Optimizing of Hyperparameters of Machine Learning Models Using Genetic Algorithms

Anna Hovakimyan
Yerevan State University
Yerevan, Armenia
e-mail: ahovakimyan@ysu.am

Siranush Sargsyan
Yerevan State University
Yerevan, Armenia
e-mail: siranushs@ysu.am

***Abstract*—The paper studies the problem of optimizing hyperparameters of machine learning models, which have a significant impact on the quality of the model and its performance. To solve the problem, a genetic algorithm (GA) was developed and applied to optimize the hyperparameters of various models. The MLPClassifier and RandomForest Classifier models were trained and tested on the MNIST dataset, and the RandomForestRegressor model was trained and tested on the California Housing dataset. The optimization results were compared with the results of traditional Random Search and Grid Search methods, demonstrating the effectiveness of the genetic algorithm in terms of both computational resources and the quality of the resulting models.**

***Keywords*—Machine learning, hyperparameters optimization, genetic algorithm.**

I. Introduction

Currently, with the development of data science and machine learning (ML), the need to increase the efficiency of models has grown. The performance of machine learning models depends significantly not only on the quality of the data, but also on the correct choice of hyperparameters for the models. Manually tuning hyperparameters can be time-consuming, require deep expertise, and often do not provide the best possible results [1,2,3,6]. In this regard, the development of automated optimization methods is important.

Many solutions have been proposed to address the problem of hyperparameter optimization (HPO) in machine learning models.

A method, Hyperband, based on the Bandit algorithm, is developed that focuses on speeding up random search through adaptive resource allocation and early stopping. This method has been shown to provide speedups of more than an order of magnitude for various deep learning and kernel-based models [7].

The advantages of the bandit-based HPO method and evolutionary search approach are combined to achieve high performance for different ML models [8].

A method is proposed that outperforms both Bayesian optimization and bandit-based methods on a wide range of problems, such as support vector machines, feed-forward neural networks, Bayesian neural networks, deep reinforcement learning, and convolutional neural networks [9].

An evolutionary approach based on genetic algorithms was also used. The NSGA-II algorithm provides a much better distribution of solutions and better convergence near the true Pareto-optimal front [10].

Genetic algorithms (GAs), inspired by the principles of biological evolution, represent an effective method for solving optimization problems, in particular, for finding optimal values for hyperparameters in machine learning models. The main advantage of GA is that it can avoid stopping at local optima and, by efficiently exploring the search domain, produce sufficiently acceptable solutions [11-13].

The paper presents the development and application of GA to the problem of optimizing hyperparameters of machine learning models. A software system has been developed, experiments have been conducted, and the results have been analyzed to demonstrate the applicability of GA in such problems. The optimization results were compared with the results of traditional Random Search and Grid Search methods.

II. Hyperparameters of Machine Learning Models and Their Impact on Model Quality

Machine learning models typically have two types of parameters: trainable parameters, which are changing during the model's training on data, and hyperparameters, that are defined before training and directly affect the model structure, training process, and final quality of the model [1-4].

The correct choice of hyperparameters is of great importance. It can significantly change the learning speed of the model, eliminate possible overfitting or underfitting, and affect the generalization ability of the model. Examples of hyperparameters are the learning rate of the model, the structure of the neural network (number of hidden layers, number of neurons in each hidden layer), the number of trees in the decision forest, and the regularization coefficients [2-6].

Hyperparameter tuning is most often done manually, through experimentation, search methods, or the use of

optimization algorithms. These methods include Grid Search, which provides the consideration of all possible combinations of values defined in the search domain, Random Search, which supposes random selection of value combinations in the search domain, and Heuristic methods, which include GAs, particle swarm optimization (PSO), simulated annealing (SA), and other evolutionary methods [11-16]. Heuristic methods are mostly used in cases when the search domain is very large and optimal solutions are difficult to find with traditional methods due to nonlinear relationships or complex parameter interactions [17-19].

III. Application of Genetic Algorithms in Machine Learning Models

Genetic algorithms find application in various machine learning problems, where efficient and automated selection of model parameters or design of model structure is required.

We consider three machine learning models: MLPClassifier, RandomForestClassifier, and RandomForest Regressor, and a GA was used to determine the optimal values of their hyperparameters [1].

The chromosome encodes the vector of model hyperparameters. The fitness function is constructed based on the model quality assessment metrics [1,3-5,12]. Genetic operations are defined as follows [13,14].

Crossover. Each gene is chosen randomly from one of the parents.

Mutation. A random change occurs within an individual. By changing the value of a random gene, mutation ensures the introduction of novelty and avoids premature convergence to suboptimal solutions.

Tournament Selection. The best ones are selected from the population.

For the MLPClassifier model, we consider the following hyperparameters: hidden layers'_size of network, activation function, L2 regularization coefficient, initial learning rate, and training batch size.

The GA for the MLPClassifier model was run with the following parameters: population size - 16, number of generations - 10, mutation rate - 0.3, tournament size - 3, early stopping criterion - 5, if there is no improvement. Fitness was assessed based on the average value of the three-dimensional cross-validation accuracy [1, 11, 12].

The list of considered hyperparameters of the RandomForestClassifier model is the following: number of trees, maximum depth of trees, minimum number of samples for branching, minimum number of samples per leaf, and bootstrap.

For this model, the GA was run two times with the following parameters: population size - 10, 16, number of generations - 16, 16; mutation rate - 0.3, 0.3; tournament size - 3, 3; early stopping criterion - 5, 5, if there is no improvement. The fitness of the chromosomes was assessed based on the average accuracy value of three-dimensional cross-validation. The model was trained on 10,000 data points.

The list of the considered hyperparameters of the RandomForestRegressor model is the same as for the RandomForestClassifier model.

The GA for the RandomForestRegressor model was run with the following parameters: population size - 16, number of generations - 16, mutation rate - 0.3, tournament size - 3, early stopping index - 5, if there is no improvement. Chromosome fitness was assessed by three-dimensional cross-validation as the negative value of the Root Mean Squared Error [1, 4, 12].

IV. Experiments

A software for optimizing the hyperparameters of machine learning models was developed in the Python.

For the training and testing of the considered models the MNIST dataset of handwritten digits and the CaliforniaHousing dataset of house prices were selected to solve the classification and regression problems.

The main characteristics of the MNIST (Modified National Institute of Standards and Technology) dataset are the number of samples - 70,000 images (60,000 for training, 10,000 for testing), image size - 28x28 pixels (784 features), target classes - 10 (for digits 0 to 9), and image color description - grayscale.

The California Housing dataset contains housing price data for various regions of California. The main characteristics of the collection are the number of samples - 20640, the number of features - 8, and the target variable - the average house price.

While preprocessing, the data was scaled to ensure the acceleration and stabilization of the model training process. During GA experiments, reduced versions of the datasets were used to increase computational efficiency.

The MNIST dataset was used to optimize the hyperparameters of the MLPClassifier and RandomForestClassifier classification models, and the California Housing dataset was used for the RandomForestRegressor regression model.

The results of GA for the MLPClassifier model for two different initial populations are presented in Figs. 1, 2. The run time was 2547 and 2234 seconds, respectively.

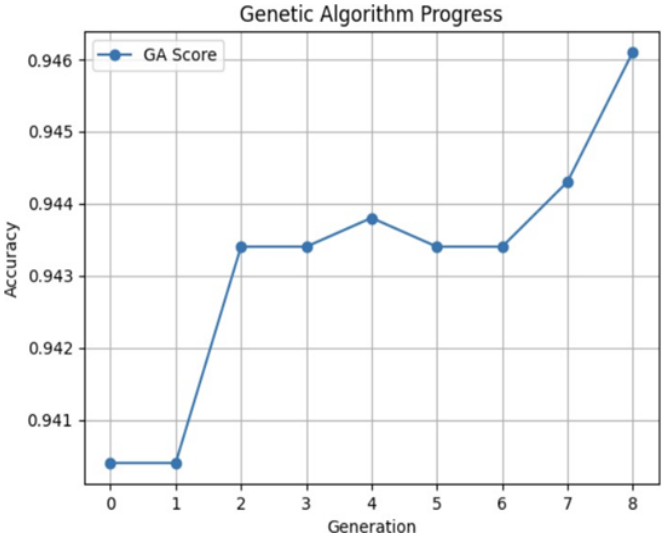

Figure 1. Generation number and accuracy of the model

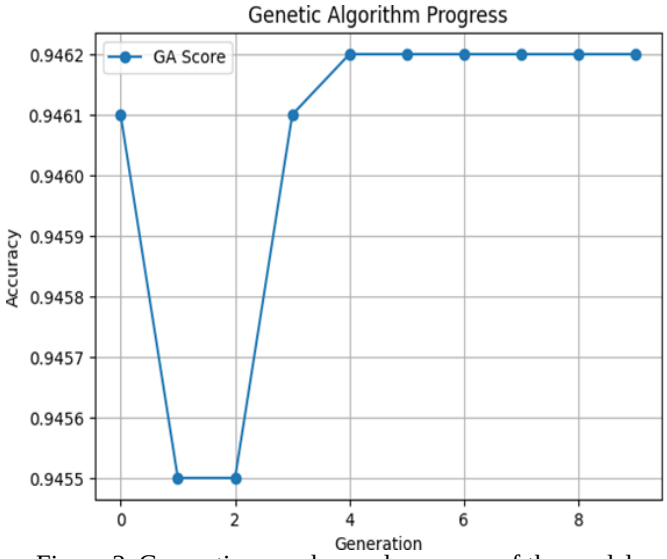

Figure 2. Generation number and accuracy of the model

The results of GA for the Random Forest Classifier model for different initial populations are presented in Figs. 3, 4. The run time was 1505 and 2234 seconds, respectively.

The results of GA for the Random Forest Regressor model for different initial populations are in Fig. 5. The run time was 303 and 309 seconds, respectively.

During the experiments, the results obtained by the GA were compared with the results obtained by the Grid Search and Random Search optimization algorithms [20]. Remember that the Grid Search method involves systematic testing of complete combinations of predefined parameter values. Although it is a simple and understandable method, it has a

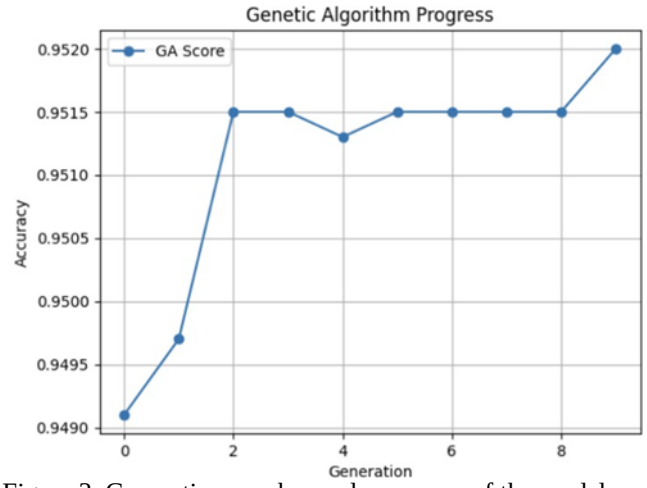

Figure 3. Generation number and accuracy of the model

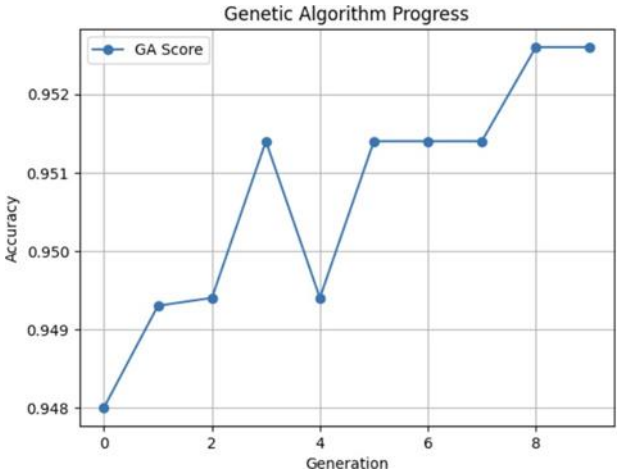

Figure 4. Generation number and accuracy of the model

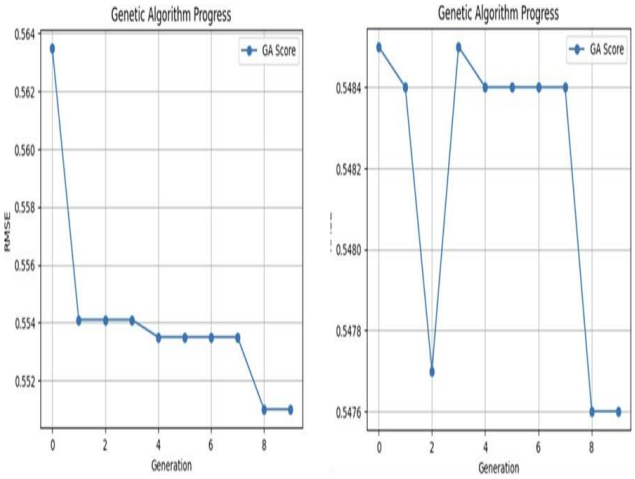

Figure 5. Generation number and RMSE metric of the model

number of limitations.

The Random Search method selects a predefined number of random combinations of parameters. This method is more efficient than Grid Search in some cases, but has its limitations [20].

Table 1 represents the results of a comparative analysis of the GA and Grid Search algorithms for the MLPClassifier model trained on the MNIST dataset, and Table 2 represents the results of a comparative analysis of the GA and Random Search algorithms for the Random Forest Classifier model trained on the same MNIST dataset.

Table 1

Model	Algorithm	Best Accuracy	Optimal values of hyperparameters	Run time	Accuracy on testing set
MLPClassifier	GA	0.9462	[(128, 128), relu, 0.01, 0.001, 64]	2547 sec.	0.9798
	Grid Search	0.9462	[(128, 128), relu, 0.01, 0.001, 64]	3303.5 sec.	0.9759

Table 2

Model	Algorithm	Best Accuracy	Optimal values of hyperparameters	Run time	Accuracy on testing set
RandomForest Classifier	GA 1	0.9516	[178 est, 69 depth, 3 split, 1 leaf, no bootstrap]	1505 sec.	0.9723
	GA 2	0.9526	[268 est, 54 depth, 2 split, 1 leaf, no bootstrap]	2234 sec.	0.9708
	Random search	0.9526	[132 est, 48 depth, 2 split, 1 leaf, no bootstrap]	3033.5 sec.	0.9658

The results show that the GA allowed us to achieve the same accuracy of the model in a significantly shorter period of time. On the other hand, the results of model testing show that the hyperparameter values obtained by the GA provide not only high classification accuracy, but also excellent generalization ability to new data.

Table 3 represents the results of a comparative analysis of the GA and Random Search algorithms for the RandomForest Regressor model trained on the California Housing dataset.

Table 3

Model	Algorithm	Mean square error (RMSE)	Best hyperparameter values	Run time	Accuracy on testing set
RandomForest Regressor	GA	0.5476	[117 est, 23 depth, 2 split, 1 leaf, 3 features]	1505 sec.	0.9723
	Random search	0.5500	[116 est, 35 depth, 5 split, 1 leaf, 3 features]	180 sec.	0.3258

These results show that the GA is effective for regression models as well, providing high accuracy of predictions.

V. Conclusion

The paper presented various methods for optimizing hyperparameters of machine learning models: Grid Search, Random Search, Genetic Algorithm. A comparative analysis of the results of the GA with the results of the considered methods was performed. The features that make genetic algorithms competitive and effective for complex problems were identified.

A GA with a general structure was presented, which was adapted for optimizing hyperparameters of various models and datasets. All the experiments show that the GA maintains the diversity of the search domain and avoids premature stopping at local optima, and gradually improve the quality of solutions within each generation.

Thus, GA are suitable for hyperparameter optimization problems in ML models, especially when the search domain is large and has complex dependencies. The results obtained in the work confirm that the GA can be a credible alternative to classical searching and optimization methods, as well as an effective tool to improve ML models and increase the overall prediction accuracy.

References

[1] Sibanjan Das and Umit Mert Cakmak, *Hands-On Automated Machine Learning: Optimize Machine Learning Models and Algorithms Using AutoML Techniques and Tools*, Packt Publishing, Birmingham, 2021.

[2] Kizito Nyuytiymbiy, (2020) Parameters and Hyperparameters in Machine Learning and Deep Learning, [Online], Available: https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac/

[3] Bischl et al., (2023) Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges, [Online], Available: https://doi.org/10.1002/widm.1484

[4] Ivan Belcic, (2024) What is hyperparameter tuning, [Online], Available: https://www.ibm.com/think/topics/hyperparameter tuning

[5] Karl et al., "Multi-Objective Hyperparameter Optimization in Machine Learning - An Overview", *ACM Transactions on Evolutionary Learning and Optimization*, vol. 3, issue 4, no. 16, pp. 1-50, 2022, https://doi.org/10.1145/36105

[6] Franceschi et al., (2024) Hyperparameter Optimization in Machine Learning, [Online], Available: https://arxiv.org/abs/2410.22854

[7] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar, "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization",*Journal of Machine Learning Research*, vol.18, pp.1-52, 2018.

[8] Noor Awad, Neeratyoy Mallik, and Frank Hutter, "DEHB: Evolutionary Hyperband for Scalable, Robust and Efficient Hyperparameter Optimization", Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21), Montreal, Canada, pp.2147-2153, 2021.

[9] Stefan Falkner, Aaron Klein, and Frank Hutter." Robust and Efficient Hyperparameter Optimization at Scale", Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, pp.20-30, 2018.

[10] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II", *IEEE Transactions On Evolutionary Computation*, vol. 6, no. 2, pp.183-188, 2002.

[11] Melanie Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, Massachusetts, 1996.

[12] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Boston, 1989.

[13] Randy L. Haupt and Sue Ellen Haupt, *Practical Genetic Algorithms*, Second Edition, Wiley, New York, 2004.

[14] Srishti Chaudhary, (2022) Applications of Genetic Algorithms in Machine Learning, [Online], Available: https://www.turing.com/kb/genetic-algorithm-applications-in-ml

[15] E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*, Second Edition, Springer, New York, 2015.

[16] Kenneth A. and De Jong, *Evolutionary Computation: A Unified Approach*, MIT Press, Cambridge, Massachusetts, 2006.

[17] Lee, S., Kim, J., Kang, H., Kang, D.-Y., and Park, J. (2021). Genetic Algorithm Based Deep Learning Neural Network Structure and Hyperparameter Optimization, *Applied Sciences*, 11, Article 2. https://doi.org/10.3390/app1102074.

[18] Shanthababu Pandian (2025), Essential Hyperparameter Tuning Techniques, [Online], Available: https://www.analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-on-hyperparameter-tuning-and-its-techniques/

[19] Sarah Lee (2025), Effective Strategies and Best Practices in Hyperparameter Tuning, [Online], Available: https://www.numberanalytics.com/blog/effective-strategies-best-practices-hyperparameter-tuning/

[20] Pedro Liashchynskyi and Pavlo Liashchynskyi, (2019) Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS, [Online], Available: https://arxiv.org/abs/1912.06059