Learning Search Heuristics by Graph Convolutional Networks

Pedro Ballester
pedro.ballester@acad.pucrs.br

Abstract

Automated Planning is highly dependent on the trade-off between high quality heuristics and computational time for computing them. Out-of-the-shelf and computationally inexpensive heuristics tend to work well on several domains but can be uninformative or expensive to compute on more peculiar scenarios. By exploiting new advances in Convolutional Neural Networks, such as Graph Convolutional Networks (GCN), we aim at learning heuristics focused on the domain by training on simple instances of the problem and thus generalizing for more demanding ones.

Introduction

Automated Planning is a widespread field and its applications range from general game playing (Genesereth, Love, and Pell 2005) to space exploration (Norris et al. 2005). A good planner can eventually provide plans, if one exists, in non-critical time. Among the most important aspects of a good planner is the correct choice of a heuristic.

There are several heuristics that work well in multiple application domains and depending on their characteristics [illegible] will always lead to a correct, commonly suboptimal, [illegible] However, in some specific scenarios, some off-the-shelf heuristics work poorly both in performance and plan quality, and urge the need to design domain specific ones.

Handcrafting a heuristic relies on both domain knowledge and planning knowledge, which leads to very limited possibility of design for complex scenarios. One alternative for such issue is, instead of depending on human knowledge on a task for designing precise heuristics, one can use a data-driven approach and develop them using Machine Learning algorithms.

Considering the complex data structures available for planning tasks, a possible path for learning in this situation is using Deep Learning. Deep Learning has been popularized by its applications on unstructured data such as image and text. More recently has also been applied to complex structures such as graphs, with promising results (Kipf and Welling 2016).

Released in 2012 with AlexNet (Krizhevsky, Sutskever, and Hinton 2012), Convolutional Neural Networks (Deep Learning most famous neural network design and known as CNNs) set record results in ImageNet classification task (Deng et al. 2009; Russakovsky et al. 2015). Several other architectures have been proposed after, such as GoogleNet (Szegedy et al. 2015) with inception layers and ResNet (He et al. 2016) with residual layers. Each one of them add design choices that further enhance learning capabilities of Convolutional Neural Networks.

To explore other types of data, architectures that aimed at graphs, instead of images were also proposed (Kipf and Welling 2016; Defferrard, Bresson, and Vandergheynst 2016), called Graph Convolutional Networks (GCNs). This would allow Convolutional Neural Networks to capture neighbourhood information and properly handle this kind of data.

To address the problem of domain specific heuristic design for complex scenarios, this work proposes the use of GCNs for learning heuristics. W[illegible] want to address transferability and how to classify such heuristics according to planning literature.

Related Work

Long before Deep Learning, traditional Machine Learning was already being used for Planning tasks. There are studies with a thorough review of such methods and addressing them is out of the scope of this work (Jiménez et al. 2012).

Some works deploy Deep Learning in an offline matter to choose a search heuristic that best suites the task (Sigurdson and Bulitko 2017; Loreggia et al. 2016).

To the best of our knowledge this work is the first to deploy Deep Learning to find new search heuristics tailored for the problem. We will improve the related work section for the next iteration of this assignment.

Graph Convolutional Networks

Graph Convolutional Networks (GCN) (Kipf and Welling 2016) are Convolutional Neural Networks that work directly on graphs and are able to take advantage on specificities of how graphs are encoded. Graphs can be dense or sparse and encode relationship between different nodes in a different perspective as traditional Convolutional Neural Networks expect (e.g. for images and text). Whereas traditional CNNs would look for local relationship by observing proximity in the input matrix, GCNs aim at understanding the

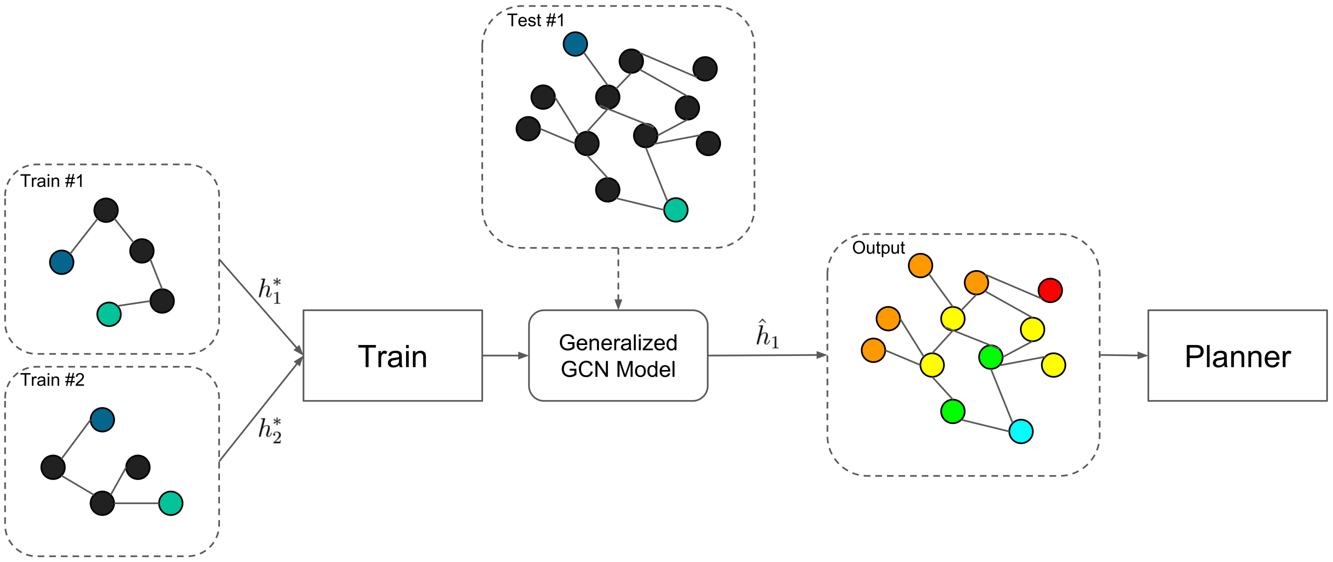

Figure 1: Training and testing protocol for learning heuristics with examples from a task. Both Train #1 and #2 are non-consuming examples from the task and Test #1 is a hard to compute example that is forwarded to the network to extract heuristic values for each state expanded. The train input h_1^* and h_2^* represent the perfect heuristic for train instance 1 and 2. The output $\hat{h}_1$ is the output heuristic value for each node of test instance 1. Best viewed in color.

node neighbourhood instead. This is a key difference considering that most common graph encoding do not group neighbourhood by the node i, j position distance in the matrix.

Formally ~~defining~~, a GCN is a Convolutional Network that takes as input a graph [illegible] this case represented as a NxN adjancency matrix b[illegible] the number of nodes and a NxF matrix, being F the [illegible] of the feature encoding for each node. The output from the network is a node-wise value NxO being O the size of output for each node.

Graph Convolutional Net[illegible] rely on spectral graph theory, thus designing spectr[illegible]volutions. Simplifications of such operations speed [illegible]aining and allow for non-prohibitive computation time for large graphs.

Method and Investigation

There are several aspects that should be considered in order to best represent the task. We aim at answering every question by the end of the semester alongside proposing the full method with enough experiments to measure possible contributions to the field.

1. How to evaluate the network loss;
2. How to encode state and goals;
3. How to penalize for dead ends;
4. How to hand[illegible]lmarks;
5. How the size of N impact the overall performance and efficacy of the generated model;
6. How to handle when the start and goal state cannot be represented in the same graph due to memory limitations;
7. Investigate how one can classify the generated heuristic at its definitions (e.g. whether it is Safe, Goal-Aware, Admissible, and Consistent);
8. How learned heuristics generalize to different domains and how adaptation between task domains improves results and training speed against training from scratch.

We already conjecture some of the possible choices that could potentially lead to satisfacing results. Regarding the loss function, we first aim at minimizing the quadratic difference between the computed heuristic by the neural network and the perfect heuristic computed in easy, non-consuming insta[illegible]f the problem:

$$L_h = \sum_{i=1}^{N} ||\hat{h}_i - h_i^*||_2 \qquad (1)$$

where $\hat{h}_i$ is the computed heuristic value by the neural network for a node in the graph, h_i^* is the perfect heuristic outcome for a node, and N is the total number of nodes.

A possible limitation of this approach is finding [illegible] consuming instances of a given task, as not all problem[illegible] be trivially simplified as to find easy e[illegible]es for training the network. One could potentially trai[illegible]rge examples, but this would drastically impact the [illegible]rmance of the training as generating full graphs for hard examples alongside computing the perfect heuristic can be very demanding.

The goals and current state should be somewhat encoded in a similar form as in traditional planning. With each node representing a state, one could include the information in a binary vector F that represents the existence of a determined condition at a given state.

Penalizing dead-ends and increasing the heuristic value for landmarks could potentially enhance $\hat{h}$. As a secondary objective, we will try to understand whether this kind of additional information during training can lead to better heuristics or decrease the amount of training data necessary for satisfactory heuristics.

Memory constraints are a big issue in planning. This approach requires the expansion of the graph for N adjacent states. In the case that the goal and start state cannot be represented in the same graph, we will explore alternatives as to how to best handle this problem. One way is to model the goal state separately from the graph and generate heuristic values for all states, choose the one that minimizes the loss and recompute the adjacent states for the chosen node. This process would be repeated until the goal state is reached.

Some relevant information for the generated heuristic is how it is defined in the space of possible heuristics. It is important for engineering and mathematical proofs purpose that one know, for example, whether the generated heuristic is Safe, Goal-Aware and other definitions. We will explore how one can try to infer those characteristics from data and by measuring boundaries for a specific domain. We believe that guaranteeing those definitions for any domain for a given heuristic will be harder due to hard to interpret internal operations of neural networks.

Finally, we will see how the generated heuristic generalizes for new domains. There are several ways we can observe this effect. First, we will run the generated heuristic for new domains and check how it behaves. As a next step, we can compare training the network from scratch and tuning in both unsupervised and supervised manners for a new domain.

The first draft of the proposed method can be seen in Figure 1. By the end of the semester we intend on validating the framework and make the necessary changes for it to work properly.

Material and Technical Details

Our work will be implemented using the available library PyperPlan[1]. We also found an implementation of GCN in PyTorch [2]. The available implementation has some differences in respect to the original code in TensorFlow, but PyTorch allows for easier prototyping and experimentation, and thus is the code of choice for the assignment due to time constraints.

References

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, 3844–3852.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 248–255. Ieee.

Genesereth, M.; Love, N.; and Pell, B. 2005. General game playing: Overview of the aaai competition. *AI magazine* 26(2):62.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

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*, 1097–1105.

Loreggia, A.; Malitsky, Y.; Samulowitz, H.; and Saraswat, V. A. 2016. Deep learning for algorithm portfolios. In *AAAI*, 1280–1286.

Norris, J. S.; Powell, M. W.; Vona, M. A.; Backes, P. G.; and Wick, J. V. 2005. Mars exploration rover operations with the science activity planner. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 4618–4623. IEEE.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115(3):211–252.

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

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

[1]https://bitbucket.org/malte/pyperplan

[2]https://github.com/tkipf/pygcn