Graph Matching Networks for Learning the Similarity

Department of Electrical Engineering and Electronics

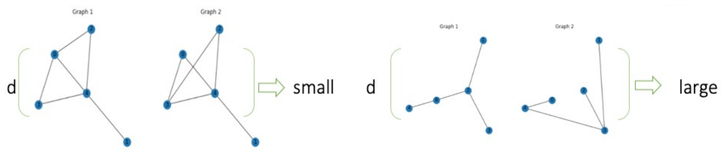
Xiaowei Shi (201522273) | Supervisor: Xinping Yi | Assessor: Valerio Selis

UNIVERSITY OF LIVERPOOL

INTRODUCTION

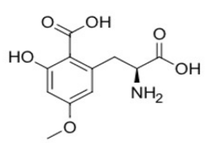
- Graph-structured data can be naturally used to represent data that have entities with relations.
- such as molecules, control flow graph, class diagram.

- Similarity Learning is to find similar graphs over a database under some notion of similarity.
- This notion varies from task to task.
- To develop a similarity function d that similar graphs output small distance, and dissimilar graphs output large distance.



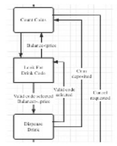
- Industrial relevance and applications

Drug discovery, code search, and software vulnerabilities.



Molecules ⇩ Drug Discovery

Program semantic relationship ⇩ Code Search



Control flow graph ⇩ Software Vulnerabilities

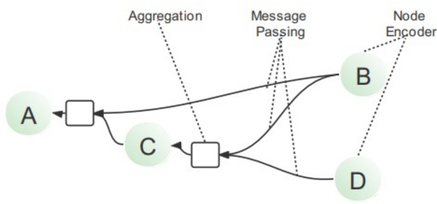
- Motivations
- Human-designed hash algorithmic functions that map each graph into a descriptor. (labor-intensive, time consuming)
- Widely used in security applications and is very good at find exact same graphs, not very good at find similar graphs.

INITIAL AND INTERMEDIATE DESIGNS

- Neural networks have worked well in many other domains.
- Traditional for sequences and grids structured data, for example, text, speech, images.
- Graphs different from other types of data (text and image)
- Up to four types of information: nodes, edges global-context and connectivity.

- Graphs to be compatible with neural networks?

- Graph Neural Networks are Neural networks that are much more broadly applicable.
- A GNN layer composes 3 separate MLP progressively transforming embeddings, without changing the connectivity.
- To be aware of graph connectivity, use message passing
- Which works in three steps: Node encoder, Message passing, Aggregation.



$$h_i^{(0)} = \mathrm{MLP}_{\mathrm{node}}(x_i)$$
$$e_{ij} = \mathrm{MLP}_{\mathrm{edge}}(x_{ij})$$
$$m_{i\to j} = f_{\mathrm{message}}(h_i^{(t)}, h_j^{(t)}, e_{ij})$$
$$h_i^{(t+1)} = f_{\mathrm{node}}(h_i^{(t)}, \textstyle\sum_{j:(j,i)\in E} m_{j\to i})$$
$$h_G = \mathrm{MLP}_{\mathrm{G}}\left(\sum_{i\in V} h_i^{(T)}\right).$$
$$h_G = \mathrm{MLP}_{\mathrm{G}}\left(\sum_{i\in V} \sigma(\mathrm{MLP}_{\mathrm{gate}}(h_i^{(T)})) \odot \mathrm{MLP}(h_i^{(T)})\right).$$

- Graph Matching Networks also get cross-graph messages
- The difference is at the message passing part, each node would not only get messages within the graph but also get cross-graph messages through an attention mechanism.

$$a_{i\to j} = \frac{\exp(s(h_i^{(t)}, h_j^{(t)}))}{\sum_j \exp(s(h_i^{(t)}, h_j^{(t)}))} \qquad \mu_i = \textstyle\sum_j a_{i\to j}(h_i^{(t)} - h_j^{(t)}) = h_i^{(t)} - \sum_j a_{i\to j} h_j^{(t)}$$
$$a_{j\to i} = \frac{\exp(s(h_i^{(t)}, h_j^{(t)}))}{\sum_i \exp(s(h_i^{(t)}, h_j^{(t)}))} \qquad \mu_j = \textstyle\sum_i a_{j\to i}(h_j^{(t)} - h_i^{(t)}) = h_j^{(t)} - \sum_i a_{j\to i} h_i^{(t)}$$

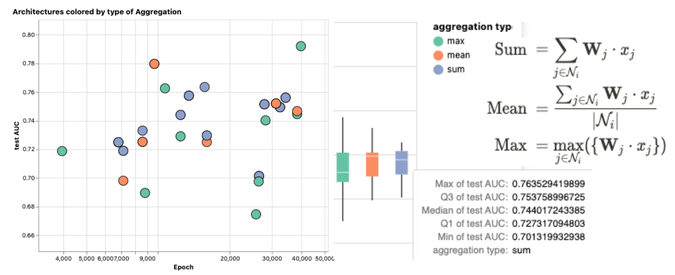
DESIGNS

- Goals achieved so far

- Implementation.
- Data loaders for "QM7b" dataset implemented with data pre-processed.
- Experiments conducted both on synthetic data and real-world dataset.
- Fine tuning with different aggregation type, and different sizes of node feature and edge feature dimension.
- alleviate the computational burden while preserving a good performance
- make it applicable to dataset containing a wider range of graph types, such as heterogeneous graphs.

PROJECT OUTPUT SO FAR

- Performance



- Experiment with aggregation type on QM7b dataset:
- Overall, it appears that sum has a very slight improvement on the AUC performance and max or mean can give equally good models.

- Discussion
- No specific aggregation type is a consistent the best option.
- Mean can be useful when a node has a highly variable number of neighbors
- Max operations are useful when you want to highlight a single
- Sum provides a balance between the above two.

CONCLUSIONS

- Summary and proposals for further work

The implementation that this project builds on top of existing code. Different data loaders of "QM7b" dataset were realized by data preprocessing, and different parameters were used and analyzed on synthetic data and real data in some experiments. In the future will focus on reduction on computation while preserving a good performance make it applicable to dataset containing heterogeneous graph types.

REFERENCE

No 3rd party images were used.

Code available:
https://github.com/ItsShi/GraphNeuralNetworkforSimilarityLearning