Optimizing the structure of Neural Network using Differential Evolution

Canberk Özen, Heidi Carolina Martinsaari, Wai Tik Chan[1]

[1]Institute of Computer Science, University of Tartu

UNIVERSITY OF TARTU

1632

Introduction

Differential evolution (DE) is a population-based metaheuristic search algorithm [2]. It is one of the evolutionary algorithms which are inspired from biology and the theory of evolution. The idea of the optimization task is to find the best individual from the generated populations. The initial population is treated as the input for the algorithm. The individuals can be any simple functions to complex structures. Actually, there are almost no restrictions for the input space. In this project, we examined the performance of differential evolution for optimizing neural networks (NN). In addition to standard DE we compared it with its different variants.

Algorithm

DE is derived from standard genetic algorithm (GA) having some differences. It doesn't need encoded chromosomes allowing more general input. But as all GAs it is a good option for non-differentiable problems. DE is also suitable for gradient vanishing and explosion issue that is a headache problem when NN goes deeper. While handling complex structures the algorithm itself is simple having same steps as standard GA but in different order. The **input** consists of randomly generated population. As our goal is to find the best performing neural network, we treat the NNs as individuals. Already the initial population is evaluated with the **fitness** of each individual. The value of the fitness will be optimized - either minimized or maximized. In this project we used the binary cross entropy (BCE) loss for the value of the NN goodness. Calculated loss is a result of training and validating the NN. Then the population is traversed for obtaining the offspring per each individual. With the help of **mutation** process it is created a unitvector which goal is to mutate genes - speed up the convergence process or allow more exploration. In our case the genes are the number of layers and neurons per layer. The unitvector is **crossed over** with the parent to obtain a child. Last step is to select between the parent and its child. The **survival** has better fitness.

DE variants

In addition to the advantages brought out in previous section, one simplicity of DE is having only 3 hyperparameters to tune: the mutation factor (β), crossover probability (CR) and population size. Intuitively, based on our results it seems that population size is the dominating factor. However, when the population size is getting larger the impact of the mutation parameter β and the CR is growing, and it is how different types of DE arise. These variants are driven by the goal of search space size and the speed, on one hand to speed up the convergence memorizing the better solutions, and on the other to let explore more in the search space [1], [4], [3]. The following table compares the three DE variants we used:

	JDE	JADE	SaDE
Beta	Current β vs predefined probabilities	Current β vs predefined probabilities	From $N(0.5, 0.3)$, where $\beta \in (0, -2]$
CR	Current CR vs predefined probabilities	None	After 5 generations from $N(mean(CR), 0.1)$
Mutation strategy	DE Rand 1 z	DE pBest 1 z	DE Rand 1 z vs DE Best 1 z

Crossover is done similarly for all variants using CR, the crossover probability to choose genes from unitvector. Rest genes are inherited from parent.

Experiment Setup

The experiment was conducted with MINST dataset using the DE methods JDE, JADE and SaDE tuning the **number of hidden layers** and the **number of neurons** per each layer. Each variant was tested with 30 sets of different settings combining the maximum allowed number of layers and maximum number of neurons from 4 to 12 with step 2. While generating the initial population these two parameters are chosen uniformly. The experiment do not set stopping criteria for iterations in order to explore and capture more behavior on each method. The fitness of the model based on validation BCE loss is the main indicator for evaluation, the best encountered loss is captured at the end of each test with its settings along with following data for further analysis:

1. The best validation loss of the generation
2. The structure of neural network of the best validation
3. Run time in seconds of the population generation
4. The number of generation where the best individual is discovered

Analysis

The diagram below shows that all those three methods archive roughly the same quality of validation loss as the constraints on the network structure is relaxed. However, it is noticed that if connecting those points by method with respect to the NN depth, the JDE is always on top of the others.

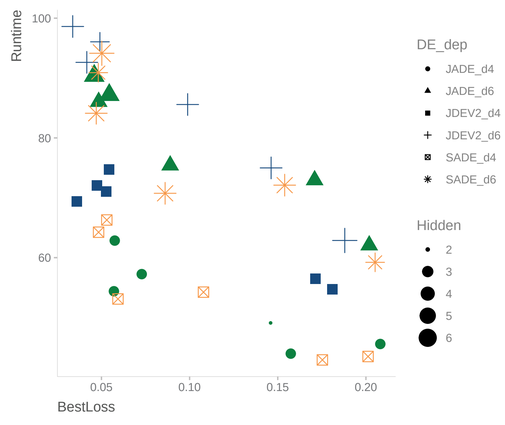

Fig. 1: Loss versus Running time of the best results of DE variants

This indicates that JDE requires more runtime in order to discover optimal settings, but it should be kept in mind that among those three methods, JDE is the most frequent in reaching the best validation loss (fitness). The extra runtime required may show that JDE's search space is relatively larger than the others as the procedures are roughly the same. Also, JDE uses less random functions requiring less computations. Furthermore, the slope of JADE and SaDE is often more steeper than JDE (in the view from left to right) which suggests that for both JADE and SaDE, there is a sudden drop of the best loss in between relaxing structural restrictions. This phenomena may be connected to the fact that JADE and SaDE that comes with specialized mutation and crossover method will have more complex population at the early stage which provides a better basis for selecting individual with smaller loss.

Conclusion

Our experiments confirmed the DE variants that focus on β and CR enable a wider search space by keeping diversifying the population. On the other hand for those using different mutation/crossover strategies show a faster convergence finding the optimal settings are reply on the search direction given by the top fit individuals. The following histogram describes the frequencies of the generations.

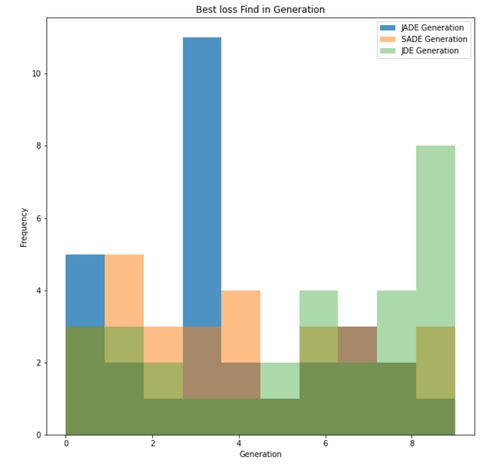

Fig. 2: Frequency on Generation found the best loss

We may see again that JDE finds the optimal at the very last generation while JADE usually finds the best at early stage. The mixture version - SaDE shows a relatively average statistics in between those two. In conclusion, DE variant which adopts the mutation/crossover schema will often provide a faster discovery of the approximate optimal parameters but will require more computations for the random functions compared to those adjusting β and crossover probability which need more time for searching but still finding the best settings.

Remarks

Experiment Environment: JDE and JADE: i7 3.2GHz (12 cores , 6 cores per method) SaDE: i5 3.0GHz (6 cores)
Repository with implementations and results: `https://github.com/tik65536/UT2021Algo_DE`

References

[1] J. Brest et al. "Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems". In: *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION* 10 (Dec. 2006), pp. 646–657.

[2] M. Georgioudakis and V. Plevris. "A Comparative Study of Differential Evolution Variants in Constrained Structural Optimization". In: *Front. Built Environ.* (July 2020). DOI: `https://www.frontiersin.org/articles/10.3389/fbuil.2020.00102/full`.

[3] A. K. Qin and P. N. Suganthan. "Self-adaptive Differential Evolution Algorithm for Numerical Optimization". In: *IEEE Congress Evolutionary Computation* 2 (Sept. 2005), pp. 1785–1791.

[4] J. Zhang and A. C. Sanderson. "JADE: Adaptive Differential Evolution with Optional External Archive". In: *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION* 13 (Oct. 2009).