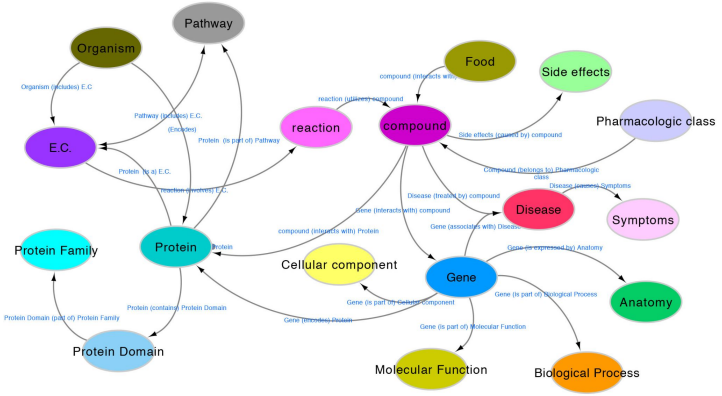


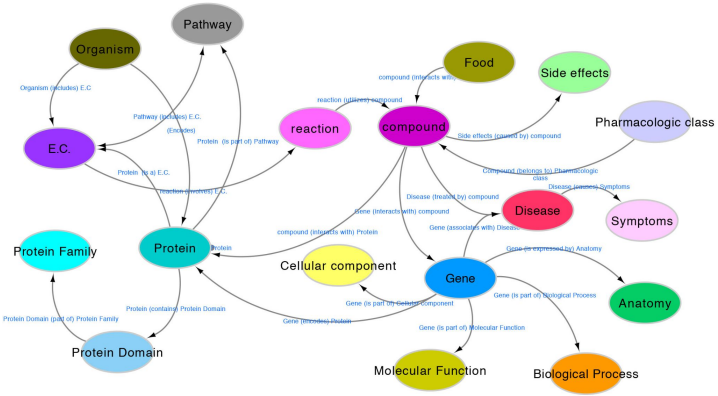
Relational Graph Networks, Knowledge Graphs, Knowledge Graph Embeddings

"*Everything connects to everything else*" - Leonardo DaVinci

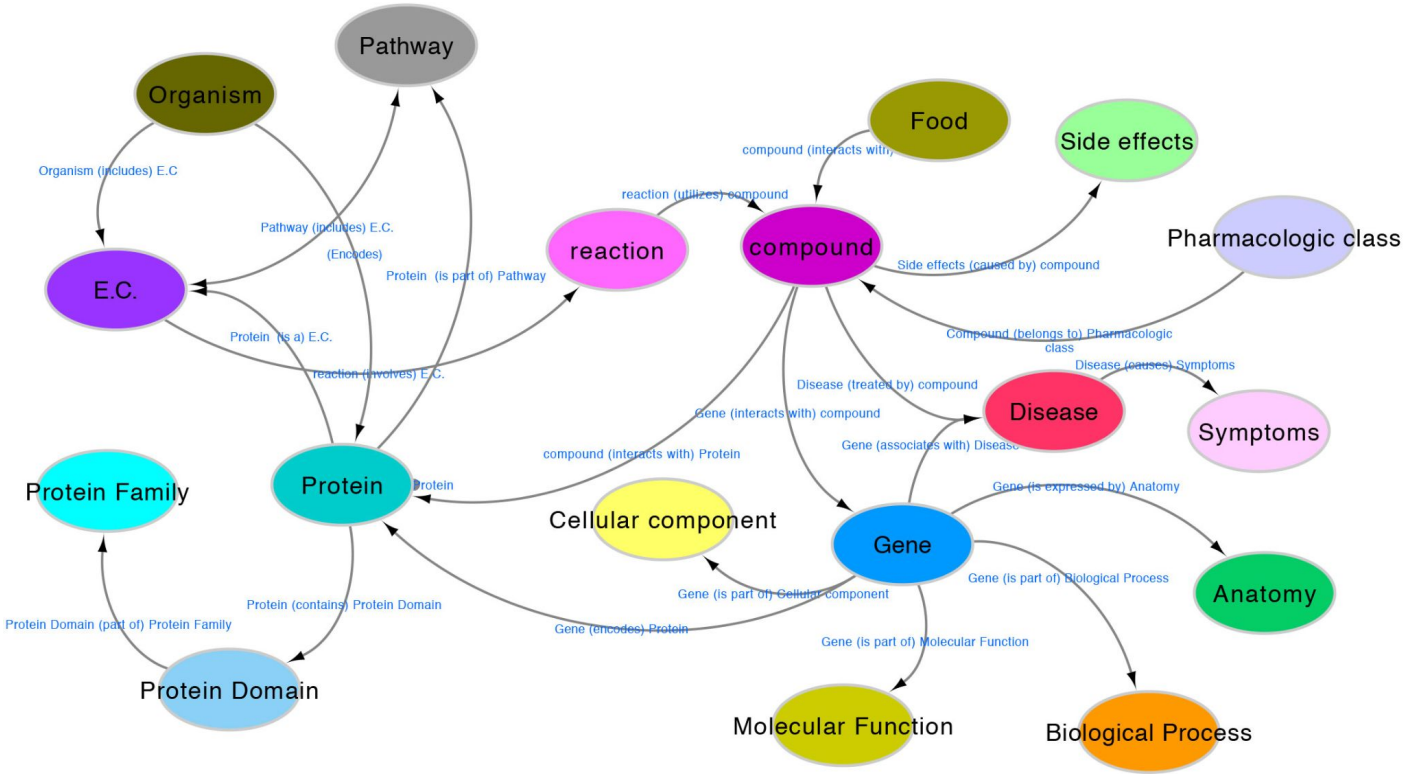
Jay Urbain, PhD - 9/27/2022

Deep Graph Search, Alignment and Knowledge Discovery

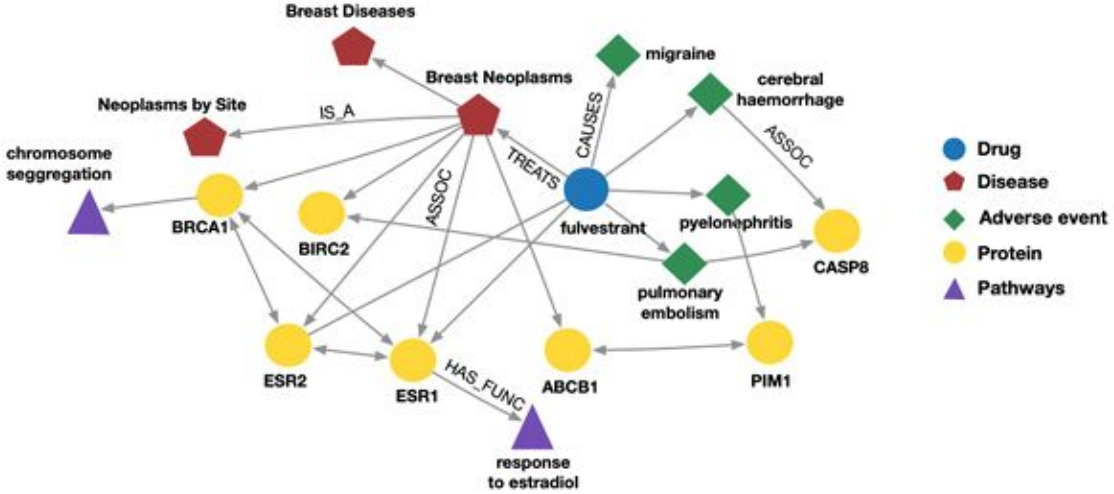
- People are overwhelmed with trying to extract actionable knowledge from large repositories of structured and unstructured data.
- Knowledge graphs are a way of representing information that can capture complex relationships more easily than conventional databases.
- Graphs can represent various complex systems: from molecular structure and biomedical knowledge to social and traffic networks.

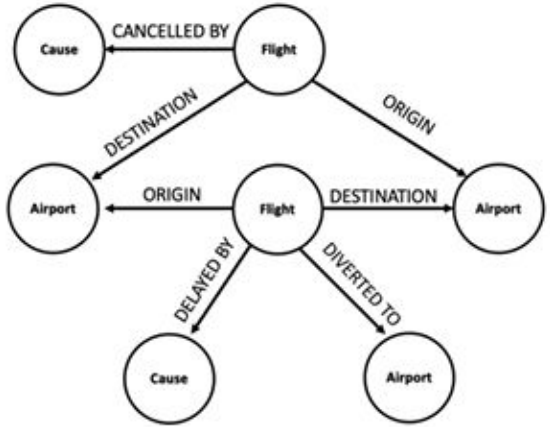


Deep Graph Search, Alignment and Knowledge Discovery



Schlichtkrull et al., Modeling Relational Data with Graph Convolutional Networks , ESWC 2018





Biomedical Knowledge Graphs

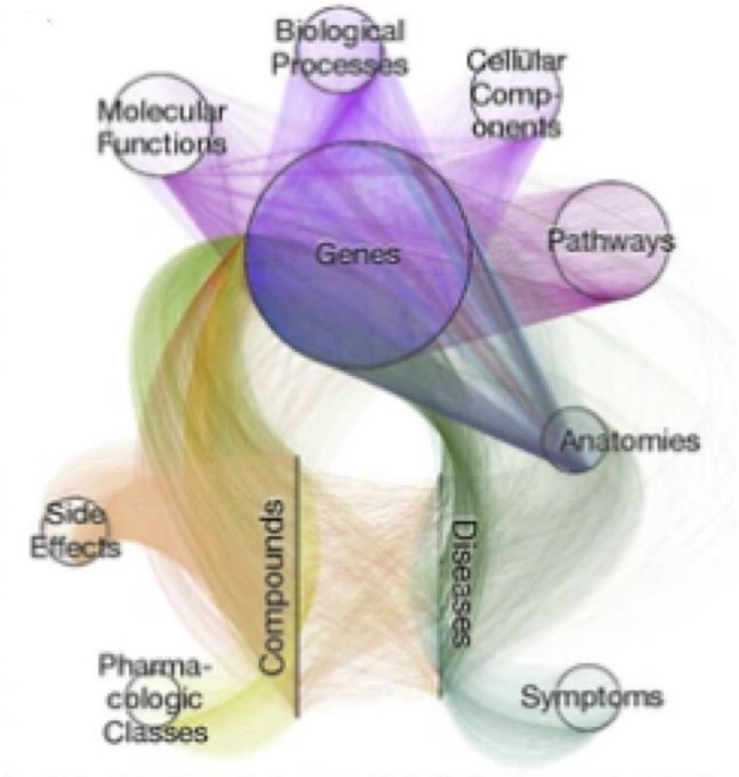
Example node: Migraine
Example edge: (fulvestrant, Treats, Breast Neoplasms)
Example node type: Protein
Example edge type (relation): Causes

Event Graphs

Example node: SFO
Example edge: (UA689, Origin, LAX)
Example node type: Flight
Example edge type (relation): Destination

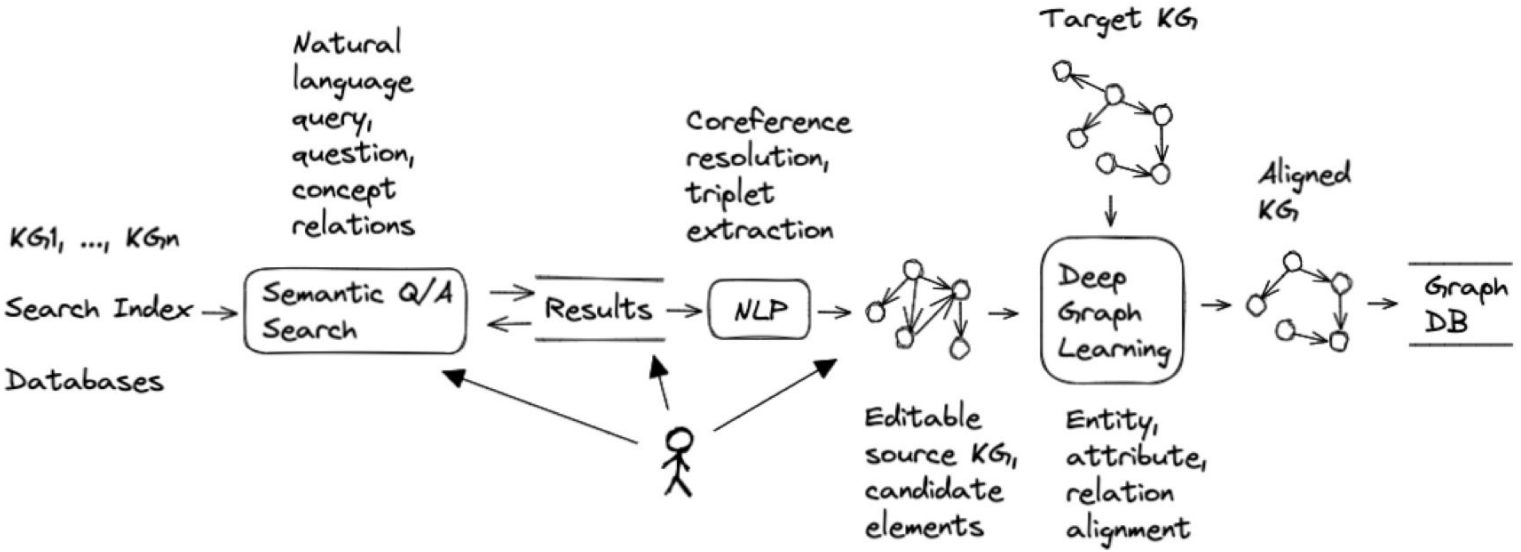
Deep Graph Search, Alignment and Knowledge Discovery

- An important task involves integrating one KG with another KG, other data sources, or search results.
- But different graphs and data sources may use different terms for the same entities, attributes, and relations, which can lead to errors and inconsistencies.
- Hence the need for more automated techniques of entity, attribute, and relation alignment for determining which elements of different graphs refer to the same entities, attributes, and relations.



Example: Knowledge Graph Processing Pipeline

- Integration and knowledge discovery via search, user relevance feedback, and user feedback on intermediary graph networks.



Heterogeneous Graphs

How to handle (directed) graphs with multiple edge types (a.k.a heterogeneous graphs)?

Heterogeneous Graphs

- **Relational GCNs - extend GCN to handle nodes and relations of different types**
- Knowledge Graphs
- Embeddings for KG Completion

Review: Single GNN Layer

(1) Message: each node computes a message

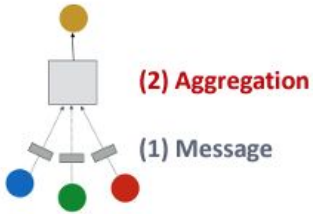
$$\mathbf{m}_u^{(l)} = \mathrm{MSG}^{(l)}\left(\mathbf{h}_u^{(l-1)}\right), u \in \{N(v) \cup v\}$$

(2) Aggregation: aggregate messages from neighbors

$$\mathbf{h}_v^{(l)} = \mathrm{AGG}^{(l)}\left(\left\{\mathbf{m}_u^{(l)}, u \in N(v)\right\}, \mathbf{m}_v^{(l)}\right)$$

Nonlinearity (activation): Adds expressiveness

- Often written as $\sigma(\cdot)$: $\mathrm{ReLU}(\cdot)$, $\mathrm{Sigmoid}(\cdot)$, ...
- Can be added to **message or aggregation**



Review: Classical GNN Layer

Graph Convolutional Networks (GCN)

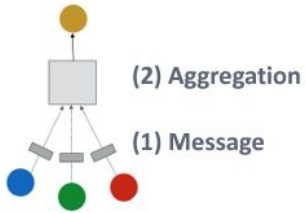
$$\mathbf{h}_v^{(l)} = \sigma\left(\mathbf{W}^{(l)} \sum_{u \in N(v)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}\right)$$

Write 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



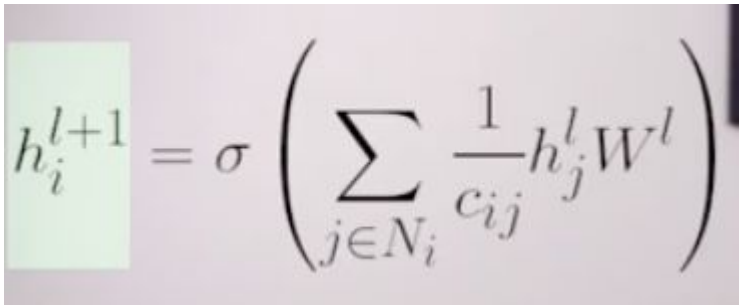
Summary: Graph Convolutional Networks

Collect neighbor embeddings

Aggregate embedding

Pass to NN - multiply by weight matrix and apply activation

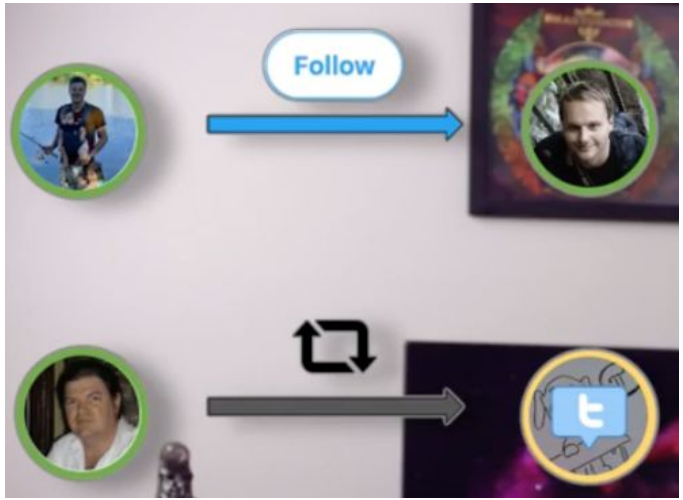
Only one weight per layer that shared by all the nodes



Knowledge Graph

Built using triples: (subject, predicate, object)

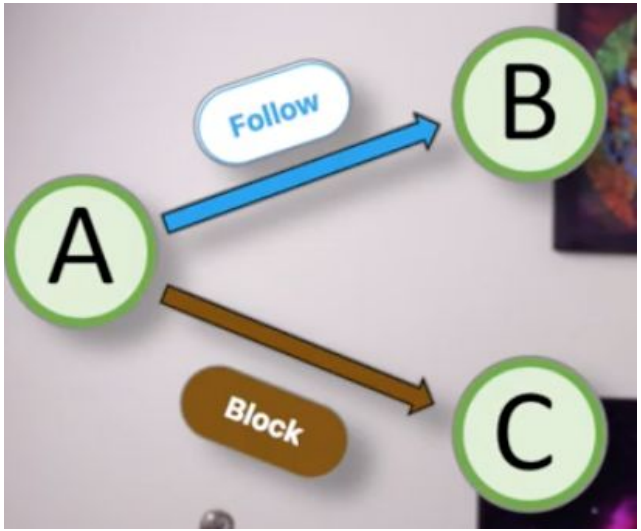
Example: Two different types of relations in one simple graph



Typical GCN

In GCN aggregate all neighbors together

May not make sense to average node messages together:



Heterogeneous Graphs

A heterogeneous graph is defined as

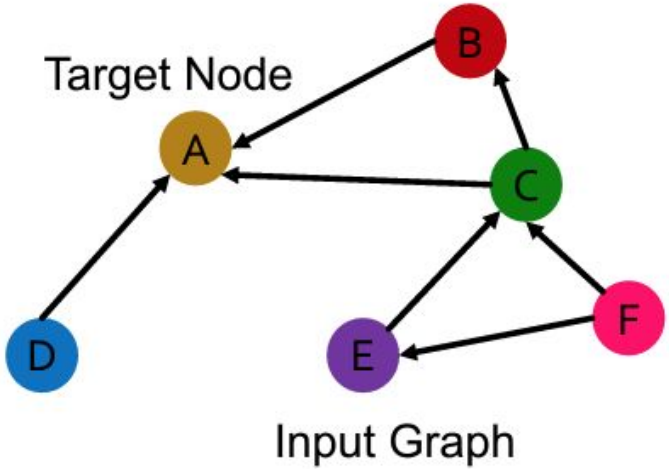
$$\boldsymbol{G = (V, E, R, T)}$$

- 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 $r \in R$

Relational GCN

Extend GCN to handle heterogeneous graphs with multiple edge/relation types.

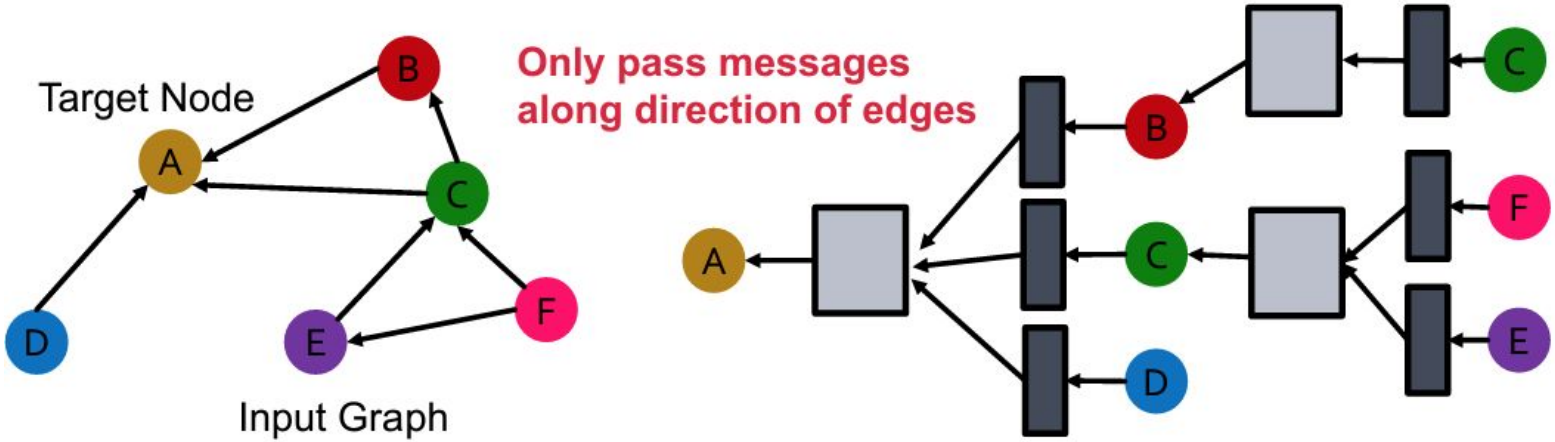
- Start with a directed graph with one relation
- How do we run GCN and update the representation of the target node A on this graph?



Relational GCN

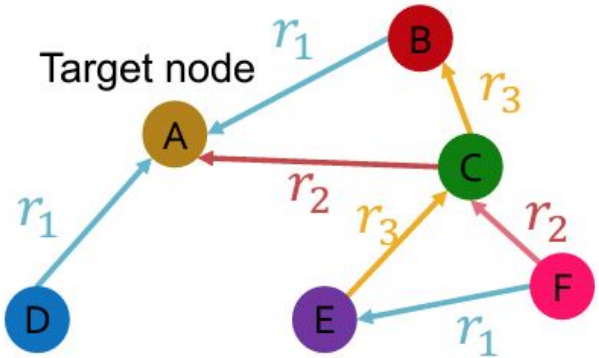
Extend GCN to handle heterogeneous graphs with multiple edge/relation types.

- Start with a directed graph with one relation
- How do we run GCN and update the representation of the target node A on this graph?



Relational GCN

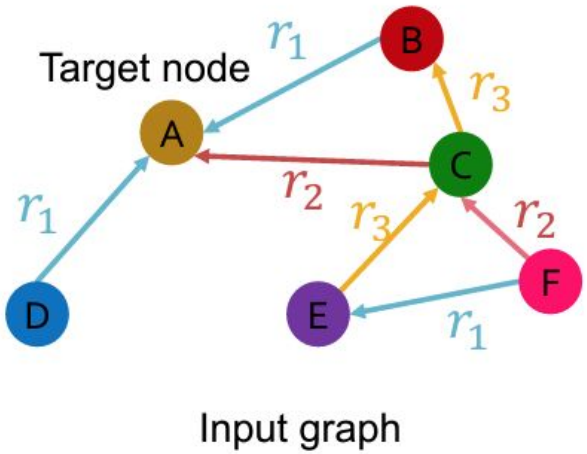
What if the graph has multiple types?



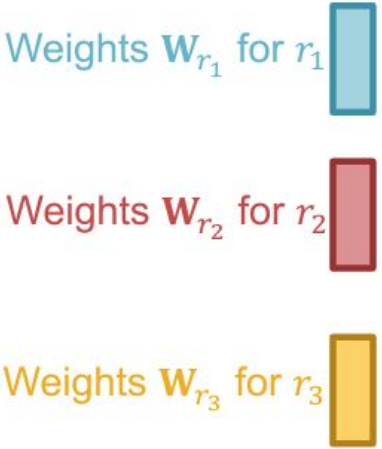
Input graph

Relational GCN

What if the graph has multiple *relation* types? Assign different weights for each type per layer!



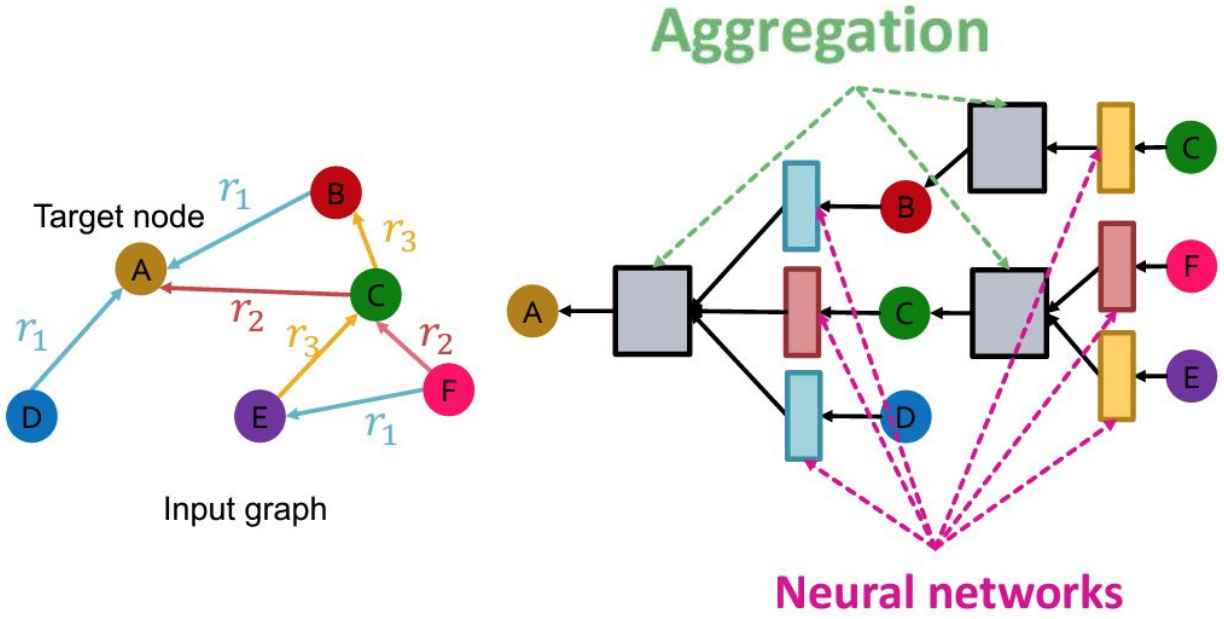
Input graph



Relational GCN

What if the graph has multiple *relation* types?

- Use different neural network weights for different relation types!



Relational GCN (RGCN)

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

Write as message + aggregation

- Message: Each neighbor of a given relation:

$$\mathbf{m}_{u,r}^{(l)} = \frac{1}{c_{v,r}}\mathbf{W}_r^{(l)}\mathbf{h}_u^{(l)}$$

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

Self-loop:

$$\mathbf{m}_v^{(l)} = \mathbf{W}_0^{(l)}\mathbf{h}_v^{(l)}$$

Relational GCN (RGCN)

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

Write as message + aggregation

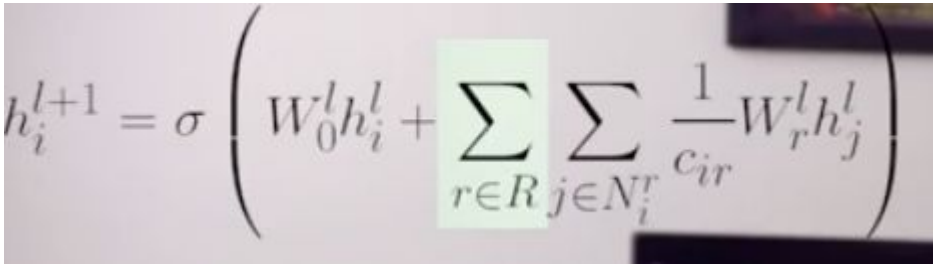
- Aggregation:

Sum over messages from neighbors and self-loop, then apply activation

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\text{Sum}\left(\left\{\mathbf{m}_{u,r}^{(l)}, u \in N(v)\right\} \cup \left\{\mathbf{m}_v^{(l)}\right\}\right)\right)$$

RGCN

- Has different W matrix for each type of relation/triple.
- Aggregate neighbors per relation type.
- W0 gives special attention to self-connections.
- Different projection matrices puts messages from different relations into same space. Person a follows person b may cancel out person b blocks person a.



Relational GCN Scalability

- Each relation has L matrices: $\mathbf{W}_r^{(1)}, \mathbf{W}_r^{(2)} \cdots \mathbf{W}_r^{(L)}$
- The size of each ***W*** 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
  - Use sparse matrices - virtually increase dimensions
  - Share weights across different relations using linear basis transformations - combine share basis matrix with local importance weight.

Basis decomposition

RGCN’s have a tendency to have a lot of parameters, since each relation has its own W per layer.

Problem especially if relation type is rare.

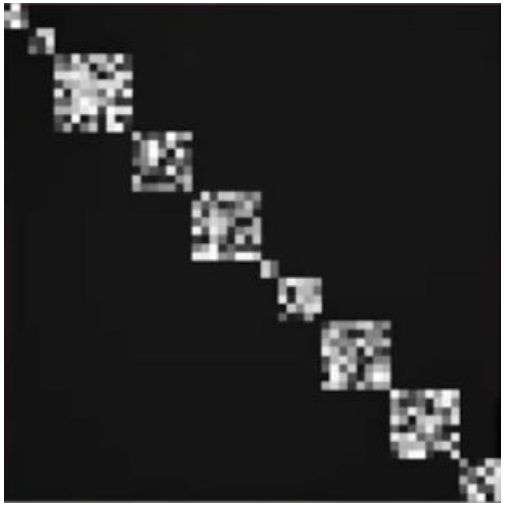
Basis decomposition regularization:

One or fewer weight matrix per layer, learn coefficient

$$W_r^l = \sum_{b=1}^{B} a_{rb}^l V_b^l$$

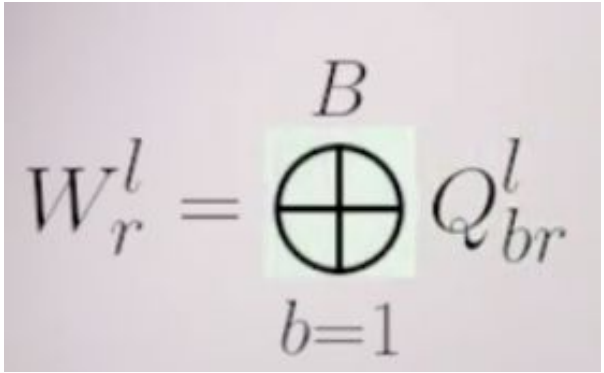
Block diagonal decomposition.

- Take smaller matrices and stack them diagonally in a larger matrix.
- Many of the parameters will be zero.
- Claim there are variables that are strongly interconnected within group, but don’t have much interaction outside of the group.



Block diagonal decomposition.

- Example: Person has physical characteristics and political affiliation.
- Way to reduce the parameters so they don’t overfit.
- GCNs throw out information about node type and relation type.
- RGCN’s allow you leverage that. Example, Twitter block and follow.
- Opens doors to modeling knowledge graphs, heterogeneous graphs.

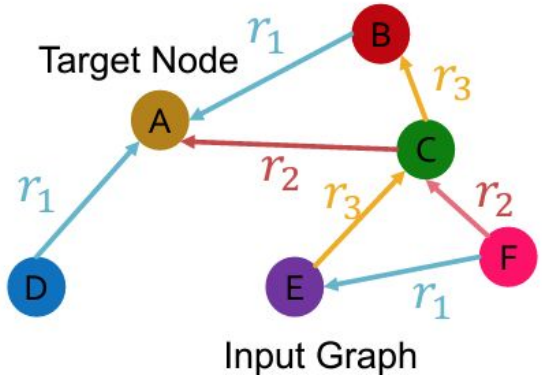


Example: Entity/Node Classification

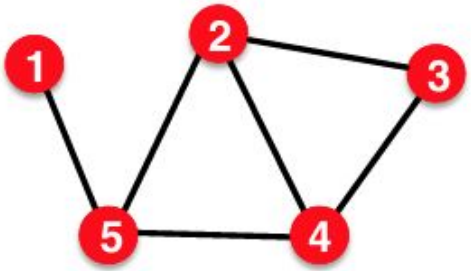
Goal: Predict the label of a given node

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): $\mathbf{h}() \in \mathbb{R}$, each item in $\mathbf{h}()$ represents the probability of that class.

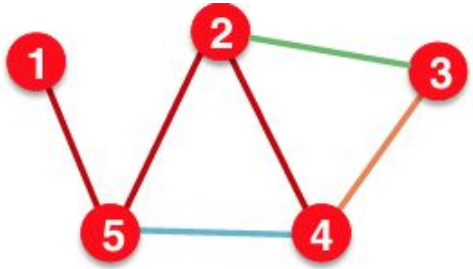


Link Prediction



The original graph

Split



Split Graph with 4 categories of edges

Summary RGCN

- Relational GCN, a graph neural network for heterogeneous graphs
- Can perform entity classification as well as link prediction tasks.
- Ideas can easily be extended into RGNN (RGraphSAGE, RGAT, etc.)