Configuring the Parameters of Artificial Neural Networks using NeuroEvolution and Automatic Algorithm Configuration

Evgenia Papavasileiou*
(1) Vrije Universiteit Brussel (VUB), Department of Electronics and Informatics (ETRO)
(2) imec
epapavas@etrovub.be

Bart Jansen
(1) Vrije Universiteit Brussel (VUB), Department of Electronics and Informatics (ETRO)
(2) imec
bjansen@etrovub.be

ABSTRACT

NeuroEvolution (NE) is a powerful method that uses Evolutionary Algorithms (EAs) for learning Artificial Neural Networks (ANNs). However, NE's performance is determined by the definition of dozens of parameters that guide the search of the EAs. In this study we apply automatic algorithm configuration for tuning the parameters of a NE method in an offline matter. The tuned NE method is then used to evolve the weights, topology and activation functions of ANNs while performing feature selection and its performance is compared to the case of using default parameters. We show that tuning the parameters results in NE methods able to solve the problems with 100% accuracy in significantly less generations.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; **Genetic algorithms**;

KEYWORDS

NeuroEvolution, parameter tuning, activation function

ACM Reference Format:
Evgenia Papavasileiou and Bart Jansen. 2018. Configuring the Parameters of Artificial Neural Networks using NeuroEvolution and Automatic Algorithm Configuration. In *GECCO '18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3205651.3208766

1 INTRODUCTION

Artificial Neural Networks (ANNs), first introduced in 1943, are considered very powerful computing systems. NeuroEvolution (NE) [5] is a machine learning method that uses Evolutionary Algorithms (EAs) to construct ANNs. The connection weights of fixed topology ANNs were one of the first parameters to be optimized by NE [3, 7]. Succeeding algorithms optimized both the connection weights and the topology of the underlying nodes [11, 13]. However, the architecture of an ANN is not solely defined by the topology of the nodes but also by their activation functions [1, 2]. Towards this goal, Heterogeneous Activation NeuroEvolution of Augmenting Topologies (HA-NEAT) is developed as a method that enables the evolution of the activation functions in addition to the weights and the topology of the ANNs. Moreover, the performance of an ANN depends on the identification of the relevant inputs, especially nowadays with the plethora of high dimensional data. Finally, the performance of a NE algorithm depends on the appropriate definition of dozens of parameters that are usually defined by trial and error or by expert's knowledge.

In this study, we first extend HA-NEAT to perform feature selection simultaneously with the ANNs' architecture optimization and we call this method Heterogeneous Activation Feature De-selective NeuroEvolution of Augmenting Topologies (HA-FD-NEAT). Moreover, we apply automatic parameter configuration based on racing. To our knowledge, this is the first time automatic algorithm configuration is applied to a NEAT-based method. We demonstrate how the appropriate choice of parameters is crucial by evaluating how efficient the tuned HA-FD-NEAT algorithm becomes in terms of finding an ANN that classifies correctly unseen instances and how many generations are required to find this ANN.

*Research funded by a PhD grant of the Research Foundation Flanders (FWO)

2 METHODS

2.1 HA-NEAT

HA-NEAT [4] is a NE algorithm based on the NeuroEvolution of Augmenting Topologies (NEAT) [11] algorithm that employs its three main principles; historical markings to facilitate crossover, speciation to protect innovations and topologies' complexification. HA-NEAT evolves the structure of ANNs by mutation and crossover operations. Structural mutations enable the addition of new nodes and connections, while mutation operators change the weights of existing connections and the nodes' activation functions.

2.2 IRACE

Irace [6] is a parameter configuration algorithm that is based on sampling configurations of parameters according to an associated distribution, select the best configurations by means of racing and update the distributions at the end of the race. Irace begins by evaluating candidate configurations on a set of problem instances based on a cost function returned by the configurable algorithm. After some iterations, a statistical test is performed in order to discard configurations that resulted in poor performance and race continues with the surviving ones. At the end of one race the elite configurations are selected and new configurations are sampled in order to perform a new race from a smaller set of configurations.

2.3 Proposed Method

In this study we extend HA-NEAT to perform feature selection while optimizing the ANNs' connectivity, topology and activation functions. Our proposed method called HA-FD-NEAT is inspired by Feature De-selective-NEAT (FD-NEAT) [12]. In HA-FD-NEAT a new mutation operator that can act in the input layer by dropping input connections is introduced. In this way, HA-FD-NEAT performs implicit feature selection since only the relevant inputs tend to survive at the end of the evolution. The fitness function of HA-FD-NEAT is based on the error between the output of the ANN and the correct output of the training set as in [8–11].

2.4 Experimental Procedure

HA-FD-NEAT is evaluated on artificial datasets based on the XOR problem of increased complexity with irrelevant inputs as in [8–10]. The resulting $2/k$ XOR problems consist of the two relevant XOR inputs and $k-2$ irrelevant inputs, where $k \in \{20, 100\}$. Instances of the 2/20 and 2/100 XOR problems are given to irace separately for training and testing the candidate configurations.

As this study shows preliminary results, the values of only three of the configurable parameters of HA-FD-NEAT are optimized by irace; the size of the population, the probability of adding nodes and a compatibility coefficient defining speciation.

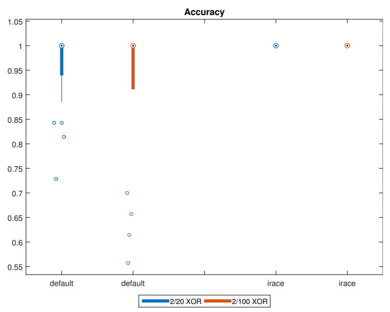
HA-FD-NEAT is allowed to evolve for maximum 1000 generations using two different settings of parameters; the default settings as these are found in the implementation of HA-NEAT [4] and the parameters discovered by irace. The different parameter settings are evaluated using the accuracy on unseen instances of the 2/20 and 2/100 XOR problems and the number of generations required to find the solution.

3 RESULTS

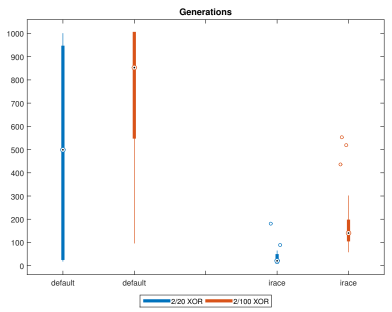
From the results in figure 1 it is evident that HA-FD-NEAT using the parameters returned by irace evolves ANNs that are able to solve the two $2/k$ XOR problems much more efficiently than when using arbitrarily defined parameters. In particular, HA-FD-NEAT using the default parameters requires 499 generations (IQR: 909) to solve the 2/20 XOR problem with median accuracy of 1 (IQR: 0.06), whereas using the parameters optimized by irace only takes 21 generations (IQR: 27.5) for an accuracy of 1 (IQR: 0). Similarly for the difficult 2/100 XOR problem the default parameters solve the problem with accuracy of 1 (IQR: 0.09) requiring 853 generations (IQR: 446) whereas using the parameters found by irace only takes 140 generations (IQR: 80) for an accuracy of 1 (IQR: 0).

4 CONCLUSIONS

The results of these experiments, although limited, constitute a clear evidence that the choice of the correct set of parameters can significantly influence the performance of a NE algorithm. In future work we will extend the experiments by configuring more parameters of the HA-FD-NEAT by irace. Finally, we plan to investigate how the parameter selection influences HA-FD-NEAT's feature selection ability by application to real world problems.



(a) Accuracy



(b) Generations

Figure 1: Accuracy and Generations for the XOR problems using default parameters and parameters found by irace.

REFERENCES

[1] Bhaskar DasGupta and Georg Schnitger. 1992. *Efficient approximation with neural networks: A comparison of gate functions.* Pennsylvania State University, Department of Computer Science.

[2] Wlodzislaw Duch and Norbert Jankowski. 2001. Transfer functions: hidden possibilities for better neural networks.. In *ESANN.* 81–94.

[3] Faustino Gomez and Risto Miikkulainen. 1997. Incremental evolution of complex general behavior. *Adaptive Behavior* 5, 3-4 (1997), 317–342.

[4] Alexander Hagg, Maximilian Mensing, and Alexander Asteroth. 2017. Evolving Parsimonious Networks by Mixing Activation Functions. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '17).* ACM, New York, NY, USA, 425–432. https://doi.org/10.1145/3071178.3071275

[5] Joel Lehman and Risto Miikkulainen. 2013. Neuroevolution. *Scholarpedia* 8, 6 (2013), 30977.

[6] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. 2016. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives* 3 (2016), 43–58.

[7] David E Moriarty and Risto Mikkulainen. 1996. Efficient reinforcement learning through symbiotic evolution. *Machine learning* 22, 1-3 (1996), 11–32.

[8] Evgenia Papavasileiou and Bart Jansen. 2016. A comparison between FS-NEAT and FD-NEAT and an investigation of different initial topologies for a classification task with irrelevant features. In *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on.* IEEE, 1–8.

[9] E. Papavasileiou and B. Jansen. 2017. The importance of the activation function in NeuroEvolution with FS-NEAT and FD-NEAT. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI).* 1–7. https://doi.org/10.1109/SSCI.2017.8285328

[10] Evgenia Papavasileiou and Bart Jansen. 2017. An Investigation of Topological Choices in FS-NEAT and FD-NEAT on XOR-based Problems of Increased Complexity. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '17).* ACM, New York, NY, USA, 1431–1434. https://doi.org/10.1145/3067695.3082497

[11] Kenneth O Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary Computation* 10, 2 (2002), 99–127.

[12] Maxine Tan, Michael Hartley, Michel Bister, and Rudi Deklerck. 2009. Automated feature selection in neuroevolution. *Evolutionary Intelligence* 1, 4 (2009), 271–292.

[13] Xin Yao and Yong Liu. 1998. Towards designing artificial neural networks by evolution. *Appl. Math. Comput.* 91, 1 (1998), 83–90.