A Discussion on the Scalability of Heuristic Approximators (Extended Abstract)

Sumedh Pendurkar,[1] Taoan Huang, [2] Sven Koenig, [2] Guni Sharon [1]

[1] Texas A&M University
[2] University of Sourthern California
sumedhpendurkar@tamu.edu, taoanhua@usc.edu, skoenig@usc.edu, guni@tamu.edu

Abstract

In this work, we examine a line of recent publications that propose to use deep neural networks to approximate the goal distances of states for heuristic search. We present a first step toward showing that this work suffers from inherent scalability limitations since — under the assumption that P≠NP — such approaches require network sizes that scale exponentially in the number of states to achieve the necessary (high) approximation accuracy.

Introduction

Principal computational problems, such as planning and scheduling (Wilkins 2014), routing (Toth and Vigo 2002), and combinatorial optimization (Papadimitriou and Steiglitz 1998), are known to be NP-hard in their general forms. There are no polynomial-time algorithms known for solving them. This complexity gap, known as the P vs. NP problem (Cook 2006), remains one of the biggest open questions in computer science.

Recognizing the challenges of developing polynomial-time solvers (or their unattainability), many researchers focus on reducing the exponential runtime of known solvers using heuristic functions (Pearl 1984). For example, it is easy to show that, given a heuristic function that closely approximates the goal distances of states, the A* algorithm (Hart, Nilsson, and Raphael 1968) can solve NP-hard search problems with a runtime that is polynomial in the length of the solution path and the branching factor. This fact motivates a line of publications (McAleer et al. 2018; Agostinelli et al. 2019, 2021) that proposes methods for approximating the goal distances of states for heuristic search using universal function approximators. These methods achieve state-of-the-art results for the studied domains. However, we present a first step toward showing that they suffer from inherent scalability limitations since — under the assumption that P≠NP — they require function approximator sizes that scale exponentially in the number of states to achieve the necessary (high) approximation accuracy.

Space limitations prevent us from providing the needed theoretical definitions and proofs. These will be provided in a future publication. Instead, we present only supporting experimental results in this paper.

Experiments

In our experiments, we use neural networks as universal function approximators and determine their smallest sizes, expressed in the numbers of parameters (namely, their weights and bias parameters), required to fit fixed-size training sets. The neural networks are trained to maps states to their goal distances for instances of given sizes in three search domains, namely the pancake, travelling salesman (TSP), and blocks world domain.

Setup

We use artificial neural networks of two different structures as universal function approximators (Cybenko 1989; Kidger and Lyons 2020), both using *Rectified Linear Unit* (ReLU) activation functions (Nair and Hinton 2010) since they satisfy the properties required by the universal function approximation theorem (Sonoda and Murata 2017). First, we use two-layer neural networks (that is, neural networks with one hidden layer). We refer to these networks as fixed-depth networks since any number of neurons in the hidden layer is allowed, but the number of layers is fixed. Second, we use neural networks with a fixed number of neurons per hidden layer, but any number of layers is allowed. We refer to these networks as fixed-width networks. We use residual connections (He et al. 2016)[1] and batch normalization (Ioffe and Szegedy 2015) to mitigate issues that we observed during training, such as vanishing gradients.

The number of states increases exponentially in the size of the instances, that is, the number of pancakes, cities, or blocks. It is therefore unrealistic to train on all states and their goal distances. We thus randomly select one million states and their goal distances (obtained by running an optimal search algorithm) for each instance size and use 80 percent of them for training and the remaining 20 percent for testing. We use the ADAM optimizer (Kingma and Ba 2015) to minimize the mean squared error on the training set, following (Agostinelli et al. 2019, 2021). ADAM is not guaranteed to converge to the global minimum. However, it is widely used in the literature (Agostinelli et al. 2019; Orseau and Lelis 2021) since it usually results in near optimal solutions. We stop training when a neural network fits

[1]Residual networks are universal function approximators (Lin and Jegelka 2018).

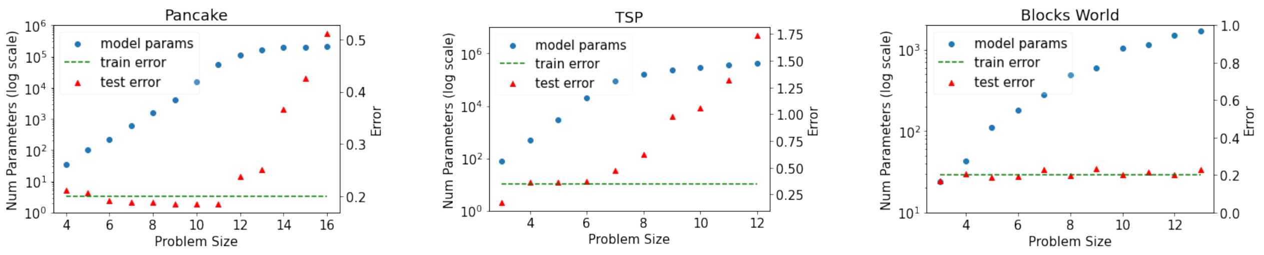

Figure 1: The smallest number of parameters (in log scale) required for fixed-depth networks to fit the training set for instances of increasing sizes. On the x-axis, we have the sizes of the instances for each search domain, namely the number of pancakes, cities, or blocks. On the y-axis to the left, we have the number of parameters (in log scale). On the y-axis to the right, we have the training and test errors. The training error thresholds are 0.20, 0.35, and 0.20 for the pancake, TSP, and blocks world instances, respectively.

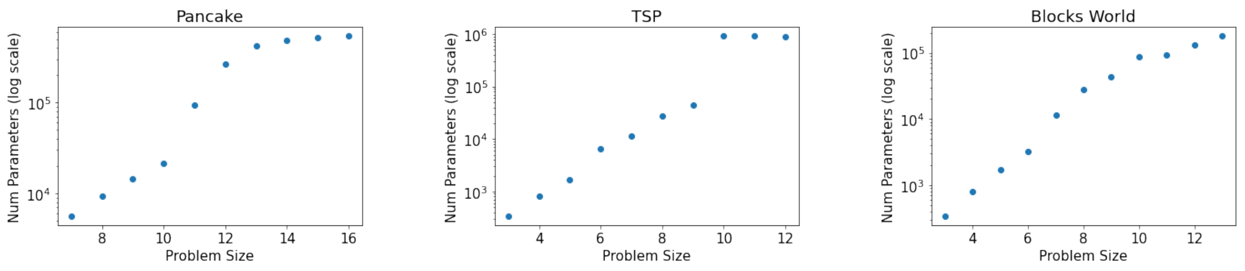

Figure 2: The smallest number of parameters (in log scale) required for fixed-width networks to fit the training set for instances of increasing sizes. On the x-axis, we have the sizes of the instances for each search domain, namely the number of pancakes, cities, or blocks. On the y-axis, we have the number of parameters (in log scale). The training error thresholds are 0.10, 0.35, and 0.02 for the pancake, TSP, and blocks world instances, respectively.

the training set, defined as the training error dropping below a given training error threshold, or after 300 epochs, whatever occurs earlier. To mitigate the impact of local minima, we train each neural network five times and report the lowest training error. Details will be provided in the full-length publication of this work.

Results

Figure 1 reports the smallest number of parameters required for fixed-depth networks to fit the training set for instances of increasing sizes. For pancake and TSP instances, we see a linear trend until a certain instance size and then a stagnation around pancake instances of size 12 and TSP instances of size 7. The test error starts to increase roughly at the point of stagnation – a common indicator for overfitting, which we did not prevent in our experiments other than by stopping training early. One explanation for the increasing test error is that the training and test sets contain fewer and fewer common states as the instance size increases since the number of states increases with the instance size and it thus becomes more and more unlikely that the same state will be part of both the training and test sets. Figure 2 shows similar patterns for fixed-width networks although the plots are noisier than those in Figure 1 since adding another layer with n neurons increases the number of parameters by n^2 weights and n bias parameters, making it more challenging to obtain smooth graphs. Overall, the smallest number of parameters is similar for fixed-depth and fixed-width networks at the point of stagnation, namely about 5×10^5. Overall, our results suggest that sufficiently accurate heuristic functions for these NP-hard problems might not have a meaningful underlying structure. As a result, the scalability of learning such heuristic functions might be similar to that of constructing a naive lookup table, which grows exponentially with the instance size.

Summary

We empirically investigated the unscalability of heuristic approximators for NP-hard search problems. Our experimental results provide a first indication that heuristic search algorithms that rely on function approximators to fit the goal distances for NP-hard search problems are unscalable. The experimental results for blocksworld instances, though, need to get investigated further (for example, for even larger instance sizes) since the test error remained small for the instance sizes used in Figure 1.

References

Agostinelli, F.; McAleer, S.; Shmakov, A.; and Baldi, P. 2019. Solving the Rubik's Cube with Deep Reinforcement Learning and Search. *Nature Machine Intelligence*, 1(8): 356–363.

Agostinelli, F.; Shmakov, A.; McAleer, S.; Fox, R.; and Baldi, P. 2021. A* Search Without Expansions: Learning

Heuristic Functions with Deep Q-Networks. *arXiv preprint arXiv:2102.04518*.

Cook, S. 2006. The P versus NP Problem. *The Millennium Prize Problems*, 87–104.

Cybenko, G. 1989. Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals and Systems*, 2(4): 303–314.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, 448–456.

Kidger, P.; and Lyons, T. 2020. Universal Approximation with Deep Narrow Networks. In *Conference on Learning Theory*, 2306–2327.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

Lin, H.; and Jegelka, S. 2018. Resnet with One-Neuron Hidden Layers is a Universal Approximator. *Advances in Neural Information Processing Systems*, 6172–6181.

McAleer, S.; Agostinelli, F.; Shmakov, A.; and Baldi, P. 2018. Solving the Rubik's Cube with Approximate Policy Iteration. In *International Conference on Learning Representations*.

Nair, V.; and Hinton, G. E. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *International Conference on Machine Learning*, 807—–814.

Orseau, L.; and Lelis, L. H. 2021. Policy-Guided Heuristic Search with Guarantees. In *AAAI Conference on Artificial Intelligence*, 12382–12390.

Papadimitriou, C. H.; and Steiglitz, K. 1998. *Combinatorial Optimization: Algorithms and Complexity*. Courier Corporation.

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc. ISBN 0201055945.

Sonoda, S.; and Murata, N. 2017. Neural Network with Unbounded Activation Functions is Universal Approximator. *Applied and Computational Harmonic Analysis*, 43(2): 233–268.

Toth, P.; and Vigo, D. 2002. *The Vehicle Routing Problem*. SIAM.

Wilkins, D. E. 2014. *Practical Planning: Extending the Classical AI Planning Paradigm*. Elsevier.