Homework 2

October 2021

1 GCN (10 points)

Consider a graph $G = (V, E)$, with node features $x(v)$ for each $v \in V$. For each node $v \in V$, let $h_v^{(0)} = x(v)$ be the node's initial embedding. At each iteration k, the embeddings are updated as

$$h_{\mathcal{N}(v)}^{(k)} = \text{aggregate}\left(\left\{h_u^{(k-1)}, \forall u \in \mathcal{N}(v)\right\}\right)$$

$$h_v^{(k)} = \text{combine}\left(h_v^{(k-1)}, h_{\mathcal{N}(v)}^{(k)}\right),$$

for some functions aggregate$(\cdot)$ and combine$(\cdot)$. Note that the argument to the aggregate$(\cdot)$ function, $\{h_u^{(k-1)}, \forall u \in \mathcal{N}(v)\}$, is a *multi-*set. That is, since multiple nodes can have the same embedding, the same element can occur in $\{h_u^{(k-1)}, \forall u \in \mathcal{N}(v)\}$ multiple times. Finally, a graph itself may be embedded by computing some function applied to the multi-set of all the node embeddings at some final iteration K, which we notate as

$$\text{readout}\left(\left\{h_v^{(K)}, \forall v \in V\right\}\right).$$

We want to use the graph embeddings above to test whether two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic*. Recall that this is true if and only if there is some bijection $\phi : V_1 \to V_2$ between nodes of G_1 and nodes of G_2 such for any $u, v \in V_1$,

$$(u, v) \in E_1 \Leftrightarrow (\phi(u), \phi(v)) \in E_2.$$

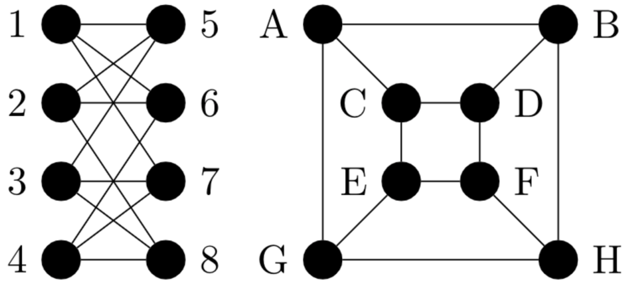
The way we use the model above to test isomorphism is the following. For the two graphs, if their readout functions differ, that is

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

we conclude the graphs are *not* isomorphic. Otherwise, we conclude the graphs are isomorphic. Note that this algorithm is not perfect: graph isomorphism is thought to be hard! Below, we will explore the expressiveness of these graph embeddings.

1.1 Question 1.1 (1 point)

Are the following two graphs isomorphic? If so, demonstrate an isomorphism between the sets of vertices. To demonstrate an isomorphism between two graphs, you need to find a 1-to-1 correspondence between their nodes and edges. If these two graphs are not isomorphic, prove it by finding a structure (node and/or edge) in one graph which is not present in the other.



1.2 Question 1.2 (3 points)

The choice of the aggregate(·) is important for the expressiveness of the model above. Three common choices are:

$$\text{aggregate}_{\text{max}}\left(\left\{h_u^{(k-1)}, \forall u \in \mathcal{N}(v)\right\}\right)_i = \max_{u \in \mathcal{N}(v)}\left(h_u^{(k-1)}\right)_i \quad \text{(element-wise max)} \tag{1}$$

$$\text{aggregate}_{\text{mean}}\left(\left\{h_u^{(k-1)}, \forall u \in \mathcal{N}(v)\right\}\right) = \frac{1}{|\mathcal{N}(v)|}\sum_{u \in \mathcal{N}(v)}\left(h_u^{(k-1)}\right) \tag{2}$$

$$\text{aggregate}_{\text{sum}}\left(\left\{h_u^{(k-1)}, \forall u \in \mathcal{N}(v)\right\}\right) = \sum_{u \in \mathcal{N}(v)}\left(h_u^{(k-1)}\right) \tag{3}$$

Give an example of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ and their initial node features, such that for two nodes $v_1 \in V_1$ and $v_2 \in V_2$ with the same initial features $h_{v_1}^{(0)} = h_{v_2}^{(0)}$, the updated features $h_{v_1}^{(1)}$ and $h_{v_2}^{(1)}$ are equal if we use mean and max aggregation, but different if we use sum aggregation.

Hint: Your node features can be scalars rather than vectors, i.e. one dimensional node features instead of n-dimensional. Also, You are free to arbitrarily choose the number of nodes (e.g. 3 nodes), their connections (i.e. edges between nodes) in your example.

1.3 Question 1.3 (6 points)

Our isomorphism-test algorithm is known to be at most as powerful as the well-known *Weisfeiler-Lehman test* (WL test). At each iteration, this algorithm updates the representation of each node to be the set containing its previous representation and the previous representations of all its neighbours. The full algorithm is below. Prove that our neural model is at most as powerful as the WL test. More precisely, let G_1 and G_2 be non-isomorphic, and suppose that their node embeddings are updated over K iterations with the same aggregate$(\cdot)$ and combine$(\cdot)$ functions. Show that if

```
Algorithm 3: Weisfeiler-Lehman Test
  Data: $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, initial features $x(\cdot)$, number of iterations $K$
  Result: Prediction of whether $G_1$ and $G_2$ are isomorphic
  for $v \in V_1 \cup V_2$ do
    $l_v^{(0)} \leftarrow x(v)$
  end
  for $k = 1, \ldots, K$ do
    for $v \in V_1 \cup V_2$ do
      $l_v^{(k)} \leftarrow \left\{ l_v^{(k-1)} \right\} \cup \left\{ l_u^{(k-1)}, \forall u \in \mathcal{N}(v) \right\}$
    end
  end
  return $\left\{ l_v^{(K)}, \forall v \in V_1 \right\} = \left\{ l_v^{(K)}, \forall v \in V_2 \right\}$
```

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

then the WL test also decides the graphs are not isomorphic.
Hint: You can use proof by contradiction by first assuming that *Weisfeiler-Lehman* test cannot decide whether G_1 and G_2 are isomorphic at the end of K'th iteration.

2 Node Embeddings with TransE (25 points)

** **What to submit**: For Q2.1-2.2, Graph and corresponding useless embeddings which minimize the respective losses to 0.. For Q2.3, Discussion of behavior of algorithm and resulting embeddings without normalization step. For Q2.4, Graph for which no perfect embedding exists, with justification. And Discussion on TransE embedding expressiveness.**
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 in Homework 1 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**. 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, $\ell \in L$ 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 $\mathbf{h} + \ell \approx \mathbf{t}$. 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 $\mathbf{e}, \ell$, etc., to denote their corresponding embeddings. TransE accomplishes this by minimizing the following loss:

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

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.

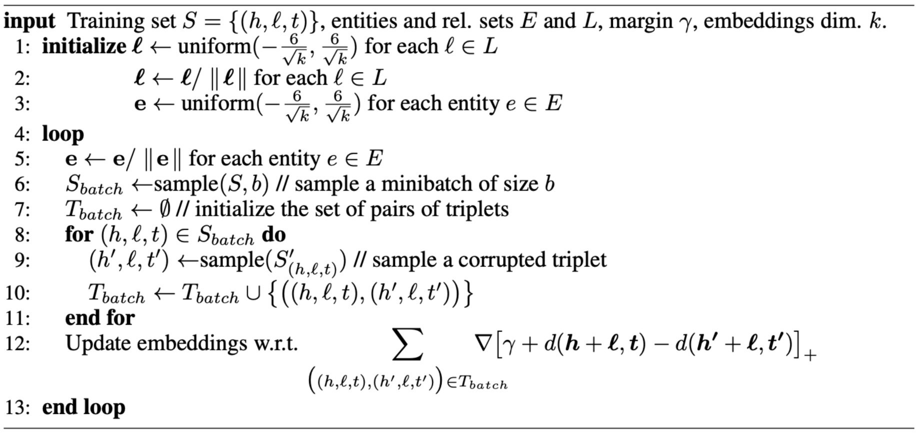
$$S'_{(h,\ell,t)} = \{(h', \ell, t) | h' \in E\} \cup \{(h, \ell, t') | t' \in E\}$$

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:

2.1 Question 2.1 (3 points)

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

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



```
input Training set $S = \{(h,\ell,t)\}$, entities and rel. sets $E$ and $L$, margin $\gamma$, embeddings dim. $k$.
 1: initialize $\boldsymbol{\ell} \leftarrow$ 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:            $\mathbf{e} \leftarrow$ 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:    $S_{batch} \leftarrow$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:       $(h',\ell,t') \leftarrow$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. $\sum_{((h,\ell,t),(h',\ell,t')) \in T_{batch}} \nabla\big[\gamma + d(\boldsymbol{h}+\boldsymbol{\ell},\boldsymbol{t}) - d(\boldsymbol{h'}+\boldsymbol{\ell},\boldsymbol{t'})\big]_+$
13: end loop
```

we would obtain a useless embedding. Give an example of a simple graph and corresponding embeddings that will minimize the new objective function 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 $\ell = \mathbf{0}$?

2.2 Question 2.2 (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

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

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 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., $k = 2$.)

2.3 Question 2.3 (7 points)

Recall that TransE normalizes every entity embedding to have unit length (see line 5 of the algorithm above). 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?

2.4 Question 2.4 (10 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.
Hint: By expressiveness of TransE embeddings, we want you to talk about which type of relationships TransE can/cannot model. As before, assume the embeddings are in 2 dimensions ($k = 2$).

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.
* Composition: A is son of B; C is sister of B, then C is aunt of A.

3.1 Question 3.1 (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.

3.2 Question 3.2 (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 $i \in 0 \dots d-1$, (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?

3.3 Question 3.3 (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.

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!