

Distance-based Kernels for Surrogate Model-based Neuroevolution^{*}

Jörg Stork, Martin Zaeferrer, and Thomas Bartz-Beielstein

TH Köln, Institute for Data Science, Engineering, and Analytics,
Steinmüllerallee 1, 51643 Gummersbach, Germany,
joerg.stork|martin.zaefferer|thomas.bartz-beielstein
@th-koeln.de

Abstract. The topology optimization of artificial neural networks can be particularly difficult if the fitness evaluations require expensive experiments or simulations. For that reason, the optimization methods may need to be supported by surrogate models. We propose different distances for a suitable surrogate model, and compare them in a simple numerical test scenario.

Keywords: neuroevolution, surrogate models, kernel, distance, optimization

1 Introduction

A crucial and difficult task in the optimization of Artificial Neural Networks (ANN) is the optimization of their topology. Often, only parameters like the edge weights, number of hidden layers, and number of elements per layer are considered. More fine-grained and less restrictive changes to the network structure are rarely examined. Two notable exceptions are NeuroEvolution of Augmenting Topologies (NEAT) [1] and Cartesian Genetic Programming of Artificial Neural Networks (CGPANN) [2]. Both NEAT and CGPANN require numerous fitness evaluations, which makes them inefficient for applications with expensive function evaluations such as simulations or real-world experiments, e.g., when an ANN is the controller of a robot that operates in a complex real-time environment. In such cases, it would be desirable to employ surrogate models that reduce the load of fitness evaluations. While surrogate models are frequently employed in continuous optimization, more complex discrete search spaces are less often investigated [3]. Recently, a first surrogate model for ANN optimization was applied to NEAT by Gaier et al. [4]. They use a distance-based surrogate model, employing a genotypic compatibility distance that is part of NEAT. We will focus on investigating the effect of different distances on the search performance of a surrogate model, optimizing ANNs with CGPANN. We compare distances that take the genotype (network structure) and/or phenotype (behavior) of the ANNs in account. A transparent and simple numerical test case is used to evaluate the performance of the different distances.

^{*} This publication was accepted to the Developmental Neural Networks Workshop of the Parallel Problem Solving from Nature 2018 (PPSN XV) conference

2 Surrogate Modeling for Neuroevolution

We use a Surrogate Model-based Optimization (SMBO) algorithm, following the idea of Efficient Global Optimization (EGO) [5]. In short, a Kriging model is constructed based on an initial candidate data set, which is used in each iteration to suggest new promising candidates by optimizing the expected improvement criterion. Kriging approximates the data by modeling the similarities of samples via kernels and distances. We employ the Kriging implementation of the R package `CEGO` [6, 7]. It uses distance-based kernels to model data from structured, combinatorial search spaces. We utilize different genotypic and phenotypic distance measures for ANNs, see Section 3. For CGPANN, we use the C library `CGP` by A. Turner¹, which also allows the application of neuroevolution [8]. `CGP` was modified with function interfaces to R, distance measures, and fitness functions. The genotype of a CGPANN individual consists of a fixed number of nodes. Each node has a number of connection genes based on the pre-defined arity with adjacent weight genes and a single categorical function gene. Nodes are connected to preceding nodes. Duplicate connections to nodes are possible. Moreover, each node has a Boolean activity gene, which signals if it is used in the active phenotype. CGPANN utilizes a (1+4)-Evolution Strategy.

3 Proposed Kernels and Distances

In the following, we always use an exponential kernel $k(x, x') = \exp(-\theta d(x, x'))$. Here, x and x' represent the optimized ANNs, and consist of the weights x_w , input labels x_i , activity labels x_a , and transfer function labels x_f , i.e., $x = \{x_w, x_i, x_a, x_f\}$. All distances $d(x, x')$ are scaled to $[0, 1]$.

Genotypic Distance (GD): The simple genotypic distance combines the distances of weights, inputs, activity of nodes, and transfer functions. The inputs are sorted for each x before computation of the distance $d(x, x') = \|x_w - x'_w\|^2 + H(x_i, x'_i) + H(x_a, x'_a) + H(x_f, x'_f)$, where $H(a, b)$ denotes the Hamming distance, i.e., the transfer function distance is 0 if the function is identical, else 1.

Genotypic ID Distance (GIDD): A distance based on the active topology, utilizing IDs for each node. Each active node in the ANN is given an ID based on the connections to prior nodes or inputs and the number of non-duplicate connections of this node. The distance is calculated by a pairwise comparison of all node IDs. If a node ID is present in both ANNs, the subgraph prior to this node is analyzed recursively. If all IDs in the subgraph also match, we assume that the corresponding nodes have an equal position in the ANN structure. For all nodes that are matched in this way, the Euclidean distance of the weights (x_w) and Hamming distance of the transfer functions (x_f) is computed. A node pair can only be used once for this comparison, as node IDs may be present several times in each individual. If all node IDs of both individuals x and x' are equal, the GIDD is simply the distance between all weights and transfer functions.

¹ <http://www.cgplibrary.co.uk> - accessed: 2018-01-12

Phenotypic Distance (PD): A data set is used to compute the outputs of each ANN. For each sample, the *softmax* of the outputs is computed, which yields the class probabilities. The PD is the Euclidean distance of all softmax values. Different data sets can be created, e.g., based on a subset of the original task data set. Else, a distinct data set can be created by design of experiment methods. We also considered employing the graph edit distance, but decided against it due to its complexity. Computing the graph edit distance is NP-hard [9].

4 Experiments

Benchmark Task and Setup: The benchmark task is the neuroevolution of an ANN with a small budget of 250 total evaluations, which provides a realistic scenario for problems with expensive fitness evaluations. The well-known IRIS data set ($n = 150$ samples, 4 variables, and 3 classes), is used as an elementary benchmark problem. The fitness function is the adjusted classification accuracy: $\text{acc} = \sum_{i=1}^n a_i$, where $a_i = 1$ if the predicted class is true, otherwise, a_i is the predicted probability for the true class. In this preliminary study, the data set was not partitioned into train and test data. As baselines, we used random search and the original CGPANN with different mutation rates. Moreover, the above described distance measures are compared. For the PD, three sample methods are used: the complete data set (baseline) and two Response Surface Method designs (RSM) based on the IRIS parameter boundaries with a small (15) and a large (60) sample size. In SMBO, the (1+4)-ES is used in each consecutive iteration alternating between exploiting (L) and exploring (G). Parameters are listed in Table 1. Algorithm parameters were not tuned.

Results and Discussion: Figure 1 visualizes the results. With the given small number of evaluations, CGPANN obtained a better median value than random search. The genotypic distances performed similarly to CGPANN, while the PD distances performed better. An analysis of the convergence (not shown here) indicates a tendency to end up at a local optimum around an accuracy of 66%, which can be explained by the underlying benchmark problem with three classes. Importantly, even the small RSM set performs only slightly worse than the base set. The poor performance of GD and GIDD can be explained by the fact that even smallest changes in the genes can have a large impact on the ANN output. In contrast, the PD distances directly exploit the ANN behavior.

Table 1. Parameter setup, where evaluations denote the initial candidates plus the budget for consecutive evaluations. 40 Nodes were used to keep the search space small, but sufficient for the IRIS problem given this small budget.

arity	nodes	weight range	function set
5	40	[-1,1]	tanh, softsign, step, sigmoid, gauss
method	mutation rate	evaluations	surrogate evaluations
Random		250	
CGPANN	5%/15%	1 + 4 · 63	
SMBO	L:5% G:15%	50+200	L:10+400 G:1000+400

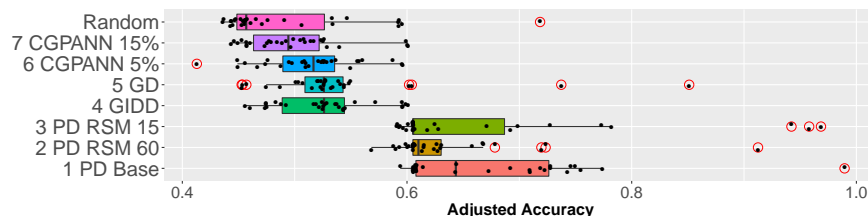


Fig. 1. Results after 250 fitness evaluations with 30 replications, comparing random search, CGPANN and different SMBO-CGP variants. Red circles depict outliers.

5 Conclusion and Outlook

In this work, we have shown that SMBO is a promising extension for CGPANN, that is able to improve the neuroevolution of ANNs in case of small evaluation budgets. SMBO with phenotypic distance kernels, which are based on generated outputs of each ANN, shows significantly better results than with genotypic distances or basic CGPANN. In future work we will investigate a larger set of tasks, generalization ability, longer runs, parameter tuning, and a more thorough investigation of the promising phenotypic distance.

Acknowledgements: This work is part of a project that has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreement no. 692286.

References

1. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. *Evolutionary computation* **10**(2) (2002) 99–127
2. Turner, A.J., Miller, J.F.: Cartesian genetic programming encoded artificial neural networks: a comparison using three benchmarks. In: *Proc. GECCO’13, ACM* (2013) 1005–1012
3. Bartz-Beielstein, T., Zaefferer, M.: Model-based methods for continuous and discrete global optimization. *Applied Soft Computing* **55** (2017) 154 – 167
4. Gaier, A., Asteroth, A., Mouret, J.B.: Data-efficient neuroevolution with kernel-based surrogate models. *arXiv* (2018)
5. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. *Journal of Global Optimization* **13**(4) (1998) 455–492
6. Zaefferer, M.: Combinatorial efficient global optimization in R - CEGO v2.2.0. online: <https://cran.r-project.org/package=CEGO> (2017) accessed: 2018-01-10.
7. Zaefferer, M., Stork, J., Friese, M., Fischbach, A., Naujoks, B., Bartz-Beielstein, T.: Efficient global optimization for combinatorial problems. In: *Proc. GECCO’14, ACM* (2014) 871–878
8. Turner, A.J., Miller, J.F.: Introducing a cross platform open source cartesian genetic programming library. *Genetic Programming and Evolvable Machines* **16**(1) (2015) 83–91
9. Zeng, Z., Tung, A.K.H., Wang, J., Feng, J., Zhou, L.: Comparing stars: On approximating graph edit distance. *Proc. VLDB Endow.* **2**(1) (2009) 25–36