Genetic Algorithm for Hyperparameter Tuning – Maximizing Convergence Speed and Ability

Xinqi Zhu

Research school of Computer Science, Australian National University
Canberra A.C.T 2601 Australia
e-mail: xinqi.zhu@anu.edu.au

Abstract. Genetic algorithm is an innovative optimisation approach while hyperparameter tuning of a model to match problem complexity has long been a challenging problem. This paper applies genetic algorithm to find the optimal hyperparameter setting of a feedforward neural network, where the optimum is defined on the convergence speed and ability of a model. The dataset used is a collection of results of a stress stimuli experiment. After experimenting with different genetic algorithm architectures, the found optimum has an increase of 24% of fitness value from baseline. Along with conducting the experiment, this paper also examines the famous exploration-exploitation trade-off in optimisation process for different genetic algorithm architectures by studying the statistical properties of the final population.

Keywords: genetic algorithm, hyperparameter tuning, feedforward neural network, convergence ability, convergence speed, exploration-exploitation trade-off

1 Introduction

1.1 Genetic algorithm

Model complexity is one key factor that influences the convergence and generalization ability of Feedforward Neural Networks (FNNs) trained by backpropagation. Model complexity is expected to highly match problem complexity to reach a good model. Hyperparameters are the factors that determine the model complexity. For FNNs, there are lots of hyperparameters that lay an important impact on complexity, such as network architecture (the number of hidden neurons, their connection topology etc.), the optimisation algorithm configuration (learning rate, weight decay etc.), it is arguably hard to find an optimal hyperparameter setting.

As for finding optimal network structure, there are three groups of approaches: one group performs a search over parameter spaces; one group simplifies from an overly complex model; one group progressively builds up a model [1]. In the previous submission, we examined the third group termed with constructive cascade network [8], especially one network structure named *casper* proposed by [1], [3] and conducted an experiment to maximize convergence ability and speed. This paper continues with the same target; however, we want to find the optimal joint setting of network architecture and training optimisation algorithm though one optimisation schema - genetic algorithm (GA).

GA, proposed by Holland in 1975 [13], is the most common category of Evolution computation (EC). EC is built upon Darwinian Evolution Theory and inspired by the theory of natural selection and survival of the fittest [2], while GA focuses on genetic evolution. Genetic algorithms provide a good alternative to traditional optimization techniques [12], it adopts the genetic information to perform a directed random search to find the optimal solution defined by fitness. The search for optimal hyperparameter setting of an FNN is an optimisation problem and fits naturally into the GA framework.

1.2 Dataset

The dataset used is termed with *dataset-thermal-stress* proposed by [10]. This dataset contains RGB images and thermal images of participants in a stress stimuli experiment and participants' stress level is labelled. A brief summary of the experiment would be that participants watch prepared pictures or videos that have stress implication. It is of research value as stress is a major and common problem worldwide. Furthermore, the data are collected by contact-free approach, which is of great practical promotion value. For the purpose of examining GA on FNN, which is not deep neural network structure, images are not used rather than their top principle components.

As a result of adopting principle components as features, it can be demonstrated *dataset-thermal-stress* becomes weakly-informative. In other words, these features consist of very little information that can be used to predict stress level. As the same principle highlighted in [5], the unsuitable nature of problem or dataset in the context of fitting neural networks leads to lack of generalization. This is one of the key reasons that we focus on convergence ability and speed during training rather than model generalization ability during testing.

1.3 Experiment and paper structure

One experiment on finding optimal hyperparameter tuning to maximize convergence ability and speed is carried out on *dataset-thermal-stress* by GA. For the key components of GA – selection, crossover, mutation etc., there are also different operators (algorithms) and tunable parameters. We shall experiment with a few and report their performance. We shall report the optimal setting found throughout experiment. [6] claims that exploration and exploitation are two cornerstones of search algorithms to solve problems. We consider the statistical properties of final population as an overview of trade-off between exploration and exploitation, so that's the second part of our report on the experiment.

In section *Method*, a detailed report is provided on profiling the dataset, justifying the nature of dataset being weakly-informative; a baseline of FNN is provided in addition to the optimal *casper* network found in previous paper; finally the instantiation of main components of GA in our problem is provided. In section *Result*, the optimal hyperparameter setting found by GA is firstly reported with a comparison to baseline and *casper*; then experiment results of different architectures and parameters of GA are reported, with discussion on results that leads to conclusion.

2 Method

2.1 System environment

All data reported below is by experimenting on one Ubuntu 19.10 desktop with Intel i7-7700 CPU set. The version of required library is: Anaconda Python 3.7, Pytorch 1.5, CUDA 10.1(though not training network on GPU).

2.2 Dataset profiling

The features of *dataset-thermal-stress* are PCA eigenvalues, ranging from -10^6 to 10^6. To prevent gradient exploding or vanishing in FNNs, normalization to -1 to 1 is performed as the first step. Then we explore the nature of *dataset-thermal-stress,* proving that it is weakly-informative, three pieces of evidence is provided.

First of all, the statistical Pearson correlation measure is conducted, and its coefficients are reported in Table 1. The largest absolute value of Pearson coefficient of label with respect to prediction target *label* is 0.022, which suggests all features by themselves only are weakly correlated with target.

	RGB_pca1	RGB_pca2	RGB_pca3	RGB_pca4	RGB_pca5
label	0.00536	0.00553	-0.00928	0.00394	0.022272
	Thermal_pca1	Thermal_pca2	Thermal_pca3	Thermal_pca4	Thermal_pca5
label	-0.00404	0.01494	0.01016	-0.00151	0.00255

Table 1. Pearson correlation of *dataset-thermal-stress*

Secondly, visualisation of distribution can be achieved by performing a dimension reduction. As shown in Figure 1, the two classes of target are highly overlapped in this parameter space. Thirdly, we heuristically determine the hyperparameters of FNN and do a complete training-validating-testing process. The average results are reported in Table 2 (an example of entry in *topology*: *i-15-10-o* means 2 hidden layers of 15 neurons and 10 neurons in both respectively). The training accuracy can reach 75% while testing accuracy is unstable and around 50%.

The instability and being around 50% of test accuracy do not change with GA, plus that we focus on behaviour in training phase, so the test accuracy will not be reported for the experiment.

	topology	activation	epoch	Learning rate	Train accuracy (%)	test accuracy (%)
1	i-25-o	sigmoid	5000	0.001	53.05	43.50
2	i-15-10-o	sigmoid	5000	0.001	51.06	45.75
3	i-15-10-o	ReLU	5000	0.001	72.48	49.03
4	i-15-10-o	ReLU	10000	0.001	74.05	42.22
5	i-15-10-o	ReLU	10000	0.0005	75.44	50.36

Table 2.

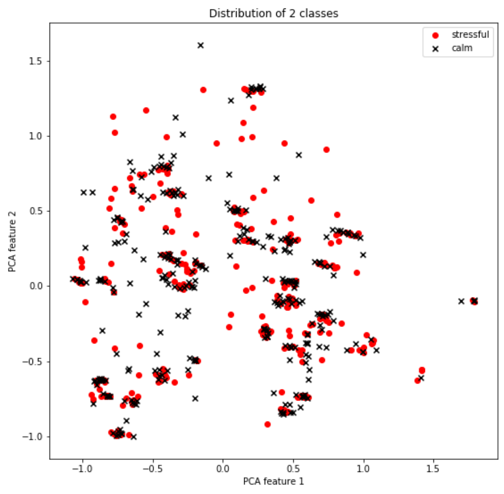

Fig. 1.

2.3 Baseline

We define the best hyperparameter setting achieved by manual tuning in data profiling phase as the baseline, the training accuracy is different as reported in 2.2 since there is no train-test split here. Table 3 demonstrates the result (see 2.4.1 for fitness), The baseline also indicates the search space of hyperparameters in GA:

	Number of epochs	Number of tunable weights	Training accuracy	Fitness value
Baseline	10000	347	0.7216	2.538

	Number of neurons in 1st hidden layer	Number of neurons in 2nd hidden layer	Number of epochs	Learning rate	Weight decay
Baseline	15	10	10000	0.0005	0.001

Table 3.

2.4 GA

2.4.1 Fitness measure: convergence speed and ability

The fitness function corresponds to an individual's ability to survive in the environment in Biology and represents the objective function in computational intelligence, which aligns with the optimization target [2]. The fitness measure in our experiment is a direct measurement of convergence speed and ability. Firstly, in the context of the training phase of we define the number of epochs in training as convergence speed; the number of trainable weights in FNN and training accuracy as convergence ability (fewer weights and higher accuracy means better convergence ability). Secondly, we quantify them in the following formula:

$$F(e,n,a) = (1-\frac{e-E_{min}}{E_{max}-E_{min}}) + (1-\frac{n-N_{min}}{N_{max}-N_{min}}) + 4 \times (a-0.5)$$

Fig. 2.

where *e, n, a* stand for the number of epochs, number of weights, training accuracy respectively. A higher *F* value means the individual is more likely to survive. The range of *F* value is theoretically 0 to 4 inclusive.

Number of epochs (*e*): E_{min} and E_{max} is the minimum and maximum possible value of number epochs, this is defined manually. For search the best number of epochs, a search space is defined which is parameterized by max and min value. Therefore, the fitness can be defined on a comparison of current value to these two.

Number of weights (*n*): N_{min} and N_{max} are defined similarly. So is the fitness criterion.

Training accuracy (*a*): Since it is a binary classification problem, a random classifier can statistically achieve 50% accuracy. So 0.5 becomes the baseline. The number 4 is a scaling coefficient to emphasize the accuracy more.

2.4.2 Chromosome

In GA, one chromosome is one candidate solution, and the genotype is the genetic composition of one candidate [2]. For our experiment, one chromosome is one FNN trained on the dataset, characterized by different hyperparameter settings. Therefore, a dictionary data structure is used to represent the genotype and then used to describe a candidate solution Table 4 shows the different genes and their value type. The clamp values in the third column are obtained manually by observing a few pre-experiment trails.

Gene	Value type	Clamp
Number of neurons in first hiddenly layer	Integer	[10, 30]
Number of neurons in first hiddenly layer	Integer	[5, 30]
Number of epochs	Integer	[7500, 20000]
Learning rate	Float	[5e-5, 5e-3]
Weight decay	Float	[5e-5, 5e-3]

Table 4.

2.4.3 Selection

The selection operator is directly associated with the concept of survival of the fittest in GA [2]. The concept of *selective pressure* is used to compare different selection operators. It is a measure of the degree of emphasizing on better solutions, and there is a trade-off between that and population diversity. We shall compare two selection operators.

2.4.4 Crossover

The crossover process is to generate one or more individuals by combining genes from randomly selected parents that pass the selection phase [2]. In our experiments, we use the *sexual* category of crossover where each offspring is produced by exactly two parents, since we want the gene combination (why not *asexual*) and the genotype size is not large (why not *multi-recombination*). All genes in our chromosome setting are of continuous value, so we shall experiment with two different floating-point-presentation crossover algorithms proposed in [2].

2.4.5 Mutation

The mutation process randomly changes the value for genes, which introduces new genetic material and therefore increase the population diversity [2]. From the optimisation perspective, mutation can be viewed as a random walk in search space. We use the famous uniform mutation for all genes and experiment with different mutation rates.

3 Results and discussion

Since the running time of GA is rather long and our time is limited, all results reported in this section are an average of 2 runs of GA. We examine different approaches and parameters in GA in order, so the optimal value obtained shall be used and verified in all later steps.

The first report is on the best hyperparameter setting found by the experiment in Table 5, which is defined by best convergence ability and speed in training (and numerically evaluated by the fitness function). The above results cannot be compared to [10], as they do not provide a report on convergence speed and ability, which is not their research focus.

	Best found by GA	Baseline	Best found by *casper*
Fitness value	3.1369	2.5380	2.9612
Train accuracy (%)	86.13	75.44	74.03

	Number of neurons in 1st hidden layer	Number of neurons in 2nd hidden layer	Number of epochs	Learning rate	Weight decay
Best found by GA	30	6	7500	0.00137	5e-5

Table 5.

3.1 Population size and number of generations

The efficiency and effectiveness of algorithms treated with GA are dependent on achieving a balance between population size and the number of generations [4]. We study the effectiveness of GA in different settings of population size and the number of generations while keeping the total number of searched solutions constant - 400. The reasons for setting it to 400 are that 400 already produces good solutions and GA takes hours to run while our time is quite limited. The optimal setting found is population size being 20 and generation being 20. Table 6 shows the details.

	POP_SIZE=5 N_GEN=80	POP_SIZE=10 N_GEN=40	POP_SIZE=20 N_GEN=20	POP_SIZE=40 N_GEN=10	POP_SIZE=80 N_GEN=5
Best fitness value in final generation	2.6766	2.7003	2.9243	2.9132	2.8771
Avg fitness value in final generation	2.545	2.5934	2.6538	2.3119	2.1052
Variance fitness value in final generation	0.0044	0.0094	0.1014	0.1038	0.2001
Best fitness value in all generations	2.7557	2.7946	3.0115	2.9502	3.0871

Table 6.

We see that a small population size results in the individuals in the final population being not so fitted to the environment even after a large number of generations, and this suggests an over-emphasis on exploitation. The result of the reverse setting is that though the best individual has high fitness value, the average of final population is quite low, indicating a large portion of underfitting individuals; and it clearly reveals search process of such setting is quite insufficient of exploitation. Balanced population size and number of generations is a good trade-off of exploration and exploitation, which obtains a stably good final population along with finding optimal fitness value. Figure 2 shows the training process of the optimal setting.

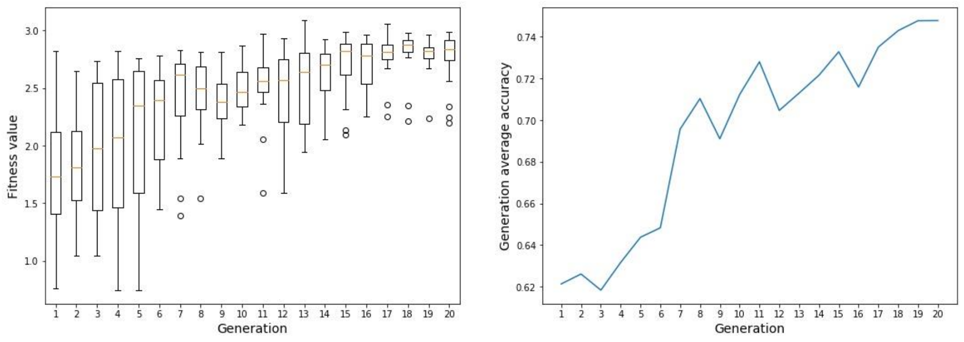

Fig. 3.

3.2 Mutation rate

GA as a search approach, its mutation process is a key control to exploitation. Intuitively, a higher mutation rate emphasizes more on exploitation. We experiment with a few different mutation rates and report their impact in Table 7. We see that although the best fitness value is found under 0.01, its overall fitness of the final population is slightly worth than that of 0.002. A general trend is that higher mutation rate leads to more emphasis on exploration over exploitation. A relatively high mutation rate explores more, but these explorations are undirected random search, so it is not robust to find a good solution. These conform to the effect of mutation rate.

Mutation rate	0.002	0.01	0.05	0.2
Best fitness value in final generation	2.9894	2.9243	3.0103	2.6631
Avg fitness value in final generation	2.7583	2.6538	2.4538	2.1315
Variance fitness value in final generation	0.0534	0.1014	0.1239	0.3313
Best fitness value in all generations	3.0115	3.1041	3.0103	2.9084

Table 7.

3.3 Selection algorithm and crossover algorithm

There are two candidate selection algorithms and two crossover algorithms. For selection, [7] proposes proportional selection and we propose its Laplace smoothing version. The formula is in Figure 4. For crossover, we study blend crossover and the simulated binary crossover (SBX) proposed by [9] and [11] respectively. The formula is in Figure 5.

$$\varphi(x_i(t)) = \frac{f(x_i(t))}{\sum_{l=1}^{n} f(x_l(t))}$$

$$\varphi(x_i(t)) = \frac{f(x_i(t)) + 1}{n + \sum_{l=1}^{n} f(x_l(t))}$$

Fig. 4. Top – proportional selection; Bottom – its Laplace smoothing

$$\tilde{x}_{ij}(t) = (1-\gamma_j)x_{1j}(t) + \gamma_j x_{2j}(t)$$
$$\gamma_j = (1+2\alpha)U(0,1) - \alpha.$$

$$\begin{aligned}\tilde{x}_{1j}(t) &= 0.5[(1+\gamma_j)x_{1j}(t) + (1-\gamma_j)x_{2j}(t)]\\ \tilde{x}_{2j}(t) &= 0.5[(1-\gamma_j)x_{1j}(t) + (1+\gamma_j)x_{2j}(t)]\end{aligned}$$

$$\gamma_j = \begin{cases} (2r_j)^{\frac{1}{\eta+1}} & \text{if } r_j \leq 0.5 \\ \left(\frac{1}{2(1-r_j)}\right)^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases}$$

Fig. 5. Left – blend crossover; Right – simulated binary crossover

We experiment with the 4 combinations and report the results in Table 8. Based on the results, we see that compared to the other one, smoothed proportional selection and SBX results in a less fitted final population, so we can conclude that they emphasize more on exploration.

Selection, crossover combination	Blend + proportional	Blend + smoothed proportional	SBX + proportional	SBX + smoothed proportional
Best fitness value in final generation	2.9243	3.0442	2.9891	2.7303
Avg fitness value in final generation	2.6538	2.4011	2.5436	1.7388
Variance fitness value in final generation	0.1014	0.2695	0.2548	0.2641
Best fitness value in all generations	3.1041	3.0834	3.1369	3.1173

Table 8.

4 Conclusion and future work

We applied genetic algorithm to find the optimal hyperparameter setting of an FNN trained on *dataset-thermal-stress*, results from a stress stimuli experiment. The optimal criterion is defined on the convergence speed and ability of a model. The reported optimum found by GA in our experiment has an increase of 24% of fitness value from baseline. During experimenting with different genetic algorithm architectures, by studying statistical properties of the final population, we conclude that exploration-exploitation trade-off is important for the performance of GA and different architectures have different implications. In future work, we can refine the fitness function and finds a more systematic criterion. Also, vector-based crossover algorithms such as parent centric crossover proposed by [14] can be examined as we only experimented with scalar-based crossover algorithms.

References

[1] Treadgold, N.K. and Gedeon, T.D., "Exploring constructive cascade networks," in IEEE Transactions on Neural Networks, vol. 10, no. 6, pp. 1335-1350, Nov. 1999.

[2] A. P. Engelbrecht, Computational Intelligence: An Introduction. (2nd ed.) Hoboken, NJ;Chichester, West Sussex, England;: J. Wiley, 2007.

[3] Treadgold, N.K. and Gedeon, T.D., "A Cascade network algorithm employing Progressive RPROP, " 10.1007/BFb0032532, 2006.

[4] K. Kalaiselvi and A. Kumar, "An empirical study on effect of variations in the population size and generations of genetic algorithms in cryptography," in 2017, . DOI: 10.1109/ICCTAC.2017.8249997.

[5] Lewis, I. J., & Beswick, S. L. (2015). Generalisation over details: The unsuitability of supervised backpropagation networks for tetris. *Advances in Artificial Neural Systems,* doi: http://dx.doi.org.virtual.anu.edu.au/10.1155/2015/157983

[6] M. Črepinšek, S. Liu and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," ACM Computing Surveys (CSUR), vol. 45, (3), pp. 1-33, 2013.

[7] J.H. Holland. Adaptation in Natural and Artificial Systems. University of Michigan, Press, Ann Arbor, 1975.

[8] Tin-Yau Kwok and Dit-Yan Yeung, "Constructive algorithms for structure learning in feedforward neural networks for regression problems," in IEEE Transactions on Neural Networks, vol. 8, no. 3, pp. 630-645, May 1997.

[9] L.J. Eshelman and J.D. Schaffer. Real-Coded Genetic Algorithms and Interval, Schemata. In D. Whitley, editor, Foundations of Genetic Algorithms, volume 2, pages 187–202, San Mateo, 1993. Morgan Kaufmann.

[10] Irani, R., Nasrollahi, K., Dhall, A., Moeslund, T. B. and Gedeon, T.D., "Thermal super-pixels for bimodal stress recognition," 2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA), Oulu, 2016, pp. 1-6.

[11] K. Deb and R.B. Agrawal. Simulated Binary Crossover for Continuous Space. Complex Systems, 9:115–148, 1995

[12] M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey," Computer, vol. 27, (6), pp. 17-26, 1994.

[13] J.H. Holland, Adaptation in Natural and Artificial Systems, Univ. of Michigan Press, Ann Arbor, Mich., 1975

[14] K. Deb, D. Joshi, and A. Anand. Real-Coded Evolutionary Algorithms with Parent-Centric Recombination. In Proceedings of the IEEE Congress on Evolutionary Computation, pages 61–66, 2002.