Contents

Overview

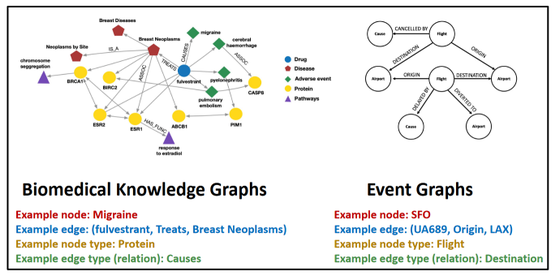
heterogeneous graphs

- **Goal**:
  - So far we only handle graphs with one edge type.
  - How to handle (directed) graphs with multiple edge types (a.k.a heterogeneous graphs)?

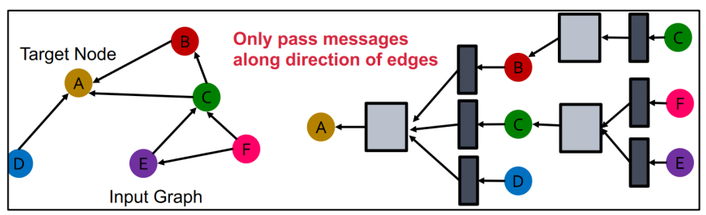
- Heterogeneous Graphs
  - Relational GCNs
  - Knowledge Graphs
  - Embeddings for KG Completion

- **A heterogeneous graph is defined as** $\boldsymbol{G = (V, E, R, T)}$
  - Nodes with node types $v_i \in V$
  - Edges with relation types $(v_i, r, v_j) \in E$
  - Node type $T(v_i)$
  - Relation type $r \in R$

Example

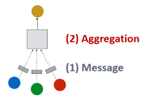


GCN



Recap

- **A single GNN layer:**
  - **(1) Message**: each node computes a message
    $$\mathbf{m}_u^{(l)} = \text{MSG}^{(l)}\left(\mathbf{h}_u^{(l-1)}\right), u \in \{N(v) \cup v\}$$
  - **(2) Aggregation**: aggregate messages from neighbors
    $$\mathbf{h}_v^{(l)} = \text{AGG}^{(l)}\left(\left\{\mathbf{m}_u^{(l)}, u \in N(v)\right\}, \mathbf{m}_v^{(l)}\right)$$
  - **Nonlinearity (activation):** Adds expressiveness
    - Often written as $\sigma(\cdot)$: ReLU$(\cdot)$, Sigmoid$(\cdot)$, ...
    - Can be added to **message or aggregation**



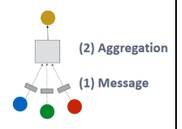
$$\mathbf{h}_v^{(l)} = \sigma\left(\mathbf{W}^{(l)} \sum_{u \in N(v)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}\right)$$

write this as Message + Aggregation

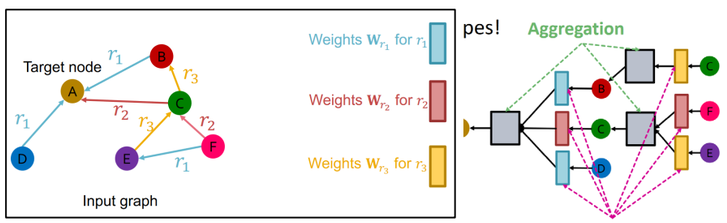
$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \mathbf{W}^{(l)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}\right)$$

Message

Aggregation



Relatinoal GCN - multiple edge/relation, different weights for different relation types



Relational GCN (RGCN) - different weights different relation

- **Relational GCN (RGCN):**

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{r \in R} \sum_{u \in N_v^r} \frac{1}{c_{v,r}} \mathbf{W}_r^{(l)} \mathbf{h}_u^{(l)} + \mathbf{W}_0^{(l)} \mathbf{h}_v^{(l)}\right)$$

- **How to write this as Message + Aggregation?**
- **Message:**
  - Each neighbor of a given relation:
    $$\mathbf{m}_{u,r}^{(l)} = \frac{1}{c_{v,r}} \mathbf{W}_r^{(l)} \mathbf{h}_u^{(l)}$$
  - Self-loop:
    $$\mathbf{m}_v^{(l)} = \mathbf{W}_0^{(l)} \mathbf{h}_v^{(l)}$$
- **Aggregation:**
  - Sum over messages from neighbors and self-loop, then apply activation
  - $\mathbf{h}_v^{(l+1)} = \sigma\left(\text{Sum}\left(\left\{\mathbf{m}_{u,r}^{(l)}, u \in N(v)\right\} \cup \left\{\mathbf{m}_v^{(l)}\right\}\right)\right)$

Normalized by node degree of the relation $c_{v,r} = |N_v^r|$

Scability - block diagonal matrices & Basis/Dictionary learning

- Each relation has L matrices: $\mathbf{W}_r^{(1)}, \mathbf{W}_r^{(2)} \cdots \mathbf{W}_r^{(L)}$
- The size of each $\mathbf{W}_r^{(l)}$ is $d^{(l+1)} \times d^{(l)}$ — $d^{(l)}$ is the hidden dimension in layer l
- **Rapid # parameters growth w.r.t # relations!**
  - **Overfitting becomes an issue**
- **Two methods to regularize the weights $\mathbf{W}_r^{(l)}$**
  - **(1)** Use block diagonal matrices
  - **(2)** Basis/Dictionary learning

block diagonal matrices - sparse & Limitation

- **Key insight**: make the weights **sparse**!
- Use **block diagonal matrices** for $\mathbf{W}_r$

$$\mathbf{W}_r = \begin{bmatrix} \square & & \\ & \square & \\ & & \square \end{bmatrix}$$

Limitation: only nearby neurons/dimensions can interact through W

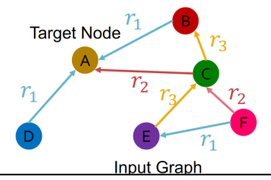
- If use B low-dimensional matrices, then # param reduces from $d^{(l+1)} \times d^{(l)}$ to $B \times \frac{d^{(l+1)}}{B} \times \frac{d^{(l)}}{B}$.

Basis learning – weigth share, linear combination of basis transformations

- **Key insight**: **Share weights** across different relations!
- Represent the matrix of each relation as a linear combination of basis transformations
  $\mathbf{W}_r = \sum_{b=1}^{B} a_{rb} \cdot \mathbf{V}_b$, where $\mathbf{V}_b$ is shared across all relations
  - $\mathbf{V}_b$ are the basis matrices
  - a_{rb} is the importance weight of matrix $\mathbf{V}_b$
- Now each relation only needs to learn $\{a_{rb}\}_{b=1}^{B}$, which is B scalars.

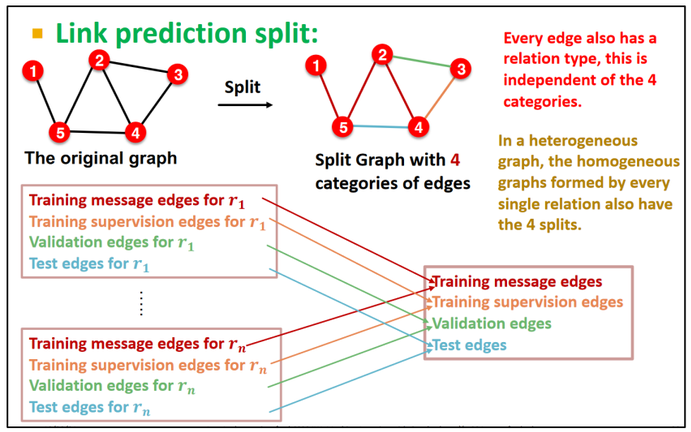
Entity / node classification

- **Goal**: **Predict the label of a given node**
- **RGCN** uses the representation of the final layer:
  - If we predict the class of **node A** from **k classes**.
  - Take the **final layer (prediction head)**: $\mathbf{h}_A^{(L)} \in \mathbb{R}^k$, each item in $\mathbf{h}_A^{(L)}$ represents **the probability of that class**.



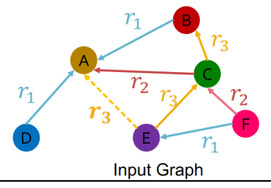
Link prediction

- Split edge based on relation type because some are rare but important so split within each relation type to emphaise same importance
- If random split, then rare type may not appear
- Split & merge



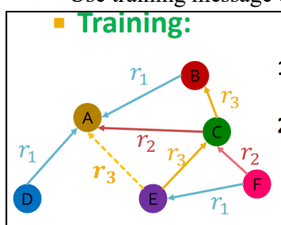
example

- Assume (E, r_3, A) is training supervision edge, all the other edges are training message edges
- Use RGCN to score (E, r_3, A)!
  - Take the final layer of E and A: $\mathbf{h}_E^{(L)}$ and $\mathbf{h}_A^{(L)} \in \mathbb{R}^d$
  - Relation-specific score function f_r: $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$
    - One example $f_{r_1}(\mathbf{h}_E, \mathbf{h}_A) = \mathbf{h}_E^T \mathbf{W}_{r_1} \mathbf{h}_A$, $\mathbf{W}_{r_1} \in \mathbb{R}^{d \times d}$

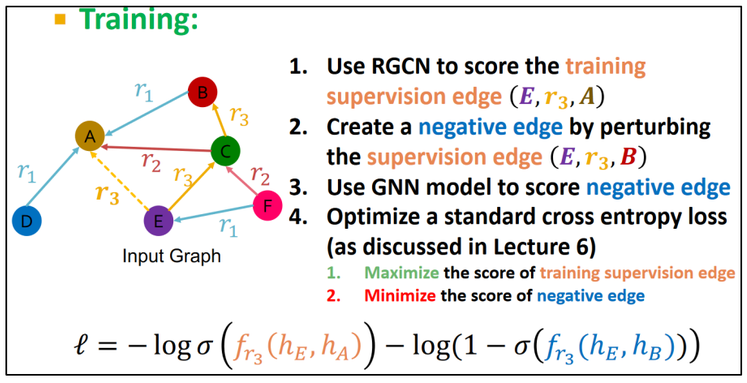


Negative edge

- Use training message edge to predict training supervision edge

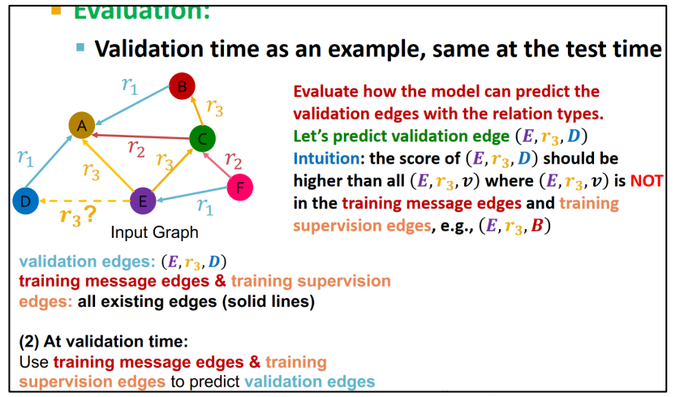


Loss fn – maximise positive & mimise negative samples



Evaluation split

- Goal. The score of connected edge is higher than those negative / don't exist



Steps & metrics: hits / reciprocal rank

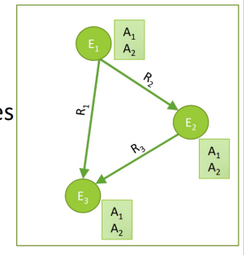
1. Calculate the score of (E, r_3, D)
2. Calculate the score of all the negative edges: $\{(E, r_3, v) | v \in \{B, F\}\}$, since (E, r_3, A), (E, r_3, C) belong to training message edges & training supervision edges
3. Obtain the ranking RK of (E, r_3, D).
4. Calculate metrics
   1. Hits@k: $\mathbf{1}[RK \leq k]$. Higher is better
   2. Reciprocal Rank: $\frac{1}{RK}$. Higher is better

Knowldege graph (KG)

graph form: entities, type, relation

Knowledge in graph form:

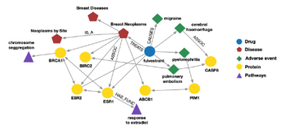
- Capture entities, types, and relationships
- Nodes are **entities**
- Nodes are labeled with their **types**
- Edges between two nodes capture **relationships** between entities
- **KG is an example of a heterogeneous graph**



Example

- **Node types**: paper, title, author, conference, year
- **Relation types**: pubWhere, pubYear, hasTitle, hasAuthor, cite

- **Node types**: drug, disease, adverse event, protein, pathways
- **Relation types**: has_func, causes, assoc, treats, is_a



application

- To capture domain and background information
- Question answering and conversation agents

Examples of knowledge graphs
- Google Knowledge Graph
- Amazon Product Graph
- Facebook Graph API
- IBM Watson
- Microsoft Satori
- Project Hanover/Literome
- LinkedIn Knowledge Graph
- Yandex Object Answer

Knowledge graph dataset

- **Publicly available KGs:**
  - FreeBase, Wikidata, Dbpedia, YAGO, NELL, etc.
- **Common characteristics:**
  - **Massive**: millions of nodes and edges
  - **Incomplete**: many true edges are missing

Given a massive KG, enumerating all the possible facts is intractable! ⇨ Can we predict plausible BUT missing links?

Example: freebase

- **Freebase**
  - ~80 million **entities**
  - ~38K **relation types**
  - ~3 billion **facts/triples**

Freebase

93.8% of persons from Freebase have no place of birth and 78.5% have no nationality!

- **Datasets:** FB15k/FB15k-237
  - A **complete** subset of Freebase, used by researchers to learn KG models

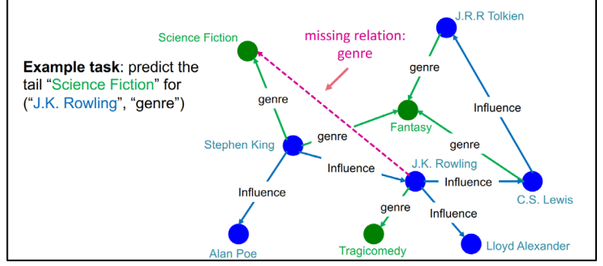
Dataset	Entities	Relations	Total Edges
FB15k	14,951	1,345	592,213
FB15k-237	14,505	237	310,079

[1] Paulheim, Heiko. "Knowledge graph refinement: A survey of approaches and evaluation methods." *Semantic web* 8.3 (2017): 489-508.
[2] Mintz, Bonan, et al. "Distant supervision for relation extraction with an incomplete knowledge base." *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2013.

KG completion task

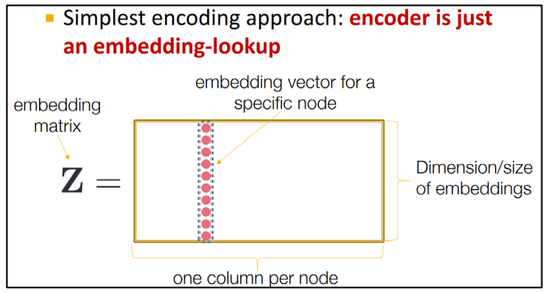
- E.g Head – obamama , relation – nationality, tail – US

Given an enormous KG, can we complete the KG?
- For a given (**head**, **relation**), we predict missing **tails**.
  - (Note this is slightly different from link prediction task)



Method: Shallow encoding

- Simplest encoding approach: **encoder is just an embedding-lookup**



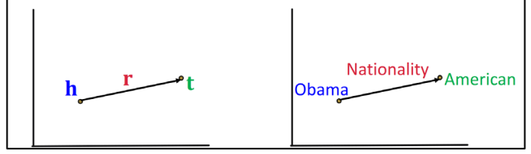
KG representation

- No GNN for simplicty

- Edges in KG are represented as **triples** (h, r, t)
  - head (h) has relation (r) with tail (t)
- **Key Idea:**
  - Model entities and relations in the embedding/vector space $\mathbb{R}^d$.
    - Associate entities and relations with **shallow embeddings**
    - **Note we do not learn a GNN here!**
  - Given a true triple (h, r, t), the goal is that the embedding of (h, r) **should be close** to the embedding of t.
    - How to embed (h, r)?
    - How to define closeness?

transE - embedding h + r = t

- **Translation Intuition**:
  For a triple (h, r, t), $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^d$, embedding vectors will appear in boldface
  $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ if the given fact is true
  else $\mathbf{h} + \mathbf{r} \neq \mathbf{t}$
  Scoring function: $f_r(h, t) = -||\mathbf{h} + \mathbf{r} - \mathbf{t}||$



Algo – TransE, contrastive loss

Algorithm 1 Learning TransE

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

initialize $\boldsymbol{\ell} \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each $\ell \in L$
$\boldsymbol{\ell} \leftarrow \boldsymbol{\ell} / \|\boldsymbol{\ell}\|$ for each $\ell \in L$
$\mathbf{e} \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each entity $e \in E$

Entities and relations are initialized uniformly, and normalized

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. $\sum_{((h,\ell,t),(h',\ell,t')) \in T_{batch}} \nabla \left[\gamma + d(\boldsymbol{h} + \boldsymbol{\ell}, \boldsymbol{t}) - d(\boldsymbol{h'} + \boldsymbol{\ell}, \boldsymbol{t'})\right]_+$
end loop

Negative sampling with triplet that does not appear in the KG

d represents distance (negative of score)

positive sample / negative sample

Contrastive loss: favors lower distance (or higher score) for valid triplets, high distance (or lower score) for corrupted ones

Connectivity pattern in KG: mentor & mentee

- **Relations in a heterogeneous KG have different properties**
  - Example:
    - **Symmetry:** If the edge $(h,$ "Roommate", $t)$ exists in KG, then the edge $(t,$ "Roommate", $h)$ should also exist.
    - **Inverse relation:** If the edge $(h,$ "Advisor", $t)$ exists in KG, then the edge $(t,$ "Advisee", $h)$ should also exist.
- Can we **categorize** these relation patterns?
- **Are KG embedding methods (e.g., TransE) expressive enough to model these patterns?**

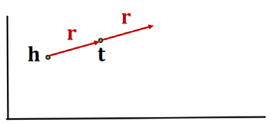
Relation pattern – symmetric, inverse, composition, 1-to-N

- **Symmetric (Antisymmetric) Relations:**
  $$r(h,t) \Rightarrow r(t,h) \quad (r(h,t) \Rightarrow \neg r(t,h)) \quad \forall h,t$$
  - **Example**:
    - Symmetric: Family, Roommate
    - Antisymmetric: Hypernym
- **Inverse Relations:**
  $$r_2(h,t) \Rightarrow r_1(t,h)$$
  - **Example** : (Advisor, Advisee)
- **Composition (Transitive) Relations:**
  $$r_1(x,y) \wedge r_2(y,z) \Rightarrow r_3(x,z) \quad \forall x,y,z$$
  - **Example**: My mother's husband is my father.
- **1-to-N relations:**
  $r(h,t_1), r(h,t_2), \ldots, r(h,t_n)$ are all True.
  - **Example**: r is "StudentsOf"

M1: TransE: applicable & limitation

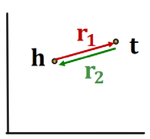
Antisymmetric

- **Antisymmetric Relations:**
  $$r(h,t) \Rightarrow \neg r(t,h) \quad \forall h,t$$
  - **Example**: Hypernym
- **TransE** can model antisymmetric relations ✓
  - $\mathbf{h} + \mathbf{r} = \mathbf{t}$, but $\mathbf{t} + \mathbf{r} \neq \mathbf{h}$



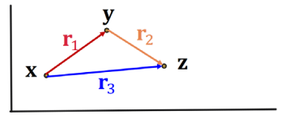
Inverse

- **Inverse Relations:**
  $$r_2(h,t) \Rightarrow r_1(t,h)$$
  - **Example** : (Advisor, Advisee)
- **TransE** can model inverse relations ✓
  - $\mathbf{h} + \mathbf{r}_2 = \mathbf{t}$, we can set $\mathbf{r}_1 = -\mathbf{r}_2$



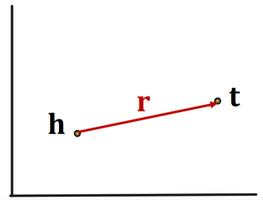
Composition (Transitive)

- **Composition (Transitive) Relations:**
  $$r_1(x,y) \wedge r_2(y,z) \Rightarrow r_3(x,z) \quad \forall x,y,z$$
  - **Example**: My mother's husband is my father.
- **TransE** can model composition relations ✓
  $$\mathbf{r}_3 = \mathbf{r}_1 + \mathbf{r}_2$$



Limitation – Symmetric, only if r = 0, but overlap

- **Symmetric Relations:**
  $$r(h,t) \Rightarrow r(t,h) \quad \forall h,t$$
  - **Example**: Family, Roommate
- **TransE cannot** model symmetric relations ✗
  only if $\mathbf{r} = 0,\ \mathbf{h} = \mathbf{t}$

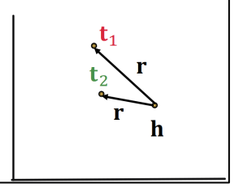


For all h, t that satisfy $r(h,t)$, $r(t,h)$ is also True, which means $\|\mathbf{h}+\mathbf{r}-\mathbf{t}\| = 0$ and $\|\mathbf{t}+\mathbf{r}-\mathbf{h}\| = 0$. Then $\mathbf{r} = 0$ and $\mathbf{h} = \mathbf{t}$, however h and t are two different entities and should be mapped to different locations.

Limitation - 1-to-N

- **1-to-N Relations:**
  - **Example**: (h, r, t_1) and (h, r, t_2) both exist in the knowledge graph, e.g., r is "StudentsOf"
- **TransE cannot** model 1-to-N relations ✗
  - $\mathbf{t}_1$ and $\mathbf{t}_2$ will map to the same vector, although they are different entities

- $\mathbf{t}_1 = \mathbf{h} + \mathbf{r} = \mathbf{t}_2$
- $\mathbf{t}_1 \neq \mathbf{t}_2$ contradictory!



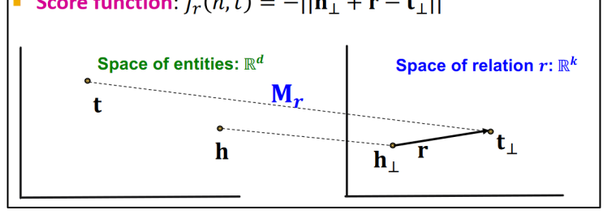
M2: transR – projection matrix + transE

relation-specific embedding space

- **TransE models translation of any relation in the same embedding space.**
- **Can we design a new space for each relation and do translation in relation-specific space?**
- **TransR**: model **entities** as vectors in the entity space $\mathbb{R}^d$ and model each **relation** as vector in relation space $\mathbf{r} \in \mathbb{R}^k$ with $\mathbf{M}_r \in \mathbb{R}^{k \times d}$ as the projection matrix.

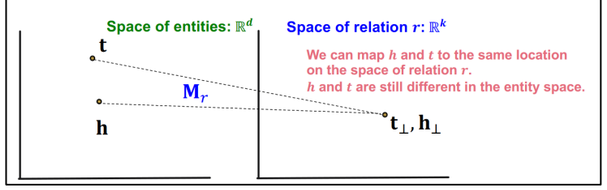
Projection matrix

- **TransR**: model **entities** as vectors in the entity space $\mathbb{R}^d$ and model each **relation** as vector in relation space $\mathbf{r} \in \mathbb{R}^k$ with $\mathbf{M}_r \in \mathbb{R}^{k \times d}$ as the **projection matrix**.
  Use $\mathbf{M}_r$ to project from entity space $\mathbb{R}^d$ to relation space $\mathbb{R}^k$!
- $\mathbf{h}_\perp = \mathbf{M}_r\mathbf{h},\ \mathbf{t}_\perp = \mathbf{M}_r\mathbf{t}$
- **Score function**: $f_r(h,t) = -\|\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp\|$



Symmetric

- **Symmetric Relations:**
  $$r(h,t) \Rightarrow r(t,h) \quad \forall h,t$$
  Note different symmetric relations may have different $\mathbf{M}_r$
  - **Example**: Family, Roommate
- **TransR** can model symmetric relations
  $$\mathbf{r} = 0,\ \mathbf{h}_\perp = \mathbf{M}_r\mathbf{h} = \mathbf{M}_r\mathbf{t} = \mathbf{t}_\perp$$ ✓



We can map h and t to the same location on the space of relation r.
h and t are still different in the entity space.

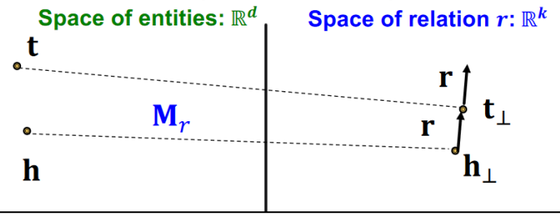
Antisymmetric

- **Antisymmetric Relations:**

$$r(h,t) \Rightarrow \neg r(t,h) \quad \forall h,t$$

  - **Example**: Hypernym
- **TransR** can model antisymmetric relations

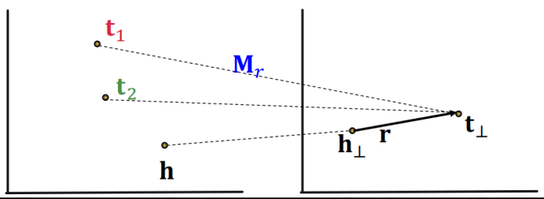
$$\mathbf{r} \neq 0, \mathbf{M}_r\mathbf{h} + \mathbf{r} = \mathbf{M}_r\mathbf{t},$$
$$\text{Then } \mathbf{M}_r\mathbf{t} + \mathbf{r} \neq \mathbf{M}_r\mathbf{h} \checkmark$$



1-to-N

- Project s.t. the new space they are the same

- **1-to-N Relations:**
  - **Example**: If (h,r,t_1) and (h,r,t_2) exist in the knowledge graph.
- **TransR** can model 1-to-N relations ✓
  - We can learn $\mathbf{M}_r$ so that $\mathbf{t}_\perp = \mathbf{M}_r\mathbf{t}_1 = \mathbf{M}_r\mathbf{t}_2$
  - Note that $\mathbf{t}_1$ does not need to be equal to $\mathbf{t}_2$!



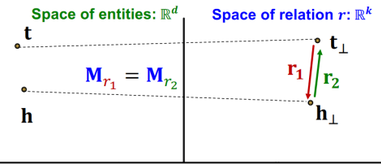
Inverse

- **Inverse Relations:**

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

  - **Example** : (Advisor, Advisee)
- **TransR** can model inverse relations

$$\mathbf{r}_2 = -\mathbf{r}_1, \mathbf{M}_{r_1} = \mathbf{M}_{r_2}$$
$$\text{Then } \mathbf{M}_{r_1}\mathbf{t} + \mathbf{r}_1 = \mathbf{M}_{r_1}\mathbf{h} \text{ and } \mathbf{M}_{r_2}\mathbf{h} + \mathbf{r}_2 = \mathbf{M}_{r_2}\mathbf{t} \checkmark$$



Limitation - Composition Relations – bc different relation spaces

- **Composition Relations:**

$$r_1(x,y) \wedge r_2(y,z) \Rightarrow r_3(x,z) \quad \forall x,y,z$$

  - **Example**: My mother's husband is my father.
- **TransR cannot** model composition relations

Each relation has a different space.
It is **not naturally compositional** for multiple relations! ×

Zero Distance

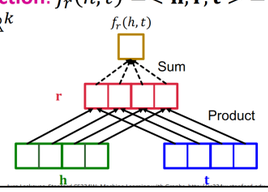
Intuition: The manifold $\{z|\exists y, f_{r_1}(x,y)=0, f_{r_2}(y,z)=0\}$ is **high dimensional** and may not be modeled using a single $\mathbf{M}_{r_3}$ and $\mathbf{r}_3$

M3: bilinear modeling

Different scoring function: multi over the distance

- Distance: H + R = T

- **So far:** The scoring function $f_r(h,t)$ is **negative of L1 / L2 distance** in **TransE** and **TransR**
- Another line of KG embeddings adopt **bilinear** modeling
- **DistMult**: Entities and relations using vectors in $\mathbb{R}^k$
- **Score function**: $f_r(h,t) =< \mathbf{h},\mathbf{r},\mathbf{t} >= \sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \mathbf{t}_i$
- $\mathbf{h},\mathbf{r},\mathbf{t} \in \mathbb{R}^k$

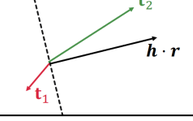


DisMult: Hypyer plane

- The coordinate is the $h \cdot r$
- Measure angle between $h \cdot r$ & t
- If dot product is positive, then fall in the LHS, otherwise, RHS

- **DistMult**: Entities and relations using vectors in $\mathbb{R}^k$
- **Score function**: $f_r(h,t) =< \mathbf{h},\mathbf{r},\mathbf{t} >= \sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \mathbf{t}_i$
  - $\mathbf{h},\mathbf{r},\mathbf{t} \in \mathbb{R}^k$
- **Intuition of the score function:** Can be viewed as a **cosine similarity** between $\mathbf{h} \cdot \mathbf{r}$ and $\mathbf{t}$
  - where $\mathbf{h} \cdot \mathbf{r}$ is defined as $\sum_i h_i \cdot r_i$
- **Example:**

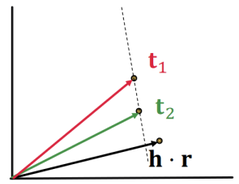
$$f_r(h,t_1) < 0, \quad f_r(h,t_2) > 0$$



1-to-N – fall in the same side

- **1-to-N Relations:**
  - **Example**: If (h,r,t_1) and (h,r,t_2) exist in the knowledge graph
- **Distmult** can model 1-to-N relations ✓

$$< \mathbf{h},\mathbf{r},\mathbf{t}_1 >=< \mathbf{h},\mathbf{r},\mathbf{t}_2 >$$



Symmetric - Summation & multiplication naturally communitive

- **Symmetric Relations:**

$$r(h,t) \Rightarrow r(t,h) \quad \forall h,t$$

  - **Example**: Family, Roommate
- **DistMult** can naturally model symmetric relations ✓

$$f_r(h,t) =< \mathbf{h},\mathbf{r},\mathbf{t} >= \sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \mathbf{t}_i =$$
$$< \mathbf{t},\mathbf{r},\mathbf{h} >= f_r(t,h)$$

Limitation

antisymmetric: opposite symmetric

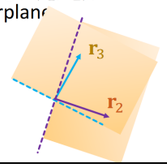
- **Antisymmetric Relations:**
  $$r(h,t) \Rightarrow \neg r(t,h) \quad \forall h,t$$
  - **Example**: Hypernym
- **DistMult cannot** model antisymmetric relations
  $f_r(h,t) =< \mathbf{h},\mathbf{r},\mathbf{t} > = < \mathbf{t},\mathbf{r},\mathbf{h} >= f_r(t,h)$ ×
  - $r(h,t)$ and $r(t,h)$ always have same score!

Inverse

- **Inverse Relations:**
  $$r_2(h,t) \Rightarrow r_1(t,h)$$
  - **Example** : (Advisor, Advisee)
- **DistMult cannot** model inverse relations ×
  - If it does model inverse relations:
    $f_{r_2}(h,t) =< \mathbf{h},\mathbf{r}_2,\mathbf{t} > = < \mathbf{t},\mathbf{r}_1,\mathbf{h} >= f_{r_1}(t,h)$
  - This means $\mathbf{r}_2 = \mathbf{r}_1$
  - But semantically this does not make sense: **The embedding of "Advisor" should not be the same with "Advisee".**

Composition – different hyper planes

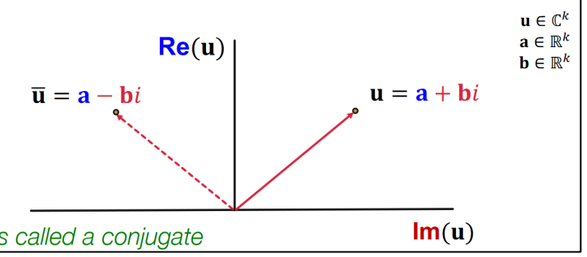
- **Composition Relations:**
  $$r_1(x,y) \wedge r_2(y,z) \Rightarrow r_3(x,z) \quad \forall x,y,z$$
  - **Example**: My mother's husband is my father.
- **DistMult cannot** model composition relations ×
  - **Intuition: DistMult** defines a hyperplane for each (head, relation), the union of the hyperplane induced by multi-hops of relations, e.g., (r_1, r_2), cannot be expressed using a single hyperplan



comlEx – complex plane embedding

conjugate

- Based on Distmult, **ComplEx** embeds entities and relations in **Complex vector space**
- **ComplEx**: model entities and relations using vectors in $\mathbb{C}^k$



Score function

- Based on Distmult, **ComplEx** embeds entities and relations in **Complex vector space**
- **ComplEx**: model entities and relations using vectors in $\mathbb{C}^k$
- **Score function** $f_r(h,t) = \mathrm{Re}(\sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \bar{\mathbf{t}}_i)$

Table – expressiveness of all models

Model	Score	Embedding	Sym.	Antisym.	Inv.	Compos.	1-to-N
TransE	$-\|\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|\|$	$\mathbf{h},\mathbf{t},\mathbf{r} \in \mathbb{R}^k$	×	✓	✓	✓	×
TransR	$-\|\|\boldsymbol{M}_r\mathbf{h}+\mathbf{r}-\boldsymbol{M}_r\mathbf{t}\|\|$	$\mathbf{h},\mathbf{t},\mathbf{r} \in \mathbb{R}^k$, $\boldsymbol{M}_r \in \mathbb{R}^{d\times k}$	✓	✓	✓	✓	✓
DistMult	$< \mathbf{h},\mathbf{r},\mathbf{t} >$	$\mathbf{h},\mathbf{t},\mathbf{r} \in \mathbb{R}^k$	✓	×	×	×	✓
ComplEx	$\mathrm{Re}(< \mathbf{h},\mathbf{r},\bar{\mathbf{t}} >)$	$\mathbf{h},\mathbf{t},\mathbf{r} \in \mathbb{C}^k$	✓	✓	✓	×	✓

summary

1. Different KGs may have **drastically different relation patterns**!
2. There is not a general embedding that works for all KGs, use the **table** to select models
3. Try **TransE** for a quick run if the target KG does not have much symmetric relations
4. Then use more expressive models, e.g., **ComplEx**, **RotatE** (**TransE** in Complex space)

Summary

GCN

- Since aggregation involves sum and node degree average so similar to – average pooling
- Non-relational
  - Weight matrix same for each layer
- Relational / hetergious
  - Depends on the edge type
  - i.e. for every layer for every relation

block diagonal matrices?

supervision edges (i.e., used to compute loss) during training

- Message edges: Used for GNN message passing
- Supervision edges: Used for computing objectives

Other notes:

- https://www.cnblogs.com/veager/articles/16485260.html

shallow encoding

- Learn embedding for each node
- Can use GNN as well

Hich menthods to choose depends on

- What relation to model & predict