

Neural Symbolic Systems on Graphs

Pedro H.C. Avelar Luis C. Lamb

Universidade Federal do Rio Grande do Sul

September 2019

Deep Learning on Graphs

Selsam et al's NeuroSAT

Inductive Bias

Neural Networks have been worked upon to provide end-to-end differentiable means to work with several data structures.

Inductive Bias

Neural Networks have been worked upon to provide end-to-end differentiable means to work with several data structures.

- ▶ Fully Connected NNs (MLPs) doesn't force a structure on the input and thus must learn both the transformation as well as the structure of the problem

Inductive Bias

Neural Networks have been worked upon to provide end-to-end differentiable means to work with several data structures.

- ▶ Fully Connected NNs (MLPs) doesn't force a structure on the input and thus must learn both the transformation as well as the structure of the problem
- ▶ Recursive NNs work by iterating over discrete sequences of items

Inductive Bias

Neural Networks have been worked upon to provide end-to-end differentiable means to work with several data structures.

- ▶ Fully Connected NNs (MLPs) doesn't force a structure on the input and thus must learn both the transformation as well as the structure of the problem
- ▶ Recursive NNs work by iterating over discrete sequences of items
- ▶ Relational Networks [Santoro et al., 2017] and set2set [Vinyals et al., 2016] work with sets

Inductive Bias

Neural Networks have been worked upon to provide end-to-end differentiable means to work with several data structures.

- ▶ Fully Connected NNs (MLPs) doesn't force a structure on the input and thus must learn both the transformation as well as the structure of the problem
- ▶ Recursive NNs work by iterating over discrete sequences of items
- ▶ Relational Networks [Santoro et al., 2017] and set2set [Vinyals et al., 2016] work with sets
- ▶ [Grefenstette et al., 2015] provided neural computers with stack and queue inspired architectures.

Inductive Bias

Neural Networks have been worked upon to provide end-to-end differentiable means to work with several data structures.

- ▶ Fully Connected NNs (MLPs) doesn't force a structure on the input and thus must learn both the transformation as well as the structure of the problem
- ▶ Recursive NNs work by iterating over discrete sequences of items
- ▶ Relational Networks [Santoro et al., 2017] and set2set [Vinyals et al., 2016] work with sets
- ▶ [Grefenstette et al., 2015] provided neural computers with stack and queue inspired architectures.
- ▶ What about Graphs?

Graph Neural Networks

[Sperduti and Starita, 1997] gave one of the first suggestions on using neural networks with graph-structure data

Graph Neural Networks

[Sperduti and Starita, 1997] gave one of the first suggestions on using neural networks with graph-structure data, but [Gori et al., 2005] constructed a model that is much akin to today's architectures

Graph Neural Networks

[Sperduti and Starita, 1997] gave one of the first suggestions on using neural networks with graph-structure data, but [Gori et al., 2005] constructed a model that is much akin to today's architectures and in [Scarselli et al., 2009] they reaffirmed the model in its full potential.

The Graph Neural Network Model

Main Ideas:

The Graph Neural Network Model

Main Ideas:

1. Build the network of neural modules according to the input graph structure

The Graph Neural Network Model

Main Ideas:

1. Build the network of neural modules according to the input graph structure
2. Propagate information until it converges

The Graph Neural Network Model

Main Ideas:

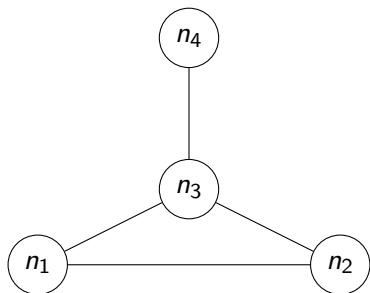
1. Build the network of neural modules according to the input graph structure
2. Propagate information until it converges
3. Readout desired information through another neural module

The Graph Neural Network Model

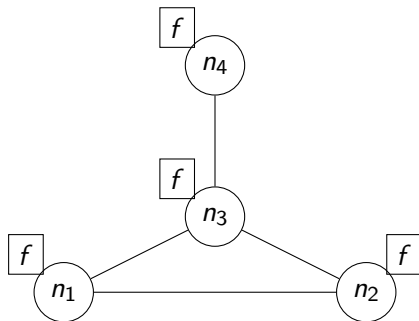
Main Ideas:

1. Build the network of neural modules according to the input graph structure
2. Propagate information until it converges
3. Readout desired information through another neural module
4. [Scarselli et al., 2009] One could adapt this to work with different “kinds” of nodes (i.e. the edge and graph nodes in [Battaglia et al., 2018])

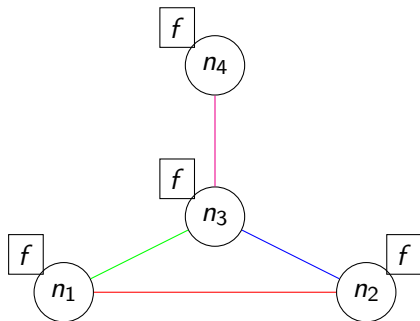
The Graph Neural Network Model: A Graph



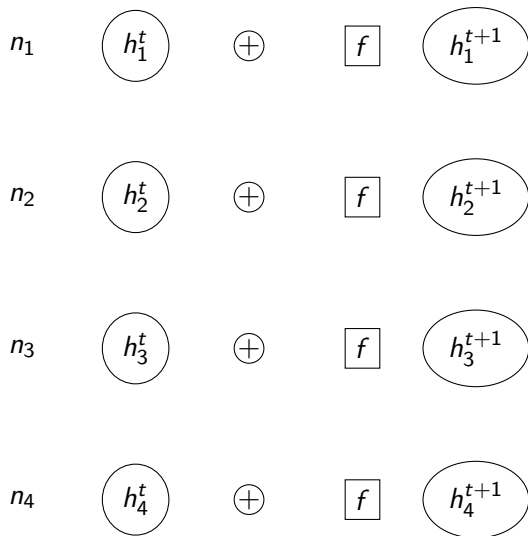
The Graph Neural Network Model: Neural Modules corresponding to each node



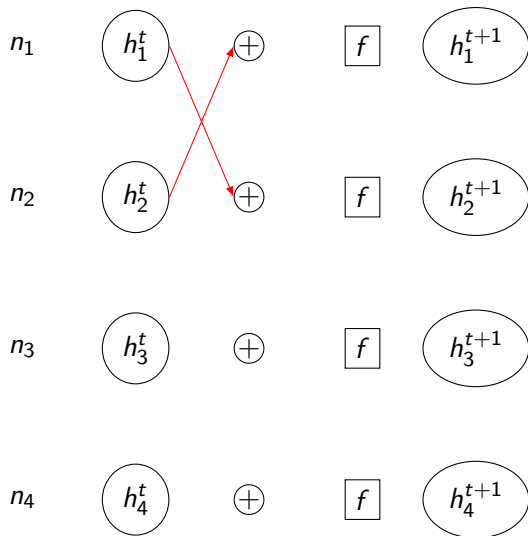
The Graph Neural Network Model: Neural Modules corresponding to each node



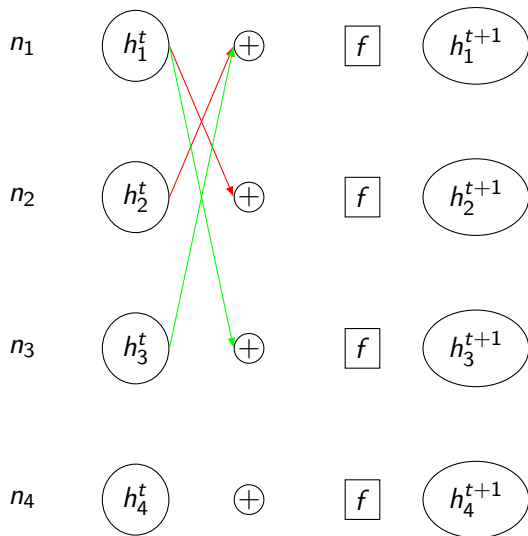
The Graph Neural Network Model: One propagation layer



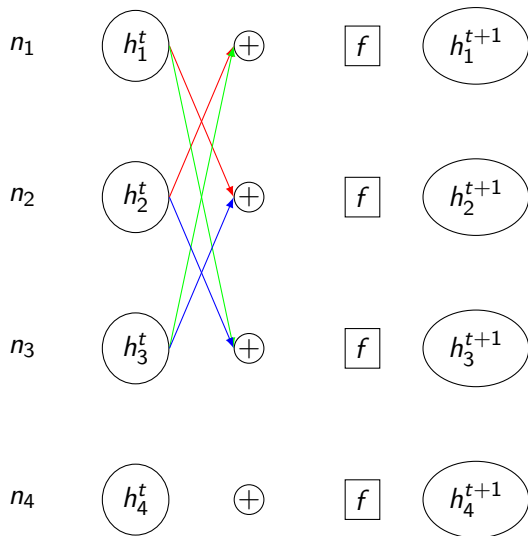
The Graph Neural Network Model: One propagation layer



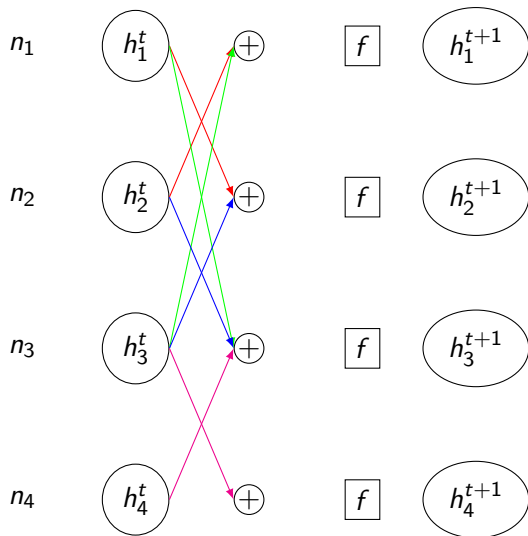
The Graph Neural Network Model: One propagation layer



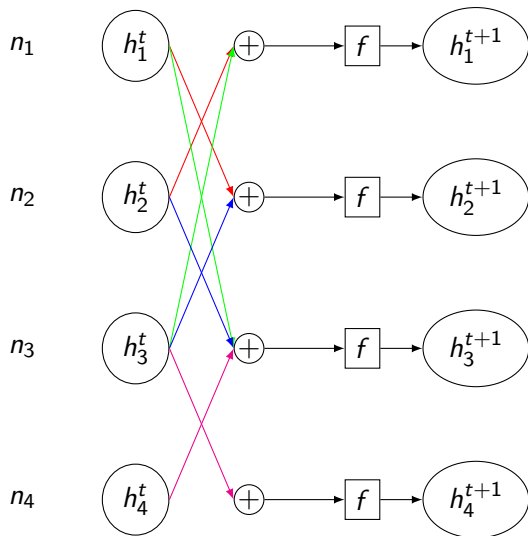
The Graph Neural Network Model: One propagation layer



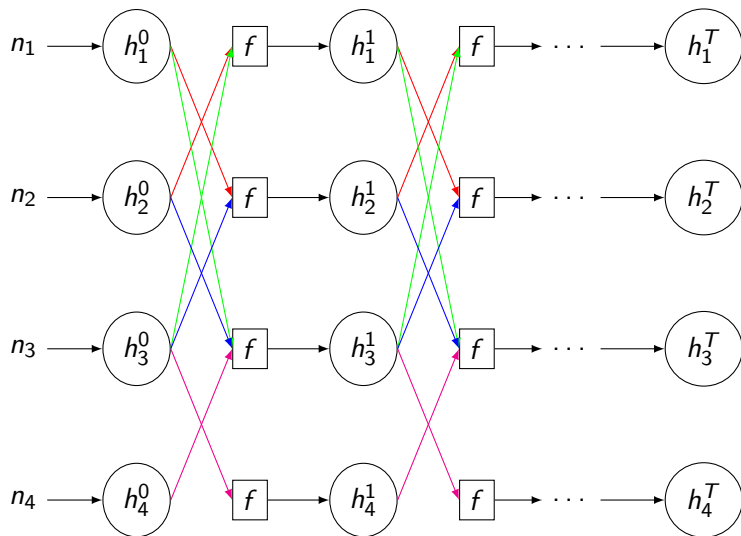
The Graph Neural Network Model: One propagation layer



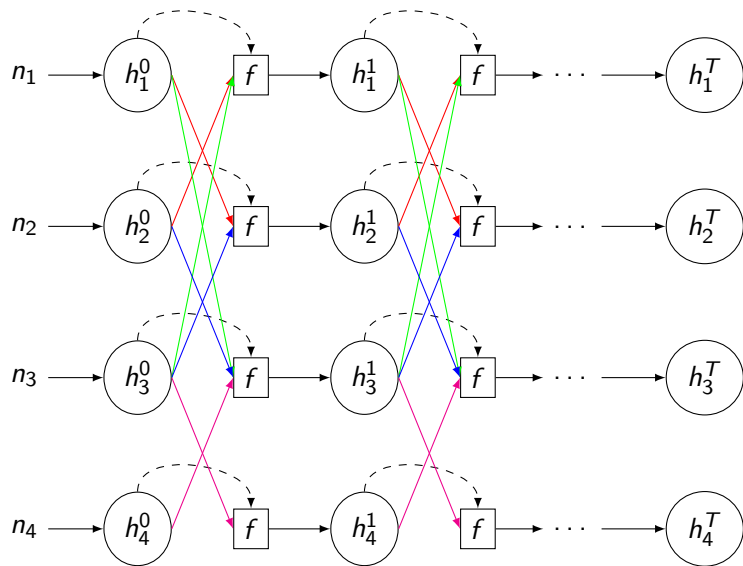
The Graph Neural Network Model: One propagation layer



The Graph Neural Network Model: Unrolling in time



The Graph Neural Network Model: Recurrent connections



Mathematical formalisation

With sum-aggregation, it is easy to organise the propagation as matrix multiplication. Given an adjacency matrix \mathbf{M} with shape (n, n) , the learned function f , and the current node “state” \mathbf{h}^t with shape (n, d) , we have:

$$\mathbf{h}^{t+1} = f(\mathbf{M} \times \mathbf{h}^t)$$

Which translates, on a node-to-node basis as:

$$\mathbf{h}_n^{t+1} = f\left(\sum_{j \in \mathcal{N}(n)} \mathbf{h}_j^t\right)$$

With $\mathcal{N}(n)$ being the neighbourhood of node n .

Message-Passing Neural Networks

Same principles as [Gori et al., 2005, Scarselli et al., 2009]:

Message-Passing Neural Networks

Same principles as [Gori et al., 2005, Scarselli et al., 2009]:

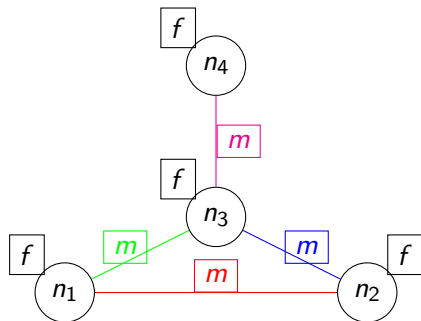
- ▶ A separate “messaging” function added between communicating nodes.

Message-Passing Neural Networks

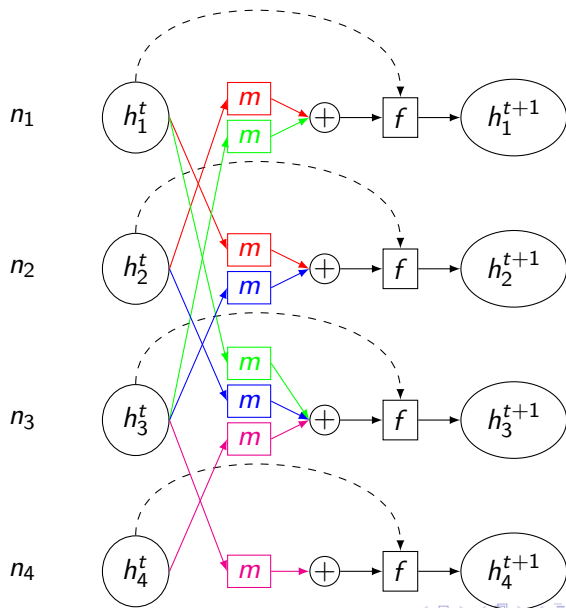
Same principles as [Gori et al., 2005, Scarselli et al., 2009]:

- ▶ A separate “messaging” function added between communicating nodes.
- ▶ “Messages” propagated for a given number of iterations instead

Message-Passing Neural Networks

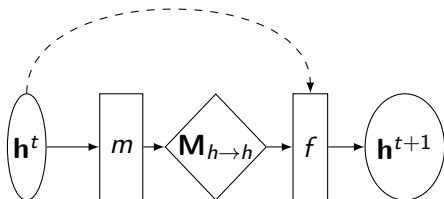


Message-Passing Neural Networks



MPNN “blocks”

We'll use the following graphical representation for simplifying the MPNN blocks, ellipses are state tensors, rectangles are learned functions, diamonds represent matrix multiplication (or any other type of neighbourhood aggregation), circles (not present) are pointwise operations, and if two arrows join at a point their tensors are most likely concatenated for every tuple of node:



Details Left Out

- ▶ Messaging functions often take more than just the source's embedding.
- ▶ Neighbourhood aggregation can be made in many ways:
 - ▶ Sum-aggregation [Gori et al., 2005, Scarselli et al., 2009, Gilmer et al., 2017, Selsam et al., 2018, Xu et al., 2019]
 - ▶ Average-aggregation [Kipf and Welling, 2017]
 - ▶ Attentional sum-aggregation [Velickovic et al., 2018]
 - ▶ Product-aggregation
 - ▶ Or any other order-invariant function
- ▶ Readout functions can be really diverse, using only specific nodes, all nodes of a kind/type, pairs of nodes, etc...
- ▶ Batching of different input graphs (done as a block-diagonal matrix)

Graph Convolutions

Independently, generalisations of convolutions to graph-domains, such as [Kipf and Welling, 2017], led to an architecture similar to [Scarselli et al., 2009], only with no recurrent connection, untied weights, and average-aggregation instead of sum-aggregation.

Graph Networks

[Battaglia et al., 2018] suggests the generalisation of the concept of graph neural networks to models other than neural networks, builds upon this idea of inductive bias and argues for three specific kinds/types of nodes: Vertex, Edge and Graph-level nodes.

Recap

We can:

- ▶ Assemble neural modules in the configuration of the input graphs and propagate information through neighbours.
- ▶ Work with different “kinds” /types of nodes by training different “update” and “message” functions, working on different sets of embeddings, and different adjacency matrices (we’ll see examples soon).
- ▶ Read out the embeddings at the last layer and use them to provide an answer to the problem at hand.

NeuroSAT: CNF formulas as Hypergraphs

A SAT formula can be seen as a hypergraph/bipartite graph between literals and clauses [Selsam et al., 2018]

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2)$$

NeuroSAT: CNF formulas as Hypergraphs

A SAT formula can be seen as a hypergraph/bipartite graph between literals and clauses [Selsam et al., 2018]

$$x_1 \quad \neg x_1 \quad x_2 \quad \neg x_2 \quad x_3 \quad \neg x_3$$

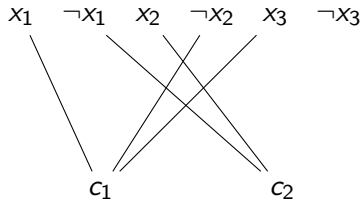
c_1

c_2

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2)$$

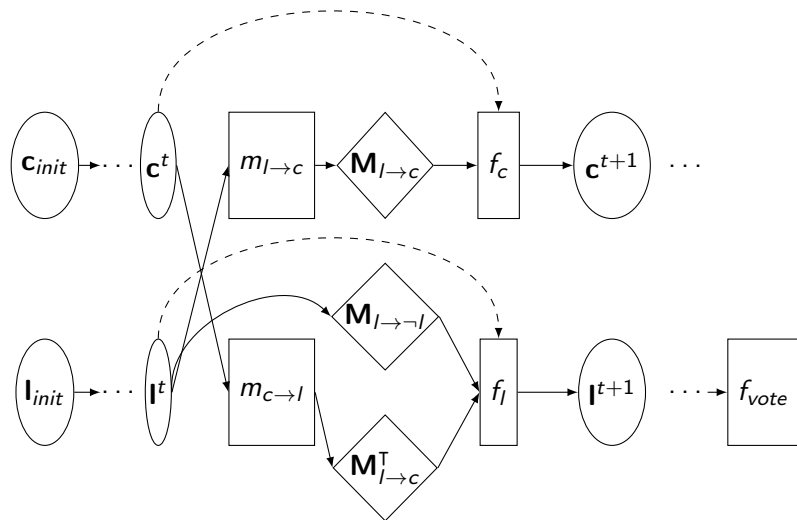
NeuroSAT: CNF formulas as Hypergraphs

A SAT formula can be seen as a hypergraph/bipartite graph between literals and clauses [Selsam et al., 2018]



$$(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2)$$

The NeuroSAT Model



Summary

- ▶ Two types of nodes: literals and clauses.
- ▶ Literals communicate with clauses they are in and with their negated counterparts
- ▶ Clauses communicate with literals they contain
- ▶ Same (learned) initial embedding for all literals and another for all clauses.
- ▶ Readout function as each literal voting on satisfiability, final answer by averaging

Major Outcomes

The 3 most important things in [Selsam et al., 2018]'s experiments were, in our opinion:

- ▶ Provided a way to build pairs of almost equal problems with radically different answers
- ▶ An algorithm-like neural message-passing procedure that has monotonically non-decreasing performance over time.
- ▶ A way to extract meaningful information to the point of reconstructing solutions to the problems

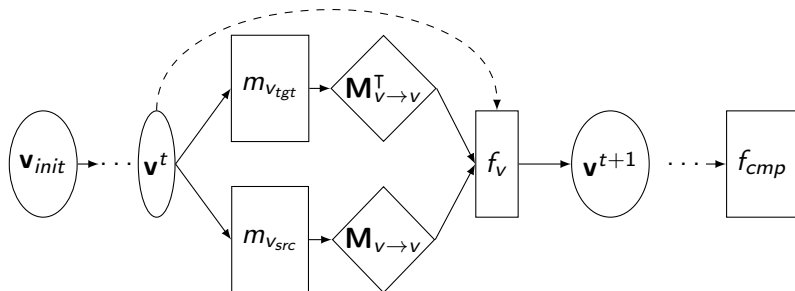
Where We Came From

The three achievements cited above came as a guidance for our experiments. All our models are direct applications of the MPNN in Selsam et al's NeuroSAT architecture, with changes made mostly at the beginning and end of the pipeline and thus we'll leave the details out and give only the overview of these models.

Centrality

Simple application of a GNN/MPNN based on previous work at our lab. For the readout we arrange every possible pair of node embeddings and produce a ranking by training a MLP module serve as a “greater than” comparison. That is: $r_c(\mathbf{h}_a, \mathbf{h}_b) = 1$ if $c(n_a) > c(n_b)$ else 0, with r_c being the learned comparison module for centrality c , \mathbf{h}_i being the embedding for node i and $c(n_i)$ the centrality value of that node under centrality measure c .

Centrality – The Model



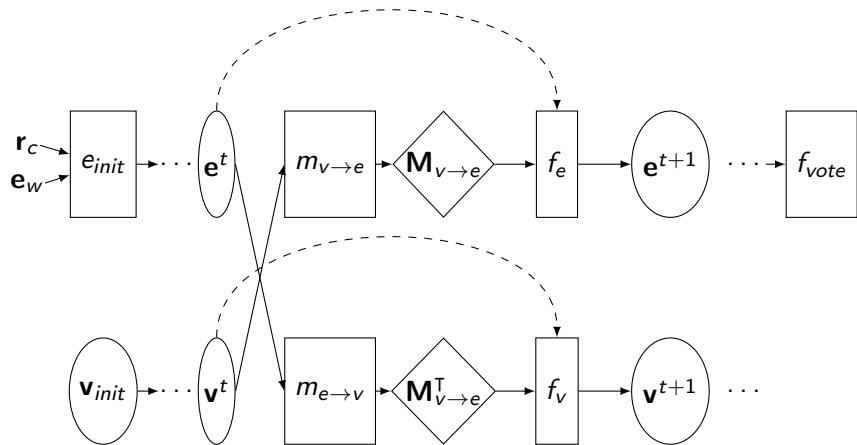
Centrality – Summary

- ▶ Only nodes.
- ▶ Nodes communicate with neighbours
- ▶ Same (learned) initial embedding for all nodes.
- ▶ Readout function as each pair of nodes being compared based on their centrality values

Travelling Salesman Problem

Exactly the same idea as NeuroSAT, only the voting occurs in the edges and the initial embedding is initialised through a learned function that takes as inputs an edge's cost and the maximum route cost. Couldn't find a way to interpret the embeddings as in NeuroSAT.

TSP – The Model



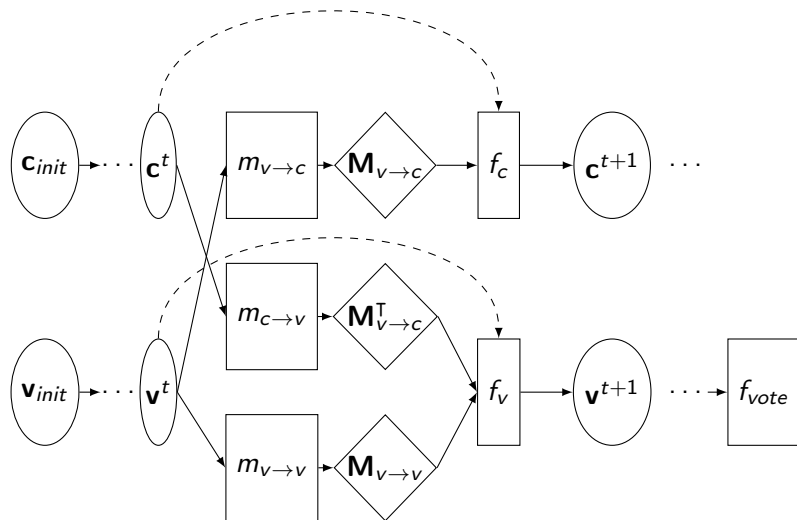
TSP – Summary

- ▶ Nodes and Edges.
- ▶ Nodes communicate with edges and edges with nodes
- ▶ Same (learned) initial embedding for all nodes.
- ▶ Edges initialised according to a (learned) function on the edge's cost and the maximum path length.
- ▶ Readout function as each edge voting on satisfiability, final answer by averaging values
- ▶ Trained on solving the same problem with a small deviation above/below the optimum.

Graph Colouring

Exactly the same idea as NeuroSAT, only that the initial embeddings for the colours are random. Results interpretable in the sense of conflicts in the clusterings.

GCP – The Model



GCP – Summary

- ▶ Nodes and Colours.
- ▶ Nodes communicate between themselves and with colours, colours communicate with nodes
- ▶ Same (learned) initial embedding for all nodes.
- ▶ Randomly initialised colour embeddings.
- ▶ Readout function as each vertex (node) voting on satisfiability, final answer by averaging values
- ▶ Trained on solving the same problem with the chromatic number and one below the chromatic number.


Link Prediction


Currently working on a model for Link Prediction, extending the work of [McCallum et al., 2017] with MPNNs, achieving good results so far.

Key Ideas from Experience

- ▶ Work in the domain of the problem: Node-centralities should focus on node embeddings, edge-centric problems such as TSP/Link prediction should focus on edge embeddings, if you need k different “graph-level” embeddings for k coloring then have them.
- ▶ Too many propagation steps can make it harder to learn.
- ▶ Training the loss on each message-passing iteration helps solutions with monotonically increasing performance over more iterations (as in [Palm et al., 2018]).
- ▶ If working on a symbolic problem, using instances that are very similar in structure but have opposite answers also help training.
- ▶ The fun part is less about the GNN/MPNN model itself and more about how you engineer the pipeline before and after the model.

 Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V. F., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gülçehre, Ç., Song, F., Ballard, A. J., Gilmer, J., Dahl, G. E., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R. (2018). Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261.

 Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR.

 Gori, M., Monfardini, G., and Scarselli, F. (2005). A new model for learning in graph domains.

In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE.



Grefenstette, E., Hermann, K. M., Suleyman, M., and Blunsom, P. (2015).

Learning to transduce with unbounded memory.

In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1828–1836.



Kipf, T. N. and Welling, M. (2017).

Semi-supervised classification with graph convolutional networks.

In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.



McCallum, A., Neelakantan, A., Das, R., and Belanger, D. (2017).

Chains of reasoning over entities, relations, and text using recurrent neural networks.

In Lapata, M., Blunsom, P., and Koller, A., editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 132–141. Association for Computational Linguistics.



Palm, R. B., Paquet, U., and Winther, O. (2018).

Recurrent relational networks.

In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 3372–3382.



Santoro, A., Raposo, D., Barrett, D. G. T., Malinowski, M., Pascanu, R., Battaglia, P. W., and Lillicrap, T. (2017).

A simple neural network module for relational reasoning.

In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors,

Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA, pages 4967–4976.



Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009).

The graph neural network model.

IEEE Transactions on Neural Networks, 20(1):61–80.



Selsam, D., Lamm, M., Bünz, B., Liang, P., de Moura, L., and Dill, D. L. (2018).

Learning a SAT solver from single-bit supervision.

CoRR, abs/1802.03685.



Sperduti, A. and Starita, A. (1997).

Supervised neural networks for the classification of structures.

IEEE Trans. Neural Networks, 8(3):714–735.



Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018).

Graph attention networks.

In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.



Vinyals, O., Bengio, S., and Kudlur, M. (2016).

Order matters: Sequence to sequence for sets.

In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.



Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019).

How powerful are graph neural networks?

In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, United States, May 6 - May 9, 2019, Conference Track Proceedings*. OpenReview.net.