Neuroevolution of Augmenting Topologies

Paul Pauls
Advisor: Michael Adam

CONTENTS

Abstract—Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

I. INTRODUCTION

THIS shall be my introduction. And this shall be my citation [1]. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

II. NEUROEVOLUTION AND EVOLUTIONARY ALGORITHMS

Neuroevolution is a machine learning technique that applies evolutionary algorithms to construct artificial neural networks, taking inspiration from the evolution of biological nervous systems in nature. [2]

A evolutionary algorithm is a generic population-based and meta-heuristically optimized algorithmic solution to an applied problem. When breaking this down to simpler terms, does this mean first of all that an evolutionary algorithm (short form: **EA**) is simply a solution to an applied problem. This solution can take the form of a classical calculation algorithm or more complex forms like the aforementioned artificial neural network. As generic as its method of solving the problem is, is its application domain. Evolutionary Algorithms - as well as Neuroevolution by which they are employed - are highly general and allow for learning without explicit targets even if provided with only minimal feedback. [2]

Evolutionary Algorithms are population-based, meaning they not only handle a single solution to the problem they are applied to, but they have a multitude of solutions to this problem. Each of the solutions is called a *member* of its arbitrarily large population and each solution is arbitrarily similar to one another. Each member of the population is judged by a common *fitness function*, which numerically expresses the members quality of its solution to the applied problem. The higher the corresponding *fitness function* score of a member, the better is the solution to the applied problem. All members are judged by the same fitness function, which is the key hyperparameter that determines how well a problem has been solved.

The key aspect of evolutionary algorithms however lies in its meta-heuristic optimization method. This optimization method improves the members of the population in the sense that along the evolutionary process almost all members of the population score an increasingly higher fitness function evaluation score - meaning they get increasingly better at solving the problem they are applied to. This optimization method is employed after each *generation* of a population. A generation in the evolutionary process is completed once every member of the population has been applied to the problem and has been assigned a fitness score by the common fitness function. The population is then mixed up through *reproduction*, *mutation*, *recombination* and *selection*. These processes spread traits of high-performing members to low-performing members, preserve high-performing members while extinguishing low-performing members and introduce novel traits to already high-performing members to further explore the solution-space.

Applying the attributes of Evolutionary Algorithms to ar-

tificial Neural Networks does result in the machine learning technique Neuroevolution. Neuroevolution algorithms start out with an initial population of neural networks of varying complexity - meaning topology and weights. The exact nature of this initial population is not stipulated by the Neuroevolution technique and depends on the specific Neuroevolution algorithm. In each generation does each neural network member perform input inference and is subsequently judged by the fitness function. The best scoring neural networks spread their traits to lower scoring neural network. Fixed-Topology Neuroevolution algorithms solely adjust the weights of the lower scoring neural networks towards the higher scoring networks while Topology and Weight Evolving Artificial Neural Network algorithms (short form: **TWEANNs**) additionally also adjust the topology. They introduce new nodes and connections or delete nodes and connections in the low scoring neural networks depending on if they are present in the better scoring neural networks or introduce completely novel nodes and connections in order to explore new, possibly even better performing, solutions. Through this do neural networks developed through Neuroevolution algorithms often have arbitrary neural models and network structures and are able to progress in learning even without explicit targets and only sparse feedback.

While concluding this introduction to Neuroevolution can be said that Neuroevolution can be classified as part of the machine learning paradigm of *Reinforcement Learning*. Neuroevolution algorithms learn through ever-changing interaction with the environment instead of a predetermined set of input-output mapping or creative self-organization of input through probabilistic models. Neuroevolution algorithms learn from the consequence of their actions on the basis of their past experiences and also by new choices - quite accurately conforming with the definition of Reinforcement learning.

A. Research in Neuroevolution

III. NEUROEVOLUTION OF AUGMENTING TOPOLOGIES (NEAT)

A. ¡Section Introduction¿

B. Key Aspects of NEAT and Differences to Preceding Neuroevolution

C. Performance of NEAT

D. Variants and Advancements of NEAT

IV. APPLICATIONS OF NEAT

V. CONCLUSIONS

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit

blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

REFERENCES

[1] Example Cite, *Source*, Apr. 2019. www.example.com
[2] http://www.scholarpedia.org/article/Neuroevolution
[3] https://www.oreilly.com/ideas/neuroevolution-a-different-kind-of-deep-learning
[4] https://eng.uber.com/deep-neuroevolution/
[5] https://en.wikipedia.org/wiki/Evolutionary_algorithm
[6] https://en.wikipedia.org/wiki/Genetic_algorithm
[7] https://en.wikipedia.org/wiki/Neuroevolution
[8] https://en.wikipedia.org/wiki/Neuroevolution_of_augmenting_topologies
[9] https://en.wikipedia.org/wiki/Reinforcement_learning
[10] http://www.scholarpedia.org/article/Neuroevolution
[11] http://www.scholarpedia.org/article/Reinforcement_learning