Learning Domain-Specific Heuristics with Graph Convolutional Networks

Matheus Z. Marcon
Graduate Program in Computer Science - School of Technology
Pontifical Catholic University of Rio Grande do Sul - PUCRS
Porto Alegre, Brazil
matheus.marcon@edu.pucrs.br

Abstract

Heuristic functions are essential tools for improving the performance of search algorithms. Off-the-shelf functions work well on a variety of domains, but may however behave poorly in more complex scenarios. In this work, we propose the usage of Graph Convolutional Networks (GCNs) for exploiting the inherent graph-like structure of planning tasks and learning domain-specific heuristic values for non-consuming problems, which could later be generalized for harder instances.

Introduction

Automated Planning (AP) is a branch of Artificial Intelligence (AI) which seeks to compute sets of actions, or plans, that fulfill a given task. Planning can, however, become expensive in certain situations, making the selection of optimal heuristic functions of great importance.

An heuristic $h(S)$ informs the planner about cost estimates from a given state S to the goal S^G, guiding the search process towards more efficient paths. In order to function properly, heuristics need to be informative, returning cost values as close as possible to the true remaining cost. When tackling complex scenarios, off-the-shelf heuristics can nonetheless present scaling problems and lead to poor planning and performance results when compared to toy problems.

As such, domain-specific designs might be required. Designing heuristics in such a manner relies on accurate domain and search control knowledge by an expert, which is unfeasible for many real-world problems. Machine Learning (ML) techniques have been widely used as an alternative to human knowledge to circumvent these issues, as thoroughly reviewed in (Jiménez et al. 2012).

Deep Learning (DL) is a sub field of ML which has been popularized for its capabilities of working on unstructured data. In the field of AP, it has been previously used to find optimal search heuristics for given tasks (Sigurdson and Bulitko 2017; Loreggia et al. 2016).

Introduced in 2012 (Krizhevsky, Sutskever, and Hinton 2012), Convolutional Neural Networks (CNNs), Deep Learning's most used network architecture, consist of a data-driven approach for feature selection, achieving state-of-the-art performance in image and text data analysis.

Planning tasks can be naturally represented in graphs, where a graph G is composed of nodes N which represent states S, and edges ε representing the actions connecting each state. Graph Convolutional Networks (GCNs), graph analogs to CNNs introduced in (Defferrard, Bresson, and Vandergheynst 2016), are networks which take graph data as input. These networks compute topological or "neighbourhood" information about each data object, departing from the purely local bias of CNNs.

In this work, we propose the usage of GCNs for learning domain-specific heuristics for non-complex scenarios, with the aim of later generalizing obtained results for more demanding problems. To the best of our knowledge, this is the first work to tackle the generation of domain-specific heuristics by employing Deep Learning techniques.

Technical Approach

In this section we will detail our intended approach and further elaborate on the chosen methods.

Graph Convolutional Networks

Graph Convolutional Networks were first introduced in (Bruna et al. 2013) and elaborated in (Defferrard, Bresson, and Vandergheynst 2016), where Graph Signal Processing techniques were employed for computing graph convolutions in the spectral domain. Graphs are mathematical structures capable of encoding complex pairwise data relationships, which can be exploited with Graph Theory techniques.

In (Kipf and Welling 2016), a simplification of spectral GCNs is proposed, with interesting results for citation networks classification. This approach makes use of Chebyshev polynomial approximations to reduce computational demands of graph convolution operations.

Proposed Method

Our proposed network for training consists of two Graph Convolutional layers with ReLU and Dropout, as described in Table 1.

The network is fed with plan graphs for a specific domain, where each node N represents a state S on the search tree.

Table 1: Proposed Network Architecture

Layer	Input	Output
Graph Conv	$\|F\|$	60
ReLU	-	-
Dropout	-	-
Graph Conv	60	1
ReLU	-	-

Each input graph is a subgraph of the whole tree, so that many different graphs are generated for a single problem.

Each node can be represented by a feature vector *F* encoding binary representations of every possible combination of predicates forming a state for that domain. For every predicate operator with arity *k*, there are o^k binary values concatenated to compose *F*, where *o* is the number of objects.

Goal states S^G are encoded with the same procedure described above. Their representations are added to each node observed for a task, and the mean average for every value is taken in order to grant weighting to predicates in accordance with the goal.

Perfect heuristic values h_i^* are attributed to each node and used as ground-truth. Thus, we can define a loss function using *l2* regularized Mean Squared Error between perfect and predicted heuristics made for each graph node:

$$L_h(g) = \sum_{i=1}^{N} \|\hat{h}_{i,g} - h^*_{i,g}\| + \lambda\|\Theta\|, \qquad (1)$$

where $\hat{h}_{i,g}$ is the computed heuristic value for node *i* in subgraph *g*, h_i^* is the perfect heuristic for node *i*, and *N* is the total number of nodes. λ and Θ refer to the *l2* regularization hyperparameter.

Since heuristic values are computed for every node, this design allows for the generated representations to be employed on traditional search methods, such as A-star.

Performance can be evaluated by comparing GCN heuristic value outputs with off-the-shelf heuristic functions, such as *Blind* or *LM-Cut*. Possible metrics are the number of search expansions required for obtaining a plan, number of failed plans, and average plan length.

One possible issue could be related to performance when generating graphs for consuming task instances. To tackle this, alternative methods for computing adjacencies could be devised, such as using backward search starting from the goal state or using greedy search methods for generating heuristic values.

Additional methods could also be employed for graph analysis and encoding. For instance, instead of binary values, adjacency matrices could be composed of heuristic values for each node. Regarding analysis, information could be gained by employing different graph metrics such as node centrality and connectivity.

Identifying dead-ends and landmarks to inform search could also improve $\hat{h}$ results. The feasibility of implementing these features within such a restricted schedule will still be assessed. For the same reason, possible network variations in architecture and hyperparameter definition will also be regarded as final concerns.

Project Management

We already have a functioning implementation of the described GCN in Pytorch capable of running experiments for various different domains. For better evaluating our results, our experiments will be restricted to varying instances of Blocksworld-4ops and Logistics domains.

Our schedule for the 5 remaining weeks will be divided as follows:

- Week 1: Running experiments and setting benchmarks for current network state according to defined metrics.
- Weeks 2-4: Heuristic values encoding in adjacency matrices will be tested. We will evaluate the relevance of different graph metrics to our problem and explore further less-consuming options for graph generation. The identification of dead-ends and landmarks will also be studied. As a final step, network hyperparameters will be tuned to improve results.
- Week 5: We will assess obtained results and write the final version of the paper.

Conclusion

In this work, we aim to investigate the possible use of Graph Convolutional Networks for learning domain-specific heuristics, a very pressing matter for facilitating planning in complex scenarios. We believe the methods herein described can provide important contributions to the task at hand and hope to achieve satisficing results for the two selected domains by the end of the semester.

References

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral networks and locally connected networks on graphs. *CoRR* abs/1312.6203.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR* abs/1606.09375.

Jiménez, S.; De La Rosa, T.; Fernández, S.; Fernández, F.; and Borrajo, D. 2012. A review of machine learning for automated planning. *The Knowledge Engineering Review* 27(4):433–467.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*. 1097–1105.

Loreggia, A.; Malitsky, Y.; Samulowitz, H.; and Saraswat, V. 2016. Deep learning for algorithm portfolios. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, 1280–1286. AAAI Press.

Sigurdson, D., and Bulitko, V. 2017. Deep learning for real-time heuristic search algorithm selection.