Learning Traveling Salesperson Routes with Graph Neural Networks

Marcelo Prates[*]
Institute of Informatics
Federal University of Rio Grande do Sul
Porto Alegre, Brazil
morprates@inf.ufrgs.br

Pedro Avelar[*]
Institute of Informatics
Federal University of Rio Grande do Sul
Porto Alegre, Brazil
pedro.avelar@inf.ufrgs.br

Luis Lamb
Institute of Informatics
Federal University of Rio Grande do Sul
Porto Alegre, Brazil
lamb@inf.ufrgs.br

Abstract

1 Introduction

2 Motivation

Graph Neural Networks (GNNs) have been applied to a wide range of relevant problems [citations needed]. Very interesting recent work has shown how GNNs can be employed to bridge the divide between the neural and symbolic schools of artificial intelligence. In particular, Selsam *et al.* [2018] has succeeded in training a GNN as a boolean satisfiability (SAT) predictor by feeding it CNF formulas as features and satisfiability bits (one per formula) as labels, thus learning a rudimentary SAT solver from single-bit supervision. The "neurosat" architecture, as the authors call it, is rather simple: it consists of assigning a multidimensional embedding $\in \mathbb{R}^d$ for each literal (i.e. x_3, $\neg x_1$, x_{10}, etc.) and each clause in a CNF formula and then connecting these embeddings accordingly so that they can send messages to one another – literals are connected with the clauses in which they appear and vice-versa, and each literal is also connected to its negated variant (i.e. x_1 and $\neg x_1$). The model is run for a given number of message passing iterations in which each node in the GNN (i.e. a literal or a clause) adds up element-wise all of the embeddings of its incoming messages, received along its adjacencies, to obtain an accumulated message. The main trainable component of this architecture is a recurrent unit assigned with updating the embedding of a node with this accumulated message. In a nutshell, neurosat iterates a process of "embedding refinement" in which literals and clauses become enriched with information about their symbolic neighborhood. After that, each refined literal embedding is translated (by the means of a multilayer perceptron) into a logit probability, all of which are averaged to produce a final satisfiability prediction.

The success story of neurosat is an invitation to assess whether similar performance can be obtained for other $\mathcal{NP}$-Hard problems. Graph problems are suitable candidates, as they fit nicely into the relational structure of GNNs. The Traveling Salesperson Problem (TSP), assigned with finding the shortest length Hamiltonian path in a graph (that which visits each vertex exactly once), is a natural choice given its last longing relevance to computer science. Studying GNN models for TSP has

[*]Equal conttribution

another advantage: it allows us to assess whether GNNs can tackle problems where the connections between elements are labeled with numerical information – in our case, edges carrying differing weights.

3 Our Model

Typical GNN architectures link adjacent vertices in the graph which represents the problem instance by opening a direct communication channel between their corresponding embeddings, in such a way that one can send messages to the other. Concretely, if we picture the collection of all n vertex embeddings at time t as a matrix $\mathbf{V}^t \in \mathbb{R}^{n \times d}$, the process of filtering the incoming messages for each node corresponds to performing a matrix multiplication with the graph's adjacency matrix, $\mathbf{M}$. Then the process of refining all embeddings at once can be expressed by Equation 1, where $\mathbf{V}_u$ is a recurrent unit and $\mathbf{V}_h^t \in \mathbb{R}^{n \times d}$ is a matrix in which each line $\mathbf{V}_h^t[i]$ contains the hidden state, at time t, corresponding to the i-th vertex.

$$\mathbf{V}_h^{t+1}, \mathbf{V}^{t+1} \leftarrow \mathbf{V}_u(\mathbf{V}_h^t, \mathbf{M} \times \mathbf{V}^t) \tag{1}$$

However one can see that this approach fails when connections are augmented with some numerical information such as edge weights or capacities. To feed this information into the network, we propose a new architecture which elevates edges to a "node" status in the context of the GNN. Concretely, we will assign embeddings to both vertices and edges, and link the embedding of each edge $e = (v_1, v_2)$ with the embeddings of the two vertices it connects (v_1 and v_2). The importance of having edge embeddings is that now we can feed them their "labels" (i.e. edge weights, edge capacities, etc.) at each message-passing timestep.

Algorithm 1 Graph Neural Network TSP Solver

```
 1: procedure GNN-TSP($\mathcal{G} = (\mathcal{V}, \mathcal{E})$)
 2:     Enumerate the edges of the graph, assigning an unique index $\mathcal{E}_{idx}[(v_1, v_2)]$ for each edge $(v_1, v_2) \in \mathcal{E}$
 3:
 4:     // Compute adj. matrix from edges to source vertices
 5:     $\mathbf{M}_{\mathbf{EV_{src}}}[e, v] \leftarrow 1$ if $\exists v'(\mathcal{E}_{idx}[(v, v')] = e)$
 6:     // Compute adj. matrix from edges to target vertices
 7:     $\mathbf{M}_{\mathbf{EV_{tgt}}}[e, v] \leftarrow 1$ if $\exists v'(\mathcal{E}_{idx}[(v', v)] = e)$
 8:
 9:     // Run $t_{max}$ message-passing iterations
10:     for $t = 1 \ldots t_{max}$ do
11:         // Refine each vertex embedding with messages received from all the edges in which it appears
12:         // either as a source or a target vertex
13:         $\mathbf{V}_h^{t+1}, \mathbf{V}^{t+1} \leftarrow V_u(\mathbf{V}_h^t, \mathbf{M}_{\mathbf{EV_{src}}} \times E_{msg}(\mathbf{E}^t), \mathbf{M}_{\mathbf{EV_{tgt}}} \times E_{msg}(\mathbf{E}^t))$
14:         // Refine each edge embedding with (two) messages received from its source and its target vertex
15:         $\mathbf{E}_h^{t+1}, \mathbf{E}^{t+1} \leftarrow E_u(\mathbf{E}_h^t, \mathbf{M}_{\mathbf{EV_{src}}}{}^T \times V_{msg}(\mathbf{V}^t), \mathbf{M}_{\mathbf{EV_{tgt}}}{}^T \times V_{msg}(\mathbf{V}^t), \mathbf{W})$
16:
17:     // Translate edge embeddings into logit probabilities
18:     $E_{logits} \leftarrow E_{vote}(E^{t_{max}})$
19:     // Convert logit probabilities into probabilities
20:     $E_{prob} \leftarrow sigmoid(E_{logits})$
21:     // Extract route
22:     $Route \leftarrow \{e \mid e \in \{1, \ldots, |\mathcal{E}|\}, E_{prob}[e] > 0.5\}$
```

4 Experimental Setup

5 Results and Analysis

6 Discussion

References

Daniel Selsam, Matthew Lamm, Benedikt Bunz, Percy Liang, Leonardo de Moura, and David L Dill. Learning a sat solver from single-bit supervision. *arXiv preprint arXiv:1802.03685*, 2018.