HetSAGE: Heterogenous Graph Neural Network for Relational Learning (Student Abstract)

Vince Jankovics, Michael Garcia Ortiz, Eduardo Alonso
Artificial Intelligence Research Centre (CitAI)
City, University of London
{vince.jankovics, michael.garcia-ortiz, e.alonso}@city.ac.uk

Abstract

This paper aims to bridge this gap between neuro-symbolic learning (NSL) and graph neural networks (GNN) approaches and provide a comparative study. We argue that the natural evolution of NSL leads to GNNs, while the logic programming foundations of NSL can bring powerful tools to improve the way information is represented and pre-processed for the GNN. In order to make this comparison, we propose HetSAGE, a GNN architecture that can efficiently deal with the resulting heterogeneous graphs that represent typical NSL learning problems. We show that our approach outperforms the state-of-the-art on 3 NSL tasks: CORA, MUTA188 and MovieLens.

Introduction

Neuro-symbolic learning (NSL) aims to combine the benefits of neural networks (NN) and logic based reasoning to efficiently identify patterns in relational data (França, Zaverucha, and d'Avila Garcez 2014). Graph neural networks (GNN) have been an increasingly popular field in the last decade, with models that allow to identify patterns from graph structured data (Hamilton, Ying, and Leskovec 2017). Many logic programs can be formulated as a knowledge graph or can be transformed into one without loss of information and node classification models can be utilised in a way that is directly comparable to NSL techniques. Therefore we argue that certain GNNs aim to solve the same problem as NSL systems, but more efficiently, as they fit the structure of the problem better, similarly to how convolutional NNs are more suitable for spatial information than fully-connected NNs.

In NSL, propositionalisation based techniques rely on the transformation of a complex relational database into an attribute-value form that enables the use of NN models, such as CILP++ (França, Zaverucha, and d'Avila Garcez 2014), considered state-of-the-art in NSL. In CILP++ the encoding of the data poses significant limitations in terms of flexibility, since each relation is explicitly included in the encoding, which is also problematic from a scalability perspective. Contrary to CILP++, GNNs preserve and exploit the structure of the data. Furthermore, GNNs can handle arbitrary number of connections between entities since they rely on learned functions that are applied across the nodes of the graph. Our proposed model, HetSAGE has similar components to previous GNN architectures, such as the sampling and aggregation from GraphSAGE (Hamilton, Ying, and Leskovec 2017), the edge feature handling from MPNN (Gilmer et al. 2017) and the heterogeneous embedding from HetGNN (Zhang et al. 2019).

Our contributions are as follows: (i) we provide a comparison between CILP++ and HetSAGE to solve relational learning tasks, (ii) we propose a method to transform a logic program to a graph, and (iii) we study the inclusion of the target labels in the neighbouring nodes, which is similar to the approach described in (Zhou et al. 2004). Our study also shows that the natural evolution of NSL leads to GNNs, providing a significant boost to the performance of state-of-the-art in NSL. Additionally, we show how logic programming can be used alongside GNNs to formulate and enrich the graph database for the GNN.

Methods

Knowledge representation Logic programming formulates the problem as a set of clauses in the form of `head:-body` which represents $body \implies head$. Rules with variables provide a way to express general relationships between entities, e.g. the rule `p2(X,Y):-p1(X,Z),p1(Z,Y)` translates to the first order logic formula: $\forall\, X, Y\ \exists\, Z,\ p1(X,Z) \wedge p1(Z,Y) \implies p2(X,Y)$. We utilise this equivalence to transform the logic program D to its grounded form D_g that only contains variable free terms but still entails the same consequences S, i.e. $D \models S \iff D_g \models S$. A typical translation from logic programs to graphs would map arity 0 terms to nodes, arity 1 to node attributes, arity 2 to labelled edges and higher arity terms can be transformed using reification (Grewe 2010).

HetSAGE Our method uses a uniform sampling technique similar to GraphSAGE (Hamilton, Ying, and Leskovec 2017), but we create non-overlapping subgraphs for each target node (see Fig. 1). The first step in HetSAGE is the heterogeneous embedding that maps each node's feature vector in a shared embedding space. The embedding $\mathbf{h}$ for node v, type t, is calculated according to $\mathbf{h}^v = f_t^h(\mathbf{x}^v)$ where f_t^h is a

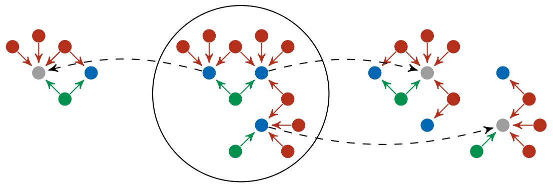

Figure 1: Our neighbourhood sampling: the initial graph in the circle with 3 node types (e.g. movie, actor, director) and blue nodes as targets. The 3 sampled neighbourhoods show the 2-step walk sampled around each target node. Target nodes in the sampled neighbourhood (gray) do not have the target label included (e.g. genre), while the surrounding movie nodes have them.

NN, and $\mathbf{x}^v$ is the node feature vector. The next step is propagating the information between the nodes along the edges by the message passing module and a readout layer outputs the predictions, according to:

$$\mathbf{m}_{i+1}^v = \sum_{u \in \mathcal{N}_s(v)} f^m(\mathbf{e}^{vu})\mathbf{h}_i^u \quad (1)$$

$$\mathbf{h}_{i+1}^v = f_i^a(\mathbf{h}_i^v, \mathbf{m}_{i+1}^v) \quad (2)$$

$$y^v = f^r(\mathbf{h}_N^v) \quad (3)$$

where $\mathbf{e}^{vu}$ is an edge feature, $\mathbf{h}_{i+1}^v$ is the hidden state of node v in layer i, and f^m, f_i^a, f^r are NNs. The final node labels y^v are only calculated for target nodes. We included batch normalisation applied to $\mathbf{h}_{i+1}^v$ after each message passing unit and layer normalisation for $\mathbf{h}_i^v$ and $\mathbf{m}_{i+1}^v$ individually before they are passed to f_i^a. This helps to balance the information carried forward from the node and the information from its neighbourhood. We used cross-entropy as loss.

Experiments

We conducted experiments on MovieLens, CORA and MUTA188 from the Relational Dataset Repository[1]. The results are shown in Table 1. MUTA188 consists of 188 molecules, but we represent the data as a single graph with `atoms` as nodes and `bonds` between them and an additional node type `drug` that has the target labels assigned. This representation matches how the other two problems are defined, providing a more fair comparison. For CORA, our model is capable of dealing with the original problem setting with 7 classes, but CILP++ can only act as binary classifier (hence the N/A in the table). For comparison purposes we run the experiments with binary labels as well, assigning 1 to every paper that has the label *neural networks* and 0 otherwise (shown as CORA-binary). For the CILP++ experiments we used the architecture and hyperparameters from (França, Zaverucha, and d'Avila Garcez 2014). We present a comparison between three different versions of CORA, one with the bag-of-words encoding of the content of each paper and the paper labels, one without the content (*no words*) and one without the labels (*no labels*) to evaluate the benefit of label propagation.

[1]https://relational.fit.cvut.cz/

Dataset	CILP++ (%)	GNN (%)
MUTA188	89.74 (±5.32)	92.11 (±4.40)
CORA-binary no words	70.34 (±0.00)	93.68 (±2.15)
CORA-binary no labels	70.34 (±0.00)	92.92 (±0.98)
CORA-binary	69.70 (±0.07)	91.98 (±2.91)
CORA-multi no words	N/A	88.17 (±0.86)
CORA-multi no labels	N/A	81.85 (±0.58)
CORA-multi	N/A	83.38 (±1.44)
MovieLens	80.12 (±0.76)	80.22 (±0.80)

Table 1: Accuracies averaged over 10 random train/test split. HetSAGE outperforms CILP++ on each benchmark. CORA without the contents of the papers outperforms the experiments that include them, which could be because the propagated labels carry more information than the contents, and the model does not need to distinguish between the noise and the more relevant information. Including the labels seem to improve the performance of the model.

Conclusion

We demonstrated that HetSAGE is capable of handling problems defined as a logic program and that our model outperforms CILP++. We argue that GNNs in general are more suitable for addressing structured data than NSL. We additionally implemented a transformation that takes a logic program and generates a graph from it. This can also provide a powerful tool to include additional information (or common sense) in the data. Future work comprises conducting a larger scale comparison on what additional information helps the model and a more thorough hyperparameter and architecture search on HetSAGE.

References

França, M. V. M.; Zaverucha, G.; and d'Avila Garcez, A. S. 2014. Fast Relational Learning Using Bottom Clause Propositionalization with Artificial Neural Networks. *Machine Learning* 94(1): 81–104. ISSN 1573-0565.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. *arXiv:1704.01212 [cs]* .

Grewe, N. 2010. A Generic Reification Strategy for N-Ary Relations in DL. In *OBML 2010 Workshop Proceedings*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*, 1024–1034.

Zhang, C.; Song, D.; Huang, C.; Swami, A.; and Chawla, N. V. 2019. Heterogeneous Graph Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, 793–803. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-6201-6.

Zhou, D.; Bousquet, O.; Lal, T. N.; Weston, J.; and Schölkopf, B. 2004. Learning with Local and Global Consistency. In *Advances in Neural Information Processing Systems*, 321–328.