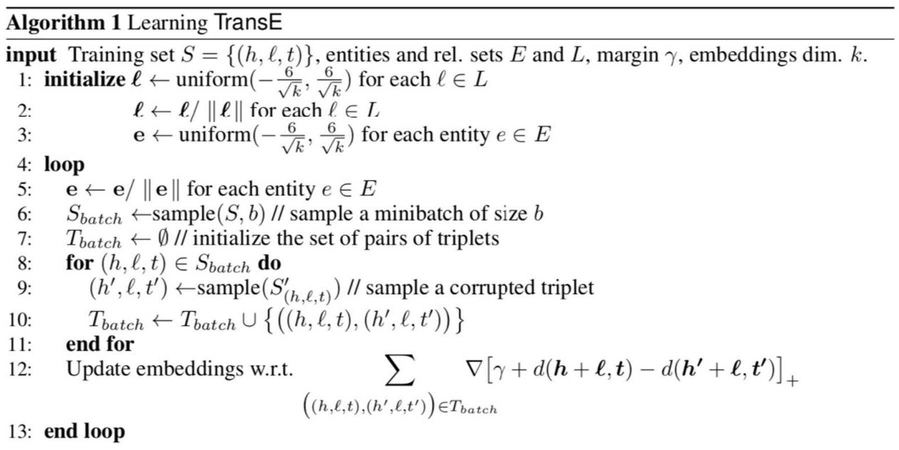
Note on notation: we will use unbolded letters e, ℓ, etc. to denote the entities and relationships in the graph, and bold letters $\mathbf{e}, \boldsymbol{\ell}$, etc., to denote their corresponding embeddings. TransE accomplishes this by minimizing the following loss:

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

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 $\gamma$, embeddings dim. $k$.
 1: initialize $\boldsymbol{\ell} \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each $\ell \in L$
 2:            $\boldsymbol{\ell} \leftarrow \boldsymbol{\ell} / \|\boldsymbol{\ell}\|$ for each $\ell \in L$
 3:            $\mathbf{e} \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each entity $e \in E$
 4: loop
 5:    $\mathbf{e} \leftarrow \mathbf{e} / \|\mathbf{e}\|$ for each entity $e \in E$
 6:    $S_{batch} \leftarrow \text{sample}(S, b)$ // sample a minibatch of size $b$
 7:    $T_{batch} \leftarrow \emptyset$ // initialize the set of pairs of triplets
 8:    for $(h,\ell,t) \in S_{batch}$ do
 9:       $(h',\ell,t') \leftarrow \text{sample}(S'_{(h,\ell,t)})$ // sample a corrupted triplet
10:       $T_{batch} \leftarrow T_{batch} \cup \{((h,\ell,t),(h',\ell,t'))\}$
11:    end for
12:    Update embeddings w.r.t. $\sum_{((h,\ell,t),(h',\ell,t')) \in T_{batch}} \nabla\left[\gamma + d(\boldsymbol{h}+\boldsymbol{\ell},\boldsymbol{t}) - d(\boldsymbol{h'}+\boldsymbol{\ell},\boldsymbol{t'})\right]_+$
13: end loop
```

1.1 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

$$\mathcal{L}_{\text{simple}} = \sum_{(h,\ell,t)\in S} d(\mathbf{h}+\boldsymbol{\ell},\mathbf{t}), \tag{2}$$

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 ★ A useless embedding can be achieved by setting all entity embeddings to be identical and all relation embeddings to be the zero vector.

Example Graph and Embeddings:

- **Graph:** Consider a simple graph with two entities, A and B, and one relationship type, R. The graph contains a single edge (A, R, B).
- **Embeddings:** Let the embeddings be in 2D space ($k = 2$). We can set the embeddings as follows:
  - $\mathbf{a} = [0.5, 0.5]$
  - $\mathbf{b} = [0.5, 0.5]$
  - $\boldsymbol{\ell_R} = [0, 0]$

Minimizing the Objective to Zero: With these embeddings, the objective function becomes:

$$\mathcal{L}_{\text{simple}} = d(\mathbf{a} + \boldsymbol{\ell_R}, \mathbf{b}) = \|(\mathbf{a} + \boldsymbol{\ell_R}) - \mathbf{b}\|_2$$

$$= \|([0.5, 0.5] + [0, 0]) - [0.5, 0.5]\|_2 = \|[0.5, 0.5] - [0.5, 0.5]\|_2 = \|[0, 0]\|_2 = 0$$

The objective is minimized perfectly to zero.

Why the Embedding is Useless: This embedding is useless because every entity in the graph is mapped to the exact same point in the embedding space. The model has learned no distinguishing features between the nodes. It is impossible to use these embeddings to determine if two nodes are the same or different, let alone predict relationships between them.

1.2 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

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

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** ★ A useless embedding can be achieved by collapsing all entities and relations into a state where the distance for any true triplet is equal to the distance for any corrupted triplet. The simplest way to achieve this is to make all these distances zero.

Example Graph and Embeddings:

- **Graph:** Consider a graph with two entities, A and B, and one relationship R, forming the edge (A, R, B).
- **Embeddings:** Let all entity embeddings be identical, and the relation embedding be the zero vector.
  - $\mathbf{a} = [0.5, 0.5]$
  - $\mathbf{b} = [0.5, 0.5]$
  - $\boldsymbol{\ell}_{\boldsymbol{R}} = [0, 0]$

Minimizing the Objective to Zero: The loss for the true triplet (A, R, B) is a sum over all its "corrupted" versions. Let's consider any corrupted triplet, for example (A', R, B) where A' is another entity in the graph. Since all entity embeddings are identical, $\mathbf{a}' = \mathbf{a} = \mathbf{b} = [0.5, 0.5]$.

We calculate the distance for the true triplet, d_{pos}, and the corrupted triplet, d_{neg}:

- $d_{\text{pos}} = d(\mathbf{a} + \boldsymbol{\ell}_{\boldsymbol{R}}, \mathbf{b}) = \|([0.5, 0.5] + [0, 0]) - [0.5, 0.5]\|_2 = 0.$
- $d_{\text{neg}} = d(\mathbf{a}' + \boldsymbol{\ell}_{\boldsymbol{R}}, \mathbf{b}) = \|([0.5, 0.5] + [0, 0]) - [0.5, 0.5]\|_2 = 0.$

The contribution to the loss for this specific corruption is:

$$[d_{\text{pos}} - d_{\text{neg}}]_+ = [0 - 0]_+ = 0$$

Since this holds for any possible corruption (as all entity embeddings are the same), the total loss $\mathcal{L}_{\text{no margin}}$ is zero.

Why the Embedding is Useless: This embedding is useless for the same reason as in the previous question: all entities are mapped to a single point, making them indistinguishable. The model has perfectly minimized the loss without learning any of the graph's structure. The margin term γ is crucial because it prevents this collapse by forcing a separation ($d_{\text{pos}} + \gamma \leq d_{\text{neg}}$), which cannot be satisfied when both distances are zero.

1.3 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** ★ If the normalization step (line 5 of the algorithm) were omitted, the model could trivially minimize the loss function by driving the norms of the entity embeddings to infinity.

Trivial Minimization Strategy: The objective is to satisfy the inequality $d(\mathbf{h}+\boldsymbol{\ell},\mathbf{t})+\gamma \leq d(\mathbf{h}'+\boldsymbol{\ell},\mathbf{t}')$ for all true triplets (h,ℓ,t) and their corrupted versions. This is equivalent to $\|\mathbf{h}+\boldsymbol{\ell}-\mathbf{t}\|_2+\gamma \leq \|\mathbf{h}'+\boldsymbol{\ell}-\mathbf{t}'\|_2$.

Without the normalization constraint, the algorithm can learn embeddings with arbitrarily large magnitudes. The model can simply scale up the norms of all entity embeddings by a very large factor $M \to \infty$. Let the scaled embeddings be $\mathbf{e}_{\text{new}} = M \cdot \mathbf{e}$ for every entity $e \in E$. The inequality becomes:

$$\|M\mathbf{h}+\boldsymbol{\ell}-M\mathbf{t}\|_2+\gamma \leq \|M\mathbf{h}'+\boldsymbol{\ell}-M\mathbf{t}'\|_2$$

As M becomes very large, the contribution of the relation vector $\boldsymbol{\ell}$ becomes negligible compared to the entity embeddings. The inequality approaches:

$$M\|\mathbf{h}-\mathbf{t}\|_2+\gamma \leq M\|\mathbf{h}'-\mathbf{t}'\|_2$$

Dividing by M, we get:

$$\|\mathbf{h}-\mathbf{t}\|_2+\frac{\gamma}{M} \leq \|\mathbf{h}'-\mathbf{t}'\|_2$$

As $M \to \infty$, the term $\frac{\gamma}{M} \to 0$. The model now only needs to satisfy $\|\mathbf{h}-\mathbf{t}\|_2 \leq \|\mathbf{h}'-\mathbf{t}'\|_2$, which it can achieve with minimal learning. The loss can be made arbitrarily close to zero by simply increasing the magnitude of all entity embeddings indefinitely.

Resulting Embeddings: The embeddings generated would be useless. Their magnitudes would grow without bound during training, leading to numerical instability. The final embeddings would have enormous norm values, and the meaningful geometric relationships that TransE is designed to capture would be lost. The normalization step is crucial as it constrains the embeddings to a compact space (the surface of a unit hypersphere), forcing the model to learn meaningful angular and directional relationships rather than simply scaling magnitudes.

1.4 Expressiveness of TransE embeddings [8 points]

Give an example of a simple graph for which no perfect embedding exists, i.e., no embedding perfectly satisfies $\mathbf{u}+\boldsymbol{\ell}=\mathbf{v}$ for all $(u,\ell,v) \in S$ and $\mathbf{u}+\boldsymbol{\ell} \neq \mathbf{v}$ for $(u,\ell,v) \notin S$, for any choice of entity embeddings ($\mathbf{e}$ for $e \in E$) and

relationship embeddings ($\boldsymbol{\ell}$ 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 ★ **Example Graph:** A simple graph for which no perfect TransE embedding exists is one that models a symmetric relationship. Consider a graph with two entities, A and B, and a single symmetric relationship type, R (e.g., "is friends with"). The set of edges S contains two triplets:

$$S = \{(A, R, B), (B, R, A)\}$$

Let's also assume there is another entity C not related to A or B, so $(A, R, C) \notin S$.

Why No Perfect Embedding Exists: A perfect embedding must satisfy $\mathbf{h}+\boldsymbol{\ell} = \mathbf{t}$ for all true triplets and $\mathbf{h} + \boldsymbol{\ell} \neq \mathbf{t}$ for all false triplets.

1. From the triplet (A, R, B), a perfect embedding requires:

$$\mathbf{a} + \boldsymbol{\ell}_{\boldsymbol{R}} = \mathbf{b} \quad \implies \quad \boldsymbol{\ell}_{\boldsymbol{R}} = \mathbf{b} - \mathbf{a}$$

2. From the triplet (B, R, A), a perfect embedding requires:

$$\mathbf{b} + \boldsymbol{\ell}_{\boldsymbol{R}} = \mathbf{a} \quad \implies \quad \boldsymbol{\ell}_{\boldsymbol{R}} = \mathbf{a} - \mathbf{b}$$

Combining these two requirements, we get $\mathbf{b} - \mathbf{a} = \mathbf{a} - \mathbf{b}$, which simplifies to $2\mathbf{b} = 2\mathbf{a}$, or $\mathbf{a} = \mathbf{b}$. This means that for TransE to perfectly model this relationship, the embeddings for entities A and B must be identical.

Furthermore, if we substitute $\mathbf{a} = \mathbf{b}$ back into the first equation, we get $\boldsymbol{\ell}_{\boldsymbol{R}} = \mathbf{a} - \mathbf{a} = \mathbf{0}$. So, the relation embedding must be the zero vector.

This leads to a contradiction. If $\mathbf{a} = \mathbf{b}$ and $\boldsymbol{\ell}_{\boldsymbol{R}} = \mathbf{0}$, then for the false triplet (A, R, C), the model would check if $\mathbf{a} + \boldsymbol{\ell}_{\boldsymbol{R}} \neq \mathbf{c}$, which is $\mathbf{a} \neq \mathbf{c}$. This part is fine. However, the model has completely lost the ability to distinguish between A and B, which is a failure. The very purpose of an embedding is to provide a unique representation for each entity. By forcing $\mathbf{a} = \mathbf{b}$, the model has failed.

Expressiveness of TransE Embeddings: This example demonstrates a key limitation in the expressiveness of TransE. The translational model ($\mathbf{h} + \boldsymbol{\ell} \approx \mathbf{t}$) is inherently directional and struggles to capture relationships that are not one-to-one or one-to-many. Specifically, **TransE cannot effectively model symmetric (or many-to-many) relations**. It is forced to learn identical embeddings for entities linked by such relations and a zero vector for the relation itself, which is not a useful representation. TransE's expressiveness is thus best suited for hierarchical or directed, non-symmetric relationships.

2 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.

2.1 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 ★

- **Symmetry: No.**

  A symmetric relation 'l' between entities 'h' and 't' implies two facts: (h, l, t) and (t, l, h). For TransE to model this perfectly, both equations must hold:

  1. $\mathbf{h} + \boldsymbol{\ell} = \mathbf{t}$
  2. $\mathbf{t} + \boldsymbol{\ell} = \mathbf{h}$

  Substituting the first equation into the second gives $(\mathbf{h} + \boldsymbol{\ell}) + \boldsymbol{\ell} = \mathbf{h}$, which simplifies to $2\boldsymbol{\ell} = \mathbf{0}$, meaning $\boldsymbol{\ell} = \mathbf{0}$. This is an undesirable trivial solution, as it implies the two related entities are identical ($\mathbf{h} = \mathbf{t}$). Therefore, TransE cannot model symmetric relations non-trivially.

- **Inverse: Yes.**

  An inverse relation pattern involves two different relations, ℓ_1 and ℓ_2, such that (h, ℓ_1, t) implies (t, ℓ_2, h). The two equations are:

  1. $\mathbf{h} + \boldsymbol{\ell_1} = \mathbf{t}$
  2. $\mathbf{t} + \boldsymbol{\ell_2} = \mathbf{h}$

From the first equation, we have $\boldsymbol{\ell_1} = \mathbf{t} - \mathbf{h}$. From the second, we have $\boldsymbol{\ell_2} = \mathbf{h} - \mathbf{t}$. This leads to the consistent and non-trivial solution $\boldsymbol{\ell_2} = -\boldsymbol{\ell_1}$. TransE can model inverse relations perfectly by learning relation vectors that are negatives of each other.

- **Composition: Yes.**

  A composition pattern involves three relations, ℓ_1, ℓ_2, ℓ_3, such that (h, ℓ_1, z) and (z, ℓ_2, t) implies (h, ℓ_3, t). The three equations are:

  1. $\mathbf{h} + \boldsymbol{\ell_1} = \mathbf{z}$
  2. $\mathbf{z} + \boldsymbol{\ell_2} = \mathbf{t}$
  3. $\mathbf{h} + \boldsymbol{\ell_3} = \mathbf{t}$

  By substituting the first equation into the second, we get $(\mathbf{h} + \boldsymbol{\ell_1}) + \boldsymbol{\ell_2} = \mathbf{t}$. Comparing this with the third equation, we must have $\mathbf{h} + \boldsymbol{\ell_1} + \boldsymbol{\ell_2} = \mathbf{h} + \boldsymbol{\ell_3}$, which simplifies to $\boldsymbol{\ell_3} = \boldsymbol{\ell_1} + \boldsymbol{\ell_2}$. This is a valid, non-trivial solution where the composed relation vector is the sum of the component relation vectors. TransE can model composition perfectly.

2.2 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 ★ In RotatE, the model aims to satisfy $\mathbf{h} \circ \boldsymbol{\ell} = \mathbf{t}$, where embeddings are in the complex plane and $\circ$ represents complex multiplication. We also have $|\mathbf{h}| = 1$ for all entities h. For a relation ℓ to be a pure rotation, we must have $|\boldsymbol{\ell}| = 1$.

- **Symmetry: Yes.**

  A symmetric relation ℓ implies $\mathbf{h} \circ \boldsymbol{\ell} = \mathbf{t}$ and $\mathbf{t} \circ \boldsymbol{\ell} = \mathbf{h}$. Substituting the first equation into the second gives $(\mathbf{h} \circ \boldsymbol{\ell}) \circ \boldsymbol{\ell} = \mathbf{h}$, which means $\mathbf{h} \circ \boldsymbol{\ell}^2 = \mathbf{h}$. This simplifies to $\boldsymbol{\ell}^2 = 1$. In the complex plane, this has two solutions: $\boldsymbol{\ell} = 1$ (0-degree rotation) or $\boldsymbol{\ell} = -1$ (180-degree rotation). The $\boldsymbol{\ell} = -1$ solution is non-trivial and perfectly models symmetry: applying a 180-degree rotation twice returns the original vector.

- **Inverse: Yes.**

  An inverse relation pattern involves two relations, ℓ_1 and ℓ_2, such that $\mathbf{h} \circ \boldsymbol{\ell_1} = \mathbf{t}$ and $\mathbf{t} \circ \boldsymbol{\ell_2} = \mathbf{h}$. From the first equation, we get $\mathbf{h} = \mathbf{t}/\boldsymbol{\ell_1}$. From the second, we get $\mathbf{t} = \mathbf{h}/\boldsymbol{\ell_2}$. This leads to the consistent, non-trivial solution where $\boldsymbol{\ell_2} = 1/\boldsymbol{\ell_1} = \boldsymbol{\ell_1}^{-1}$. This means the second relation is the inverse rotation of the first (e.g., if one is a +45 degree rotation, the other is a -45 degree rotation). RotatE can model this perfectly.

- **Composition: Yes.**

  A composition pattern involves three relations, ℓ_1, ℓ_2, ℓ_3, such that $\mathbf{h} \circ \boldsymbol{\ell_1} = \mathbf{z}$ and $\mathbf{z} \circ \boldsymbol{\ell_2} = \mathbf{t}$ implies $\mathbf{h} \circ \boldsymbol{\ell_3} = \mathbf{t}$. Substituting the first equation into the second gives $(\mathbf{h} \circ \boldsymbol{\ell_1}) \circ \boldsymbol{\ell_2} = \mathbf{t}$, which simplifies to $\mathbf{h} \circ (\boldsymbol{\ell_1} \circ \boldsymbol{\ell_2}) = \mathbf{t}$. Comparing this with the third equation, $\mathbf{h} \circ \boldsymbol{\ell_3} = \mathbf{t}$, we get the non-trivial solution $\boldsymbol{\ell_3} = \boldsymbol{\ell_1} \circ \boldsymbol{\ell_2}$. The composed relation corresponds to the product of the component relation rotations. RotatE can model this perfectly.

2.3 Failure Cases [4 points]

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

★ Solution ★ **Example Graph:** Consider a simple graph representing a symmetric relationship, which RotatE can model but TransE cannot (given the constraints). Let the graph contain two entities, A and B, and one relation R (e.g., "is_roommate_with"). The graph consists of two facts:

- (A, R, B)
- (B, R, A)

Why RotatE Can Model This Graph: RotatE needs to satisfy two equations:

1. $\mathbf{a} \circ \boldsymbol{\ell_R} = \mathbf{b}$
2. $\mathbf{b} \circ \boldsymbol{\ell_R} = \mathbf{a}$

As shown in question 2.2, this is perfectly modeled by setting the relation $\boldsymbol{\ell_R}$ to be a 180-degree rotation (i.e., $\boldsymbol{\ell_R} = -1$ in the complex plane). This is a valid, non-zero relation embedding.

Why TransE Cannot Model This Graph: TransE needs to satisfy two equations:

1. $\mathbf{a} + \boldsymbol{\ell_R} = \mathbf{b}$
2. $\mathbf{b} + \boldsymbol{\ell_R} = \mathbf{a}$

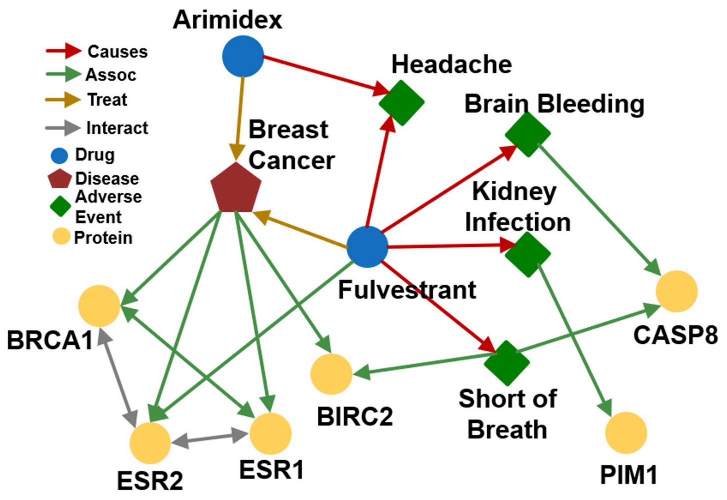
As shown in question 2.1, solving these two equations forces the conclusion that $\boldsymbol{\ell_R} = \mathbf{0}$. However, the problem states that we must assume relation embeddings cannot be $\mathbf{0}$. Since the only way for TransE to model this symmetric structure is with a zero embedding, it fails to model the graph under the given constraints.

3 Queries on Knowledge Graphs [14 points]

Knowledge graphs (KGs) can encode a wealth of information about the world. Beyond representing the information using knowledge graphs, we can often derive previously unknown insights about entities and relations in the graphs. In this question, we will explore different approaches for reasoning over knowledge graphs. Recall from that lecture that we are interested in predicting `tail` nodes given (`head`, `relation`). We will use the same formulation throughout this question.

3.1 Path Queries on Complete KGs [3 points]

Consider the biomedicine knowledge graph from lecture. Assume the question of interest is: "What proteins are associated with diseases treated by Arimidex?" Write the question in query form (eg. (e:AnchorEntity, (r:Relation))) and find the answer(s) to the query. Partial credit will be rewarded to correct intermediate steps.



★ **Solution** ★ **Query Form:** The question "What proteins are associated with diseases treated by Arimidex?" can be broken down into a path query. We start at the anchor entity "Arimidex", find what it treats, and then find what proteins are associated with that disease.

- Anchor Entity (e): `Arimidex`
- First Relation (r1): `Treat`
- Second Relation (r2): `Assoc`

The query in the specified form is: **(e: Arimidex, (r: Treat, r: Assoc))**

Answer(s) to the Query: To find the answers, we trace the path on the knowledge graph:

1. **Step 1:** Start at the anchor node `Arimidex`. Following the `Treat` edge (orange arrow) leads to the intermediate entity `Breast Cancer`.
2. **Step 2:** From the intermediate node `Breast Cancer`, we follow all outgoing `Assoc` edges (green arrows) to find the final answer set. These edges point to three protein nodes.

The final set of answers to the query is: **{BRCA1, ESR1, BIRC2}**

3.2 Conjunctive Queries on Complete KGs [1 point]

Consider the same biomedicine knowledge graph from before. Write a conjunctive query to which BIRC2 is the only answer using drugs as anchor entities. If such a query doesn't exist, provide a one-sentence explanation.

★ Solution ★ No, such a query does not exist. Any path query starting from a drug anchor entity that reaches BIRC2 must first pass through Breast Cancer, and from Breast Cancer, the 'Assoc' relation also leads to BRCA1 and ESR1, making it impossible to isolate BIRC2 as the sole answer.

3.3 Incomplete KGs [2 points]

A major issue with direct traversals on knowledge graphs is that they are usually incomplete in reality. One solution is to encode entities, relations, and queries in an embedding space that meaningfully organizes information. We would then be able to impute missing relation links by considering all nearby points of the query embedding as answers to the query. From lecture, we learned that TransE embeddings can be used for this. Can you come up with a way to adopt DistMult embeddings, which uses bilinear modeling, for answering path queries? If yes, describe in one or two sentences what can be modified from the TransE application. If no, provide a one-sentence explanation.

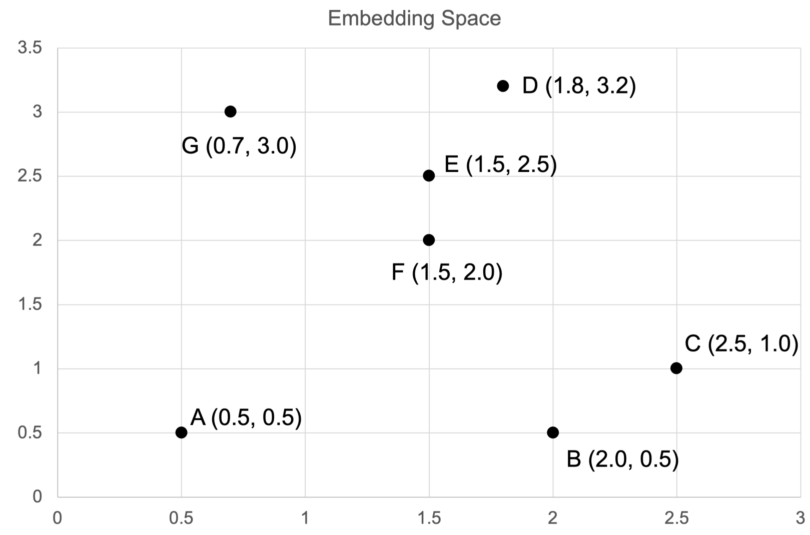
★ Solution ★ No, DistMult embeddings cannot be adapted to answer path queries in the same way as TransE. The core operation in TransE is vector addition, which allows a path query to be represented as a single query embedding by summing the anchor and relation vectors; however, DistMult's element-wise product operation does not have a similar mechanism for composing relations to represent a multi-hop path.

3.4 Query2box [8 points]

Query2box is an effective approach for answering complex conjunctive queries. Consider the following 2-dimensional embedding space. Assume that there are 7 entities $A, B, C, D, E, F, G \in V$, whose embeddings are shown below. There are 3 relations: R_1, R_2, R_3. $R_1 \in R$ shifts the center of a box by $(0.25, 2)$ and increases the width and height of a box by $(0.5, 2)$. R_2 shifts the center of a box by $(1, 0)$ and increases the width and height of a box by $(1, 0)$. R_3 shifts the center of a box by $(-0.75, 1)$ and increases the width and height of a box by $(1.5, 3)$.

Use the Query2box projection operator to find the answers to the conjunctive query: ((e:A, (r:R_1, r:R_2), (e:C, (r:R_3)). Show your work. Partial credit will be rewarded to correct intermediate steps.

Note: Shifting by a negative value means moving towards the left or bottom. Increasing the width and height by an amount means adding that amount in absolute value, not multiplying that amount as a factor. Assume that each path query starts with a box centered at the anchor entity with zero width and height.



★ **Solution** ★ The final answer is the set of entities that lie within the intersection of the result boxes from the two path queries. We will solve each path query separately and then find the intersection of their answer sets.

Path Query 1: ((e:A, (r:R1, r:R2)))

1. Start with a zero-sized box at anchor entity A:
   - Center: (0.5, 0.5)
   - Size (width, height): (0, 0)

2. Apply the projection for relation R1 (Shift by (0.25, 2), Increase size by (0.5, 2)):
   - New Center: (0.5, 0.5) + (0.25, 2) = (0.75, 2.5)
   - New Size: (0, 0) + (0.5, 2) = (0.5, 2)
3. Apply the projection for relation R2 (Shift by (1, 0), Increase size by (1, 0)):
   - Final Center: (0.75, 2.5) + (1, 0) = (1.75, 2.5)
   - Final Size: (0.5, 2) + (1, 0) = (1.5, 2)
4. Determine the boundaries of the final box for Query 1:
   - X-range: $[1.75 - 1.5/2, 1.75 + 1.5/2] = [1.0, 2.5]$
   - Y-range: $[2.5 - 2/2, 2.5 + 2/2] = [1.5, 3.5]$
5. The entities inside this box are: **{D, E, F}**.

Path Query 2: (e:C, (r:R3))

1. Start with a zero-sized box at anchor entity C:
   - Center: (2.5, 1.0)
   - Size: (0, 0)
2. Apply the projection for relation R3 (Shift by (-0.75, 1), Increase size by (1.5, 3)):
   - Final Center: (2.5, 1.0) + (-0.75, 1) = (1.75, 2.0)
   - Final Size: (0, 0) + (1.5, 3) = (1.5, 3)
3. Determine the boundaries of the final box for Query 2:
   - X-range: $[1.75 - 1.5/2, 1.75 + 1.5/2] = [1.0, 2.5]$
   - Y-range: $[2.0 - 3/2, 2.0 + 3/2] = [0.5, 3.5]$
4. The entities inside this box are: **{B, C, D, E, F}**.

Final Answer (Intersection): The answer to the conjunctive query is the intersection of the two answer sets:

$$\{D, E, F\} \cap \{B, C, D, E, F\} = \textbf{\{D, E, F\}}$$

4 Honor Code [0 points]

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

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

Note: Read our website on our policy about collaboration!