Computational Biology and Chemistry 97 (2022) 107619

Contents lists available at ScienceDirect

Computational Biology and Chemistry

journal homepage: www.elsevier.com/locate/cbac

Tuning hyperparameters of machine learning algorithms and deep neural networks using metaheuristics: A bioinformatics study on biomedical and biological cases

Sajjad Nematzadeh [a,c,*], Farzad Kiani [b], Mahsa Torkamanian-Afshar [a], Nizamettin Aydin [c]

[a] Department of Computer Engineering, Faculty of Engineering and Architecture, Nisantasi University, Istanbul, Turkey
[b] Software Engineering Department, Faculty of Engineering and Natural Sciences, Istinye University, Istanbul, Turkey
[c] Department of Computer Engineering, Faculty of Electrical and Electronics, Yildiz Technical University, Istanbul, Turkey

ARTICLE INFO

Keywords:
Tuning
Hyperparameters
Machine learning
Deep learning
Metaheuristics
Bioinformatics

ABSTRACT

The performance of a model in machine learning problems highly depends on the dataset and training algorithms. Choosing the right training algorithm can change the tale of a model. While some algorithms have a great performance in some datasets, they may fall into trouble in other datasets. Moreover, by adjusting hyperparameters of an algorithm, which controls the training processes, the performance can be improved. This study contributes a method to tune hyperparameters of machine learning algorithms using Grey Wolf Optimization (GWO) and Genetic algorithm (GA) metaheuristics. Also, 11 different algorithms including Averaged Perceptron, FastTree, FastForest, Light Gradient Boost Machine (LGBM), Limited memory Broyden Fletcher Goldfarb Shanno algorithm Maximum Entropy (LbfgsMxEnt), Linear Support Vector Machine (LinearSVM), and a Deep Neural Network (DNN) including four architectures are employed on 11 datasets in different biological, biomedical, and nature categories such as molecular interactions, cancer, clinical diagnosis, behavior related predictions, RGB images of human skin, and X-rays images of Covid19 and cardiomegaly patients. Our results show that in all trials, the performance of the training phases is improved. Also, GWO demonstrates a better performance with a p-value of 2.6E-5. Moreover, in most experiment cases of this study, the metaheuristic methods demonstrate better performance and faster convergence than Exhaustive Grid Search (EGS). The proposed method just receives a dataset as an input and suggests the best-explored algorithm with related arguments. So, it is appropriate for datasets with unknown distribution, machine learning algorithms with complex behavior, or users who are not experts in analytical statistics and data science algorithms.

1. Introduction

1.1. Background

Machine learning is the process of building patterns from sample observations to generalize for whole instances of the population. Although phenomena have deterministic reasons of occurrence, extracting and measuring all related attributes in a wide range of scientific problems is too difficult and in some cases such as continuous domains are impossible. The sampling, which converts continuous domain into discrete, is the method that approximates a sample dataset to the real world. But this process would lead to missing some information. In essence, machine learning approaches use these kinds of observations in datasets that contain sampled information for each instance. While supervised approaches of machine learning work on known instances, unsupervised and reinforcement learning assist in grouping samples and estimating the best rewards. Moreover, some algorithms classify instances with discrete labels, and another type of algorithms that predicts a real number for instances is called regression. This research focuses on both classification and regression in supervised learning. Also in recent years, deep learning as a modern approach is widely used especially in computer vision.

Generally, in these algorithms, a hypothesis function tries to discriminate instances in classification and to fit a curve in regression. Besides, a cost function calculates how a predicted value by the hypothesis function for an instance is close to the real answer. Although

* Corresponding author at: Department of Computer Engineering, Faculty of Engineering and Architecture, Nisantasi University, Istanbul, Turkey.
E-mail address: sajjad.nematzadeh@nisantasi.edu.tr (S. Nematzadeh).

https://doi.org/10.1016/j.compbiolchem.2021.107619

Available online 24 December 2021

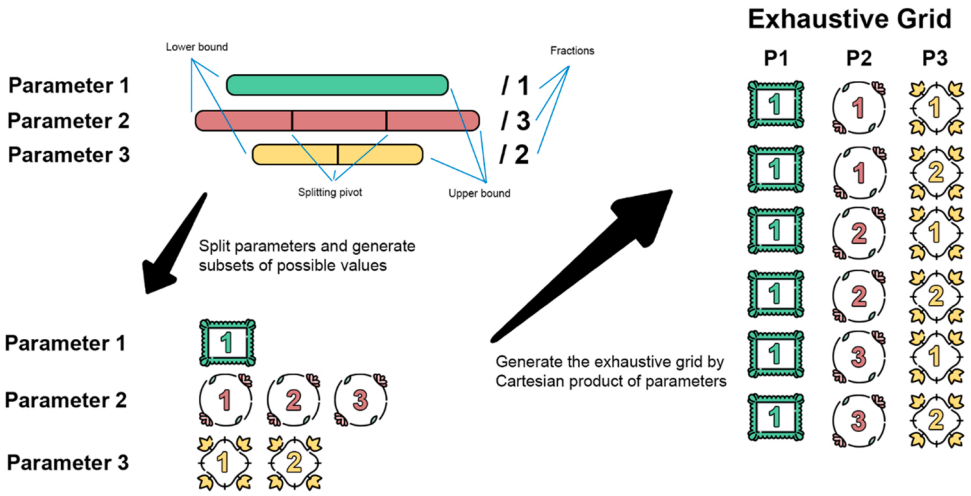


Fig. 1. An overview for generating exhaustive grid.

lower costs can express a better train, some other undesired issues like overfitting may remain hidden. To sort out these problems, a set of hyperparameters inside algorithms such as the learning rate and generalization rate is employed to control the behavior of the learning process. Each algorithm includes its hyperparameters with predefined initialization values. Generally, while many parameters of machine learning algorithms are derived through training, in recent years' modern machine learning algorithms have gained importance to tune hyperparameters to control the behavior of the relevant machine. Whereas other parameters (e.g., weights) are usually learned. The hyperparameters differ from parameters that are internal weights or coefficients for a model found by the learning algorithm. Thus, the performance of a machine learning algorithm is related to the proper setting of these parameters. To achieve this goal, machine learning practitioners can tune these hyper-parameters. In general, these parameters, although outside of the model, can be useful in machine performance by a direct and logical relationship between them.

Regarding the impact of the composition of various datasets on the performance of trained models, one of the most important issues in modern machine learning problems is the tunability of hyperparameters (Probst and Bischl, 2019). In other words, different initializing values for hyperparameters potentially can influence the machine learning metrics such as accuracy. Several studies propose methods to tune hyperparameters for Decision Tree (Alawad et al., 2018), Support Vector Machine (SVM) (Duarte and Wainer, 2017), Deep Neural Network (DNN) (Zhou et al., 2019), Random Forest (Probst et al., 2019), etc. These studies show that well tuning the hyperparameters of each algorithm increases the performance of the machine learning training process. Considering the different performances of learning algorithms on different datasets, the other crucial point in machine learning is to choose an appropriate algorithm for each dataset.

In this context, choosing a proper algorithm with a suitable configuration of hyperparameters can improve the performance of the model. The researchers who use machine learning and they are not expert in designing algorithms face two challenges. First is the manipulation of the algorithm so that it gets adapted to the tuning strategy. The second problem is to choose the algorithm itself. Moreover, similar to internal optimizers of algorithms that try to assign coefficients to minimize the cost, tuning the hyperparameters to obtain the best performance for a dataset with enormous permutations of values is a Non-deterministic Polynomial-time (NP-Hard) problem (Hutter et al., 2014; Yang and Shami, 2020; Bacanin et al., 2020; Talbi, 2020; Fuchs et al., 2019). In other words, trying blindly to find the appropriate tuple solution is quite an expensive procedure, and it becomes more expensive as the search dimension increases. It is worth mentioning that until now, there is no empirical evidence in the literature as to which hyper-parameters should be set and with what values this should be done. Metaheuristic algorithms have a good reputation to solve complex and NP-Hard problems and they are used widely in different fields (Hussain et al., 2019). There is a search area according to the nature of the problem and the most appropriate solution is tried to be searched in this area. If the whole area is searched with precise methods, a very complex and costly process is entered, while metaheuristic algorithms try to find a solution close to the best solution with less cost. Therefore, these algorithms can be important candidates for solving such problems. The evolutionary, swarm, physics, nature-inspired are examples of types of metaheuristics (Khanduja and Bhushan, 2021; Halim et al., 2021).

In recent years, by growing generated data in biology and biomedical, the process costs are increasing and the solution of the mentioned problem becomes even more difficult and critical. So, the getting benefits of optimized solutions and algorithms are needed. A portion of these solutions is covered by machine learning. Regarding the crucial role of precise predictions in life-related sciences, the reliability of the machine learning models should be emphasized. Although simulation approaches like molecular dynamics typically have better accuracy, the high order of their time complexity is a big disadvantage, especially in high throughput processes. So, predictions using accurate machine learning models are appropriate and faster alternatives. For example, in (Torkamanian-Afshar et al., 2021), an algorithm tries to generate a bindable RNA as an aptamer for a specific protein such as CD13 biomarker. Enormous candidates of RNA sequences are generated in each run and quickly predicted by a model.

Also, in (Lanjanian et al., 2021), the analysis of interactions between viral proteins and host cell RNAs are possible with quick and accurate predictive models. Despite the machine learning has a supreme position in bioinformatics, computational biology, and biomedical, there are not adequate researches about improving the performance of machine learning utilizing the optimization of hyperparameters. Not only the improvement of the performance leads to having a more accurate model, but also it can justify the causality of features in their applications. Therefore, in addition to proposing a fully operative tuning method, biomedical and biological datasets are employed in this study. In this way, it is ensured that the work is audited in various situations such as high-dimensional, sparse, imbalanced, and small sample-sized datasets.

In this study, 11 different datasets with various applications are chosen to be inspected. Moreover, six classifiers including Light Gradient Boost Machine (LGBM) (Ke et al., 2020), the Limited memory Broyden Fletcher Goldfarb Shanno algorithm (Lbfgs) (Wah June and Abu Hassan, 2005), FastTree (Price et al., 2009), FastForest (Yates and Islam, 2021), Averaged Perceptron (Mcdonald et al., 2010), and Linear

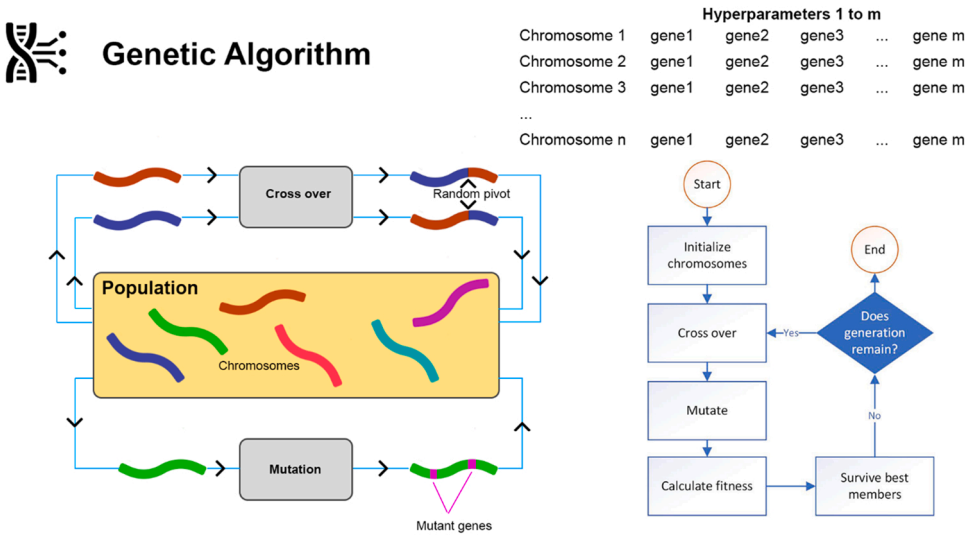


Fig. 2. The general working mechanism of GA.

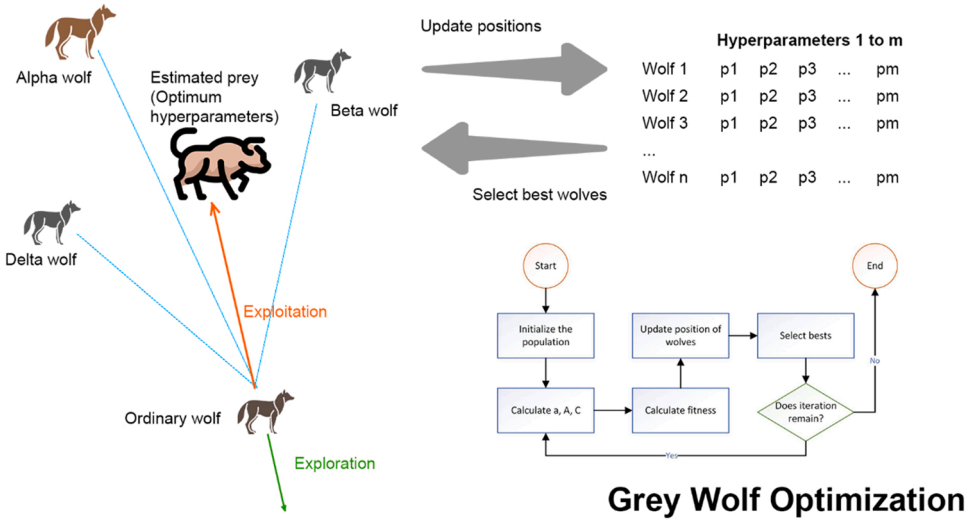


Fig. 3. Working mechanism of the GWO.

Support Vector Machine (LinearSVM) (Chamasemani and Singh, 2011) are employed. Furthermore, the regression versions of the first four classifiers are used in this study. Besides, one DNN algorithm utilizing four different architectures classify the images of common pigmented skin lesions and Covid19 X-rays.

1.2. Exhaustive grid search

EGS is a brute-force method that considers all possible combinations to find the optimal answer. In hyperparameters tuning problems, the EGS is one of the most used approaches. A set of several values for each parameter forms the domain vector of that parameter. The cartesian product of these vectors generates an exhaustive grid. Eq. (1) denotes the number of all possible states obtained from the cardinality (number of different values) of each feature.

$$number \quad of \quad all \quad states \quad = \quad \prod_{i=1}^{n} C_i \tag{1}$$

Where the C represents the number of splitting points and n denotes the number of features. Any row of the exhaustive grid is a hyperparameter input for a training algorithm. Finally, the row with the best performance of training is selected. The EGS is a convenient-to-implementation and easy-to-understand method. However, depending on the number of hyperparameters, the composition of datasets, and the mechanism of the training algorithms, it might perform inefficiently. The discrete nature of this method can lead to missing some intermediate values, especially in continuous values like the learning rate. In addition, the large scale of the grid in training algorithms with many hyperparameters is the disadvantage of this method. Fig. 1 illustrates the overview of generating the exhaustive grid. Finally, a searching mechanism scans the results of all members in the Exhaustive Grid and selects the best one.

1.3. Metaheuristic algorithms

This study employs two types of optimizers to determine the most appropriate hyperparameters. The first is the Genetic Algorithm (GA) (Whitley, 1994), which takes place among the most famous evolutionary algorithms. As the second optimizer, the Grey Wolf Optimization (GWO) (Mirjalili et al., 2014) is a swarm-based optimizer that gives reasonable performance for finding both local and global optima (Dokeroglu et al., 2019). In general, metaheuristic algorithms are proper alternatives for

exhaustive solutions for problems with enormous possible combinations. Moreover, in problems that proposing analytic deterministic solutions are complicated, metaheuristics work well.

GA: The GA has been inspired by the biological evolution of life. Sharing genes between survived parents of previous generations is the primary idea of this algorithm. Also, the main loop up to the maximum number of iterations controls the flow of the algorithm. In each iteration, two cross-over and mutation functions generate new members utilizing previous members of the population. Then, the best members with the highest scores survive for the next generation. This operation repeats until performing the last generation. The GA is a suitable method for finding solutions, especially for the discrete domain. Nevertheless, this algorithm suffers from the convergence trap of all members in a population and finding optimum solutions for continuous domains. Fig. 2 gives the general mechanism of GA.

GWO: The GWO is a swarm particle-based algorithm that prefers to update the position of its search agents (grey wolves) by moving them toward an estimated prey (optimum solution). Each update which is called exploitation can lead to converging the search agent to a local optimum. On the other hand, a randomly inspired threshold can navigate a search agent far away from the prey. This action is called exploration. Accordingly, the GWO has a better performance to escape from a local optimum trap. Fig. 3 shows the information about the GWO.

1.4. Machine learning algorithms

Finding repeatable patterns from input data is the expectation from machine learning. Some types of methods inspect the known (labelled) data to perform supervised learning. On the other hand, another approach called unsupervised learning attempts to discriminate data and explore patterns from unlabelled data. Also, reinforcement learning gives rewards and penalties for the actions of agents to build proper solution paths. While regression methods estimate real numbers for inputs, classifiers predict the class of each input vector. The algorithms used in this study are as follows:

Averaged Perceptron: This algorithm is a simple version of the neural network. The linear function inside this classifier separates inputs and then combines them based on the weights of feature vectors. As the linearity of this algorithm provides fast learning, it is suitable for continuous training (Mcdonald et al., 2010; Goldberg and Elhadad, 2021).

FastForest: The FastForest algorithm is the improved version of Random Forest. This ensemble algorithm performs an average of 24% improvement in the duration of training in empirical tests. Therefore, it is a suitable training algorithm for devices with the lower hardware equipment. The algorithm follows these five steps: Initialize the tree loop, initialize tree, create a sub-bagged dataset, build a tree from the sub-bagged dataset, and add the tree to the forest (Yates and Islam, 2021).

FastTree: In (Price et al., 2009), the researchers have proposed an algorithm to speed up the decision tree for building large phylogenies and evaluating their reliability. This algorithm stores sequence profiles of internal nodes in the tree and ignores the distance matrix. Also, the algorithm uses nearest-neighbor interchanges to reduce the length of the tree. Accordingly, FastTree is a memory-friendly algorithm that suits large datasets on devices with limited memory.

LBFGS: Limited memory Broyden Fletcher Goldfarb Shanno algorithm (Lbfgs) (Liu and Nocedal, 1989) was proposed based on a limited-memory quasi-Newton method for large-scale optimization. Reducing the iteration cost is the primary advantage of this algorithm. Also, the algorithm performs global convergence on uniformly convex problems.

LightGBM: The Light Gradient Boosting Machine (LightGBM) algorithm (Ke et al., 2020) has been proposed based on Gradient Boosting Decision Tree (GBDT) to improve the efficiency and scalability for high-dimension and large-sized datasets. Instead of the time-consuming scanning of all data for estimating the information gain of all possible split points, The LightGBM pursues faster methods. It uses two Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) techniques to overcome this problem. GOSS estimates the accurate gain information with a smaller data size, and EFB helps to reduce the number of features. The authors claim that the LightGBM is 20 times faster than GBDT with achieving the same accuracy.

LinearSVM: The Linear Support Vector Machine (Chamasemani and Singh, 2011; Abe, 2010a, 2010b, 2010c) is one of the most famous training algorithms. This algorithm discriminates the regions of the space by decision hyperplanes. The algorithm tries to find the best hyperplane with optimum margins between classes. Moreover, instead of the single-kernel approach, multi-kernel approaches like polynomial, gaussian, and sigmoid can be used. Although this method is a binary classifier, some techniques like One Versus All (OVA) can accomplish multiclassification.

DNN: The Deep Neural Network is a learning technique that is inspired by the human brain. DNN consists of multiple nodes that form several sub-categories like Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), etc. These nodes connect to each other and create a graph to perform tasks like image and voice recognition. Many studies have tried to reduce the cost of learning in DNNs. For example, the idea of transfer learning is that portions of pre-trained materials can be reused in secondary modelling tasks. Reducing cost can be corresponded with increasing the accuracy of the model. Because resources like hardware and time can be utilized more efficiently (Canziani et al., 2021; Pan and Yang, 2010).

1.5. Related works

In the literature, there are researches about optimizing hyperparameters using statistical-based algorithms and metaheuristics. However, a wide range of these studies has focused on specific algorithms or applications. In (Lentzas et al., 2019), authors compare results of two datasets using GA and Quantum Genetic Algorithm (QGA). This study focuses on image classification. Although the quantum genetic algorithm supports more states than the genetic algorithm, the performance of the method for regression and numerical datasets is unclear. Due to internal replicating for each chromosome, the quantum genetic approach needs more resources. Adan et al. in (Godínez-Bautista et al., 2018) have been studied the tuning of SVM using four bio-inspired metaheuristics including Bat Algorithm, Firefly Algorithm, Particle Swarm Optimization Algorithm, and Social-Emotional Optimization Algorithm. The proposed method is proper for SVM. However, our study shows that some other algorithms appear with a better performance. As another study, the authors in (Passos et al., 2018) propose a method to fine-tune the Deep Boltzmann Machines through metaheuristics. They proposed a model to reconstruct binary images. In this model, it has been tried to optimize the number of hidden units, the learning rate, the penalty hyperparameter, and the weight decay. Gustavo et al. in (Lujan-Moreno et al., 2018) report their experiments about tuning the hyperparameters of a Random Forest case study. They proposed a factorial designs-based methodology to screen potential hyperparameters to get a response to evaluate the machine`s performance. In addition, they have used Response Surface Methodology (RSM) to optimize this response variable by fitting a second-order polynomial function. The authors have claimed that the proposed method is helpful for both optimizing the machine learning performance metric and which factors have the greatest impact on the response variable.

In (Neary, 2018), authors claim that asynchronous reinforcement learning can converge an optimal solution for image classification using Convolutional Neural Network (CNN) for the MNIST dataset. In (Lee et al., 2018), the authors have suggested a method for hyperparameter tuning in CNN's feature extraction phase by using a parameter-setting-free harmony search (PSF-HS) algorithm. They simulated the proposed method on LeNet-5 and an MNIST dataset, and

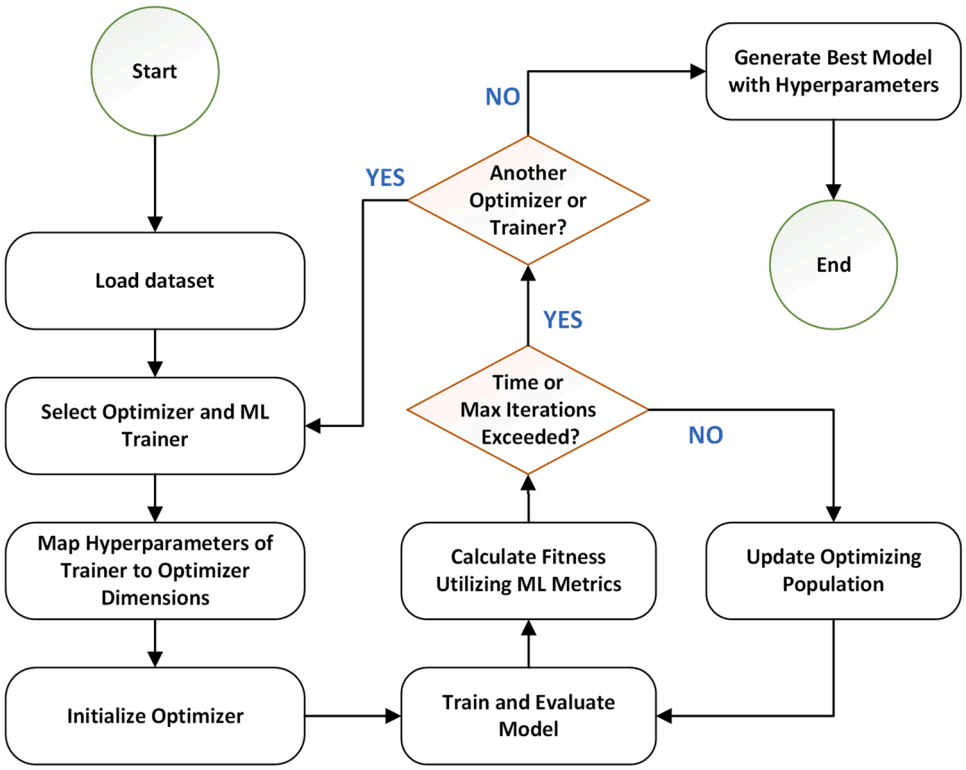


Fig. 4. The workflow of the proposed method.

also on CifarNet and a Cifar-10 dataset. According to their results, they claimed that they have improved the classic CNN. In another similar work (Bacanin et al., 2020) on computer vision using CNN, authors use an enhanced version of a swarm optimizer to tune the hyperparameters of CNN. They have tried to optimize hyperparameters using enhanced versions of the tree growth and firefly algorithms. In (Kabir Anaraki et al., 2019), a method that combines GA and CNNs was proposed for the non-invasive classification of glioma using Magnetic Resonance Imaging (MRI). In the structure of this method, the typical trial and error-based selection of neural networks for a particular problem were avoided. In (Darwish et al., 2020), the authors have developed an Orthogonal Learning Particle Swarm Optimization (OLPSO) algorithm to classify healthy and unhealthy leaf images in plant disease diagnosis by optimizing hyperparameter values for VGG16 and VGG19 CNNs.

Jia et al. in (Wu et al., 2019) suggest a method to optimize the hyperparameters based on Bayesian optimization. Their method sets a prior over the optimization function and collects related information about previous samples to update the posterior of the optimization function. They have used the random forest algorithm, multi-grained cascade forest, and neural networks. Also, some works study the application-oriented hyperparameter tuning (Tsai and Fang, 2021; Zhang et al., 2020; Zhou et al., 2021). Considering the variety of performances for different algorithms for different datasets, lack of a detailed inspection of various types of datasets and/or algorithms is felt in most studies in this field. Thus, choosing the best algorithm for machine learning users can be difficult. In short, the limited number and variety of algorithms and data sets used may be the common disadvantages of the proposed methods.

1.6. Contribution

In addition to extracted features, the number of samples, and the training algorithm, values of hyperparameters have a crucial role in the performance of machine learning models. The composition of hyperparameter values depends on datasets and training algorithms. Therefore, it varies from one problem to another problem. A considerable part of hyperparameters like "learning rate" is continuous. Accordingly, tuning these hyperparameters for a specific problem is an NP-hard problem. In conclusion, metaheuristic algorithms are suitable candidates to solve these kinds of predicaments. The contributions of the proposed method can be summarized as follows:

1) The proposed method can improve the performance of the machine learning model such as accuracy independent from the composition of datasets. Potentially, the method increases the probability of better predictions for future and unseen samples by relieving the entropy at the training phase.
2) It assists to gain a better performance faster than tuning by blindly chosen hyperparameters (in all tries, performances get better than initialization populations which are generated randomly).
3) The idea of this method is not specific to a limited number of training algorithms. In other words, it is applicable to a vast range of training algorithms utilizing various optimizers.
4) Both single-objective and multi-objective approaches can be applied.
5) This study suggests a clear method to be implemented in the automation of the real world.
6) An implemented version of this method will be convenient and easy to use for users who are not experts in algorithm designing.
7) The study shows that the metaheuristic methods give better result and converge faster than blind approaches like Exhaustive Grid Search.

The remainder of the article is organized as follows. Section 2 describes the proposed method in detail. Section 3 shows the results of our experiments using the proposed method and datasets. Section 4 describes the analysis and discussion about the results achieved. The last section is about conclusions and future works.

Table 1
Benchmark datasets.

ID	Dataset Name	# of Samples	# of Features	Cardinality of target	Description
1	RPINBASE (Torkamanian-Afshar et al., 2020)	18190	3075	2	RNA Protein interaction from complexes
2	Mutant P53 protein (Danziger et al., 2009)	16772	5409	2	Transcriptional activity prediction of mutant p53 proteins using in vivo determination.
3	Cervical Cancer Behavior (Fernandes et al., 2017)	72	19	2	Behavior determinant based cervical cancer early detection
4	QSAR Androgen Receptor (Grisoni et al., 2019)	1687	1024	2	Quantitative structure–activity relationship binding to androgen receptor
5	Heart disease (Detrano et al., 1989)	303	13	2	Diagnosis of coronary artery disease
6	QSAR Oral Toxicity (Ballabio et al., 2019)	8992	1024	2	Quantitative structure–activity relationship models to predict acute oral systemic toxicity
7	Parkinson (Naranjo et al., 2016)	240	46	2	Parkinson Dataset with replicated acoustic features
8	Obesity levels (Palechor and Manotas, 2019)	2111	17	7	Estimation of obesity levels based on eating habits and physical condition
9	Heart Failure (Chicco and Jurman, 2020)	299	13	2	Heart failure clinical records dataset
10	Covid19 X-rays (Minaee et al., 2020)	4600	Image	3	Lung X-rays scan of covid19, cardiomegaly, and healthy
11	HAM10000 (Tschandl, 2018)	10015	Image	4	Multi-source dermatoscopic images of common pigmented skin lesions

Table 2
General information of training algorithms with arguments and hyperparameters.

Training algorithm	Argument			
	Name	Bounds[a]	Name	Bounds[a]
LGBM	Number Of Leaves	[5,150]:100	Learning Rate	[1E-4,1]: 0.01
(multi-class classifier & regression)	Minimum Example Count Per Leaf	[5,150]:100	Number Of Iterations	[5,200]:100
Lbfgs Maximum entropy (LbfgsMxEn)	Dense Optimizer	[-1,1]: false	L1 Regularization	[1E-3,1]:0.1
(Multiclass classifier)	Enforce NonNegativity	[-1,1]: false	L2 Regularization	[1E-3,1]:0.1
Lbfgs Poisson regression (LbfgsP)	Optimization Tolerance	[1E-15,0.01]: 1E-7	History Size	[2,50]:20
(regression)	Maximum Number Of Iterations	[10,200]:100		
Linear SVM (multiclass classifier)	Batch Size	[1,5]:1	Lambda	[1E-4,1]:0.001
	Initial Weights Diameter	[0,1]:0	Number Of Iterations	[1,4]:1
Averaged Perceptron (multiclass	L2 Regularization	[0,4.9999]:0	Learning Rate	[1E-4,1]:1
classifier)	Initial Weights Diameter	[0,1]:0	Number Of Iterations	[4,20]:10
FastTree	Bagging Example Fraction	[0,1]:0.7	Learning Rate	[0,1]:0.1
(multi-class classifier & regression)	Dropout Rate	[1E-09F,1]:0.2	Number Of Leaves	[2,128]:64
	Entropy Coefficient	[0,1]:0	Gain Confidence Level	[1,0):0.95
	Feature Fraction	[0,1]:0.9	Shrinkage	[1E-5,10]:1
	Feature Fraction Per Split	[0,1]:0.9	Smoothing	[0,1]:0.5
	Minimum Example Fraction For Categorical Split	[0,1]:0.1	Minimum Examples For Categorical Split	[10,300]:100
	Softmax Temperature	[0,1]:0.5		
Fast Forest	Bagging Example Fraction	[0,1]:0.7	Number Of Leaves	[5,50]:20
(multi-class classifier & regression)	Entropy Coefficient	[0,1]:0	Smoothing	[0,1]:0
	Feature Fraction	[0,1]:0.7	Softmax Temperature	[0,1]:0
	Feature Fraction Per Split	[0,1]:0.7	Number Of Trees	[20,300]:100
	Minimum Example Fraction For Categorical Split	[1E-5,1]:0.01	Minimum Examples For Categorical Split	[10,300]:100
	Gain Confidence Level	[0,1):0.95		
DNN	Batch Size	[5,25]: 10	Epoch	[10,50]:40
(image classifier)	Architecture	ResnetV2101 ResnetV250 InceptionV3 MobilenetV2	Learning Rate	[1E-5,1]:0.01

[a] The format of bound is: [lower bound, upper bound]: default value

2. Method and materials

The primary goal of this study is to improve the performance of machine learning models via tuning hyperparameters using meta-heuristics. This improvement can be performed after other techniques such as data cleaning, scaling, and regularization. A little improvement in the performance of training algorithms would have a great effect on applications. As the attributes and features of observations in datasets are capable of having vast variances in various cases, configurations of any training algorithm can be affected by these contents. So, a successful configuration of a training algorithm for a dataset can reduce the performance of another dataset. Therefore, machine learning users need effective methods to tune their training algorithms for each case. The idea of this research is the tuning of training algorithms by optimizer methods. The method starts with loading the target dataset. Users can prefer an optimizer among GWO and GA to tune hyperparameters. Corresponding to the composition of training algorithm arguments, a function maps the arguments (hyperparameters) of each training algorithm to the dimensions (positions or chromosomes) of the optimizer. The best member of the population contains the optimal hyperparameter values. Also, the most suitable training algorithm can be chosen by comparing performances. As a short conclusion, the method improved the performance of models in 100% of our experiments.

For evaluating the efficiency of the proposed method, 11 machine learning training algorithms are tuned by two optimizers. Also, 11 datasets in various scales are used as input datasets. The workflow of the

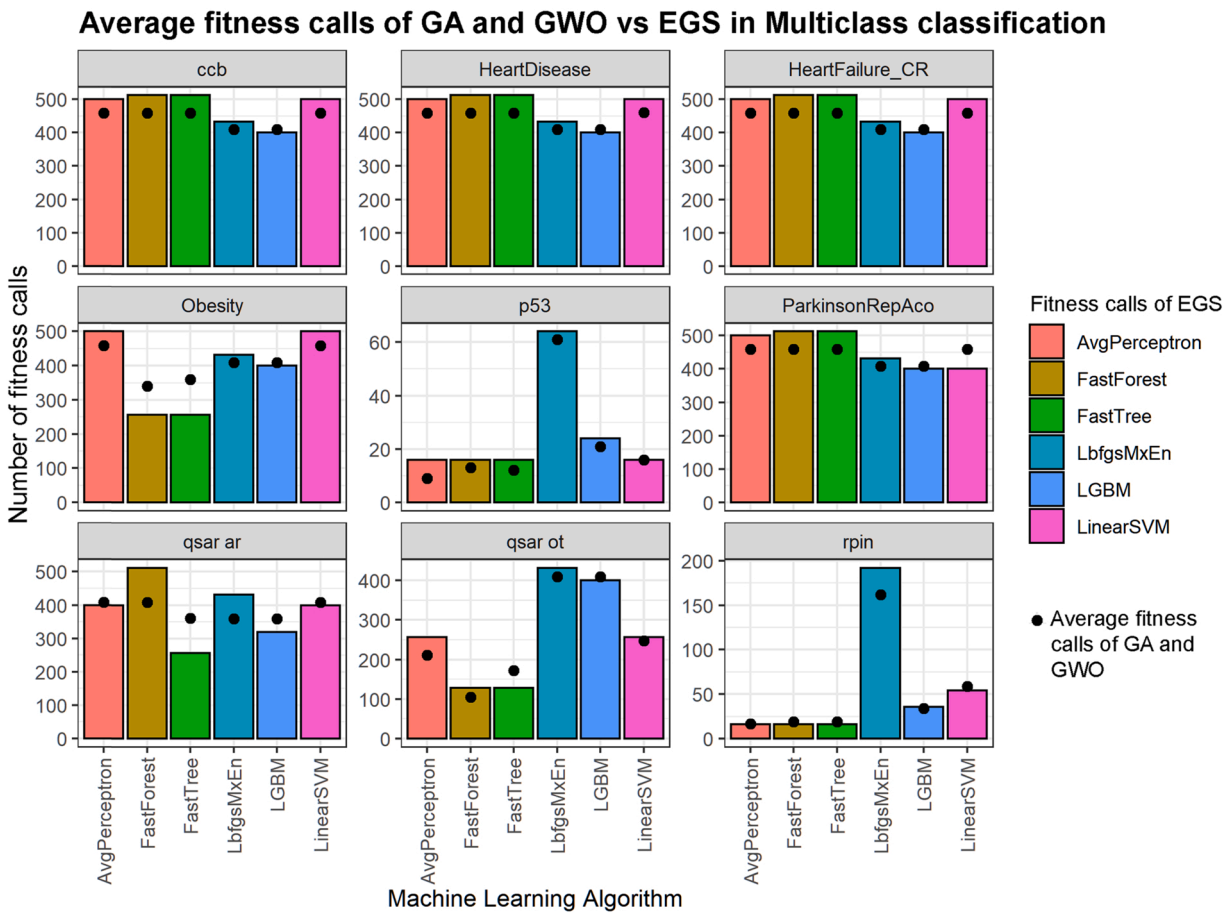


Fig. 5. Number of fitness calls in metaheuristics and EGS for multi-class classification.

proposed method is illustrated in Fig. 4. As mentioned before, in this study the GWO and GA are used as optimizers. Moreover, the LGBM, Lbfgs, FastTree, FastForest, and the regression versions of them are employed as ML training algorithms. Furthermore, Averaged Perceptron, LinearSVM, and one DNN algorithm are used as other ML training algorithms.

2.1. Benchmark datasets

Considering the scale of a dataset, machine learning training algorithms need different durations and hyperparameter configurations. The list of 11 datasets is shown in Table 1. Moreover, all these datasets are chosen among biological, biomedical, and nature-related categories such as molecular interactions, cancer, clinical diagnosis, behavior related predictions, RGB images of human skin, and X-rays images of Covid19 and cardiomegaly patients. The aim of choosing these datasets is to check the performance of hyperparameter tuning. All these datasets, which contain various numbers of samples and features, have a label target feature that is proper for classification and regression.

2.2. Machine Learning training algorithms

Three main approaches to supervised machine learning have been employed in this research. The first type of training algorithms contains six multiclass classifiers, the next group consists of four regression algorithms, and finally, a Deep Neural Network (DNN) with four architectures classifies images. While both groups of multiclass classifiers and regression algorithms work on precalculated and feature extracted datasets, DNN deals with raw images. The regression algorithm of the Light Gradient Boost Machine (LGBM), the Limited memory Broyden Fletcher Goldfarb Shanno algorithm (Lbfgs), FastTree, and FastForest were employed in this study. Besides, multi-class classification versions of these algorithms, the Averaged Perceptron, and the Linear SVM were used as classifiers.

General information of 11 training algorithms including hyperparameters is presented in Table 2. These training algorithms are directly used in the fitness functions of optimizers. Besides a dataset, each training algorithm receives some arguments as hyperparameters to configure its tasks. By default, a set of initializing values have been suggested by algorithm designers. The proposed method of this study tries to alter and find the best composition of these arguments for each training algorithm and dataset. The ML.Net framework (Ahmed et al., 2019) of Microsoft was employed to implement this method.

Deep Neural Network training algorithms: A DNN training operation using InceptionV3, ResnetV2101, ResnetV250, and MobilenetV2 architectures was performed to recognize classes of images. Similar to machine learning training algorithms, a mapper function converts values of arguments, which are shown in Table 2, to use in training algorithms. As proposing a general method is the primary aim of this study, instead of redesigning sub-layers of DNN in-depth, we focused on tuning general parameters. Shuffling and splitting the dataset with a 0.2 test fraction was used to evaluate the training and validation performances. As both increasing accuracy and decreasing cross-entropy are desirable, the early stopping criteria have been disabled to give the opportunity of gaining a non-greedy and minimized cost for the whole 80 epochs in training and validation. Regardless of any manipulation such as augmentation, cropping, denoising, and pre-processes, all images have been used in raw and untouched status.

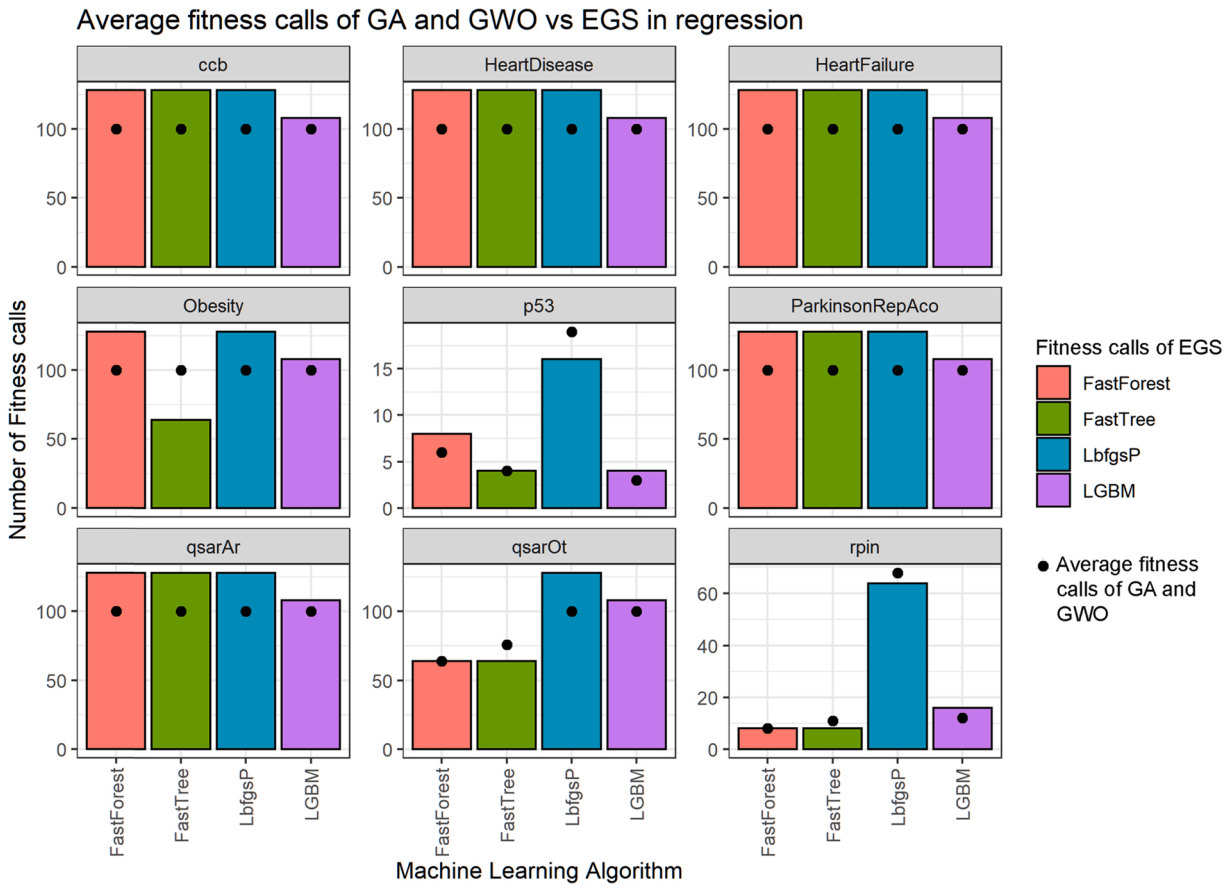


Fig. 6. Number of fitness calls in metaheuristics and EGS for regression.

2.3. Tuning Hyperparameters utilizing metaheuristics

Generally, hyperparameters of training algorithms can be initialized by default values. However, changing these values potentially changes the performance of training. As mentioned before, trying blindly to find the appropriate tuple solution is quite an expensive procedure, and it becomes more expensive as the search space expands (increasing the number of hyperparameters). This process is an NP-hard problem, and the metaheuristic algorithms can be used for solving this type of problem by avoiding unnecessary investigations in irrelevant regions. To find the optima, these hyperparameters are passed into metaheuristics. In the proposed method, the metaheuristic adjusts these hyperparameters to obtain the best-explored combination. In this study, two optimizers are chosen within evolutionary and swarm algorithms. The classic GA represents the evolutionary algorithms and the GWO is selected as a swarm optimizer. Members of the population in optimizers hold the hyperparameters of machine learning and DNN training algorithms as genes for GA and positions for GWO. Indeed, the input arguments of each training algorithm correspond to the dimensions of optimizers. Besides, mapper functions convert this decimal information to consumable arguments for training algorithms. For example, if a training algorithm requires an integer parameter, a decimal optimized value originated from an optimizer is rounded and in Boolean data type, the negative values are considered as false and, positives are turned into true.

The bottleneck of this experiment is the calculations inside the Fitness function because the fitness function performs a complete training and evaluation. So, the minimal configurations for optimizers were chosen to acquire a reasonable number of iterations. The population size, crossover rate, and mutation rate were set to 12, 0.25, and 0.09, respectively for GA, and the number of wolves in the GWO algorithm was initialized by 4. Moreover, for both optimizers in machine learning, max iterations were set to 100. In the case of DNN, the max iteration number was reduced to 50 iterations. To hold a fair competition between GA and GWO, in each iteration four new members are sent to the fitness function. Also, a 4 hours duration limit was provided to terminate the execution if exceeded.

2.4. Fitness calculation

Each fitness function trains a model and evaluates its performance. Accuracy is one of the most important metrics in classification and it is calculated using Eq. 2 for each class.

$$accuracy_i = \frac{TP_i + TN_i}{TP_i + FP_i + TN_i + FN_i} \tag{2}$$

Where i is the index of the class (label), TP_i is the count of correct predictions that actually belong to class i, and FP_i is the count of wrong predictions that actually belong to classes else than class i. In addition, TN_i is the count of correct predictions that actually belongs to classes else than class i, and FN_i is the count of wrong predictions that actually belong to class i.

In multi-class classifications, overall macro accuracy is calculated by Eq. (3) and gives general information about the overall accuracy of all classes. In a test with uniform distribution of samples, the Macro accuracy reflects the total performance of a model. This metric is the average of all classes.

$$Macro\ \ accuracy = \frac{\sum_{i=1}^{n} accuracy_i}{n}$$

$$Where\ \ n = \ number\ \ of\ \ classes \tag{3}$$

Fig. 7. Performance of the multi-class classification.

Similar to the Macro accuracy, Eq. (4) gives another metric called overall micro accuracy. This metric differs slightly from Macro accuracy and expresses the accuracy of all models considering the frequency of samples in each class. This metric is useful for imbalanced datasets to detect bias.

$$micro \quad accuracy \quad = \frac{\sum_{i=1}^{n} TP_i + TN_i}{n \quad \times \quad totalSamples}$$

$$Where \quad n = \quad number \quad of \quad classes \tag{4}$$

For multi-class classification and regression, the fitness function of each optimizer contains a training algorithm that performs 5-fold cross-validation and generates relevant machine learning metrics. The value of the fitness function is calculated by Eq. (5) for multi-class classification and Eq. (6) for regression. Each value is assumed as the cost that the optimizer tries to minimize.

$$cost \quad = \quad 1 - \quad \frac{Macro \quad accuracy \quad + \quad micro \quad accuracy}{2} \tag{5}$$

$$cost \quad = \quad |1 - \quad RSquared| \tag{6}$$

In some applications of deep learning such as the image classification for multiple classes, the probability of the estimation is used for evaluation. The result of each prediction is a vector with the same length as the number of classes. Next, a vector called one-hot vector that contains only one 1 value for the highest probability and the remaining values are zero. The difference of probabilities of predicting vector and the true vector gives the cross-entropy. The lower entropy value expresses the higher confidence in training. Moreover, the accuracy is calculated using one-hot vectors. The cost value of the fitness in Eq. (7) navigates optimizers to train models with higher accuracy and lower cross-entropy and Eq. (7) focuses on increasing validation accuracy. On the other hand, Eq. (8) helps to increase the validation accuracy. In other words, the cost for DNN based on accuracy and entropy is calculated by Eq. (7), and the cost for DNN based on validation accuracy is obtained by Eq. (8).

$$cost \quad = \quad 1 - \quad ((\frac{TA \quad + VA}{2}) \quad - \quad (\frac{TCE \quad + VCE}{2})) \tag{7}$$

Where *TA* and *TCE* show the train accuracy, and train cross entropy,

Fig. 8. Performance of the regression.

respectively. Besides *VA* and *VCE* parameters represent the validation accuracy and validation cross entropy, respectively.

$$cost = 1 - ValidationAccuracy \tag{8}$$

2.5. Tuning with exhaustive grid search

EGS is a method that attempts to explore the search space via uniformly separated regions. Although this method picks these splits in equal slices, skipping optimum values due to the limitation of splitting pivots can reduce the performance. For example, consider the learning parameters in the range between 0 and 1; this hyperparameter should be split by three. Suppose the optimum value of the learning rate is 0.2. While the EGS produces 0, 0.5, and 1 for the Exhaustive Grid, the method misses the optimum of 0.2. However, this method is a classic approach to approximate the optima of grid searches like the hyperparameter tuning problem. Also, there is a trade-off between the approximation accuracy and the size of the Exhaustive Grid. For each row in this grid, the algorithm must generate a solution. Eventually, the best solution and its corresponding row are chosen. Please note that increasing the number of hyperparameters and their splitting pivots increase the size of the Exhaustive Grid dramatically.

In this study, the search space of a training algorithm consists of its hyperparameters. As a result of the fact that the number of splitting pivots for continuous values can be infinite, selecting limited pivots leads to missing some information. Moreover, these pivots highly depend on the problem and can vary. As there is no general strategy for determining these pivots analytically, empirical or trial and error approaches would be helpful. In this study, for each training algorithm and dataset, the average of fitness calls in GWO and GA have been considered to specify the size of the corresponding Exhaustive Grid. Fig. 5 gives the number of fitness calls in metaheuristic approaches and EGS for Multiclass classifiers. Note that Eq. (1) denotes the number of rows of the Exhaustive Grid. So, this number is the result of the product of all hyperparameters. The closest values to the metaheuristic have been determined for EGS. Fig. 6 illustrates the information of metaheuristics and EGS for regression algorithms.

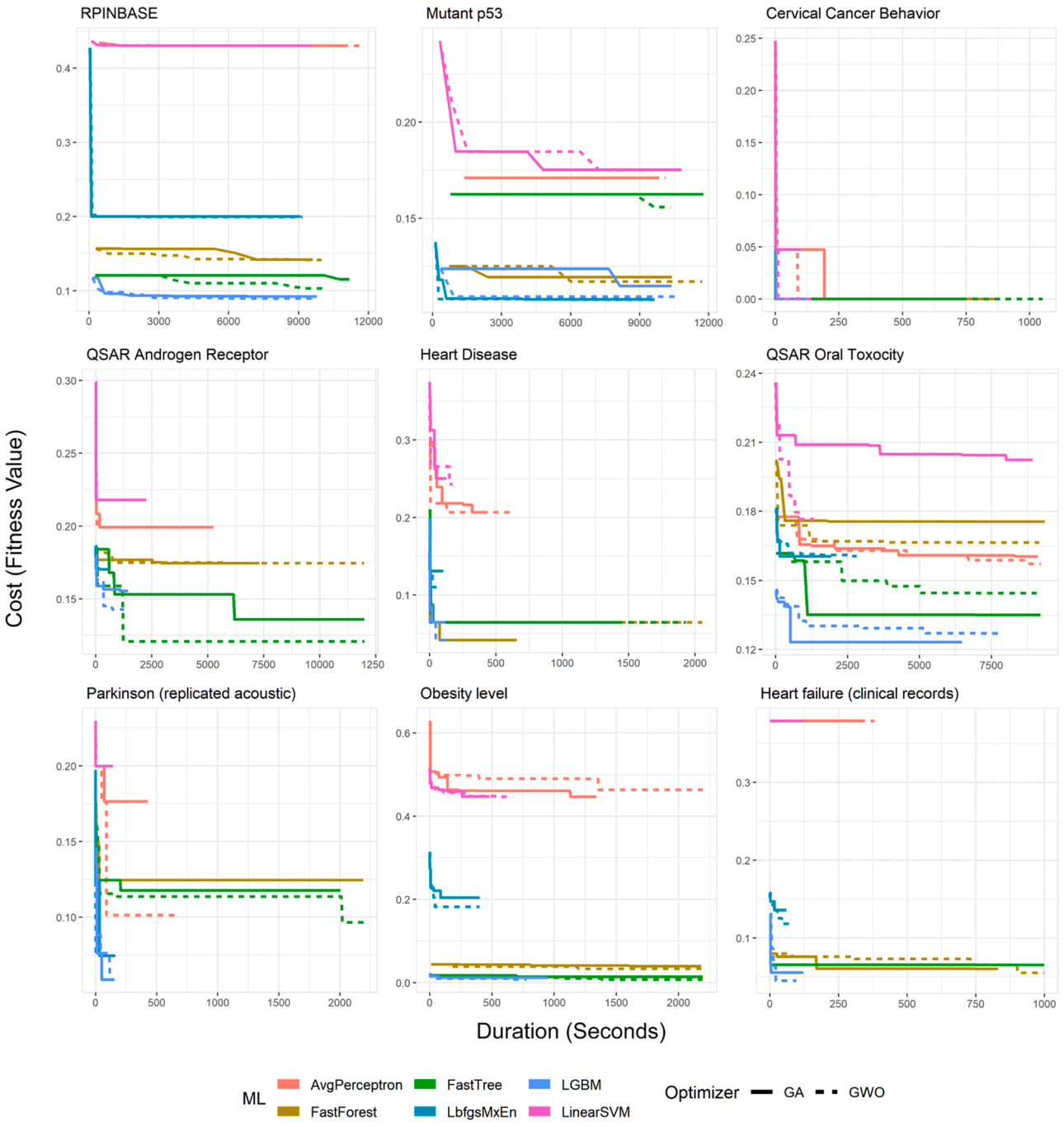


Fig. 9. Convergence curves of multi-class classification.

2.6. Experiment environment

To perform a fair benchmark between all experiments especially in durations and converging time, operations were accomplished on the same computer without any extra processes. The PC contains 12 logical processors with a 3.4 GHz base speed and 32 GB of RAM on Windows 10. The codes of this method have been written with C# and the ML.Net framework was employed to train and evaluate datasets.

3. Results

Despite the data cleaning is an important process of machine learning, but it highly depends on the data content of each dataset. To achieve a fair comparison between optimizers and algorithms, the results of this study are calculated without any data cleaning and extra regularizations. For multi-class classification and regression, 180 runs of optimizations have been done by nine datasets, two optimizers, and 10 training algorithms. To navigate the optimizer to train all classes having a balanced performance, the metrics of classifiers are calculated by the mathematical average of micro accuracy and macro accuracy and presented in percentage. On the other hand, in regression training algorithms, the measurements are based on RSquared. The performances of multi-class classification and regression are given in Figs. 7 and 8, respectively. Default performance is the achievement of each training algorithm using default and suggested arguments without any manipulation.

Figs. 9 and 10 show the convergence curves for classification and regression processes. In this study, there are two conditions to terminate

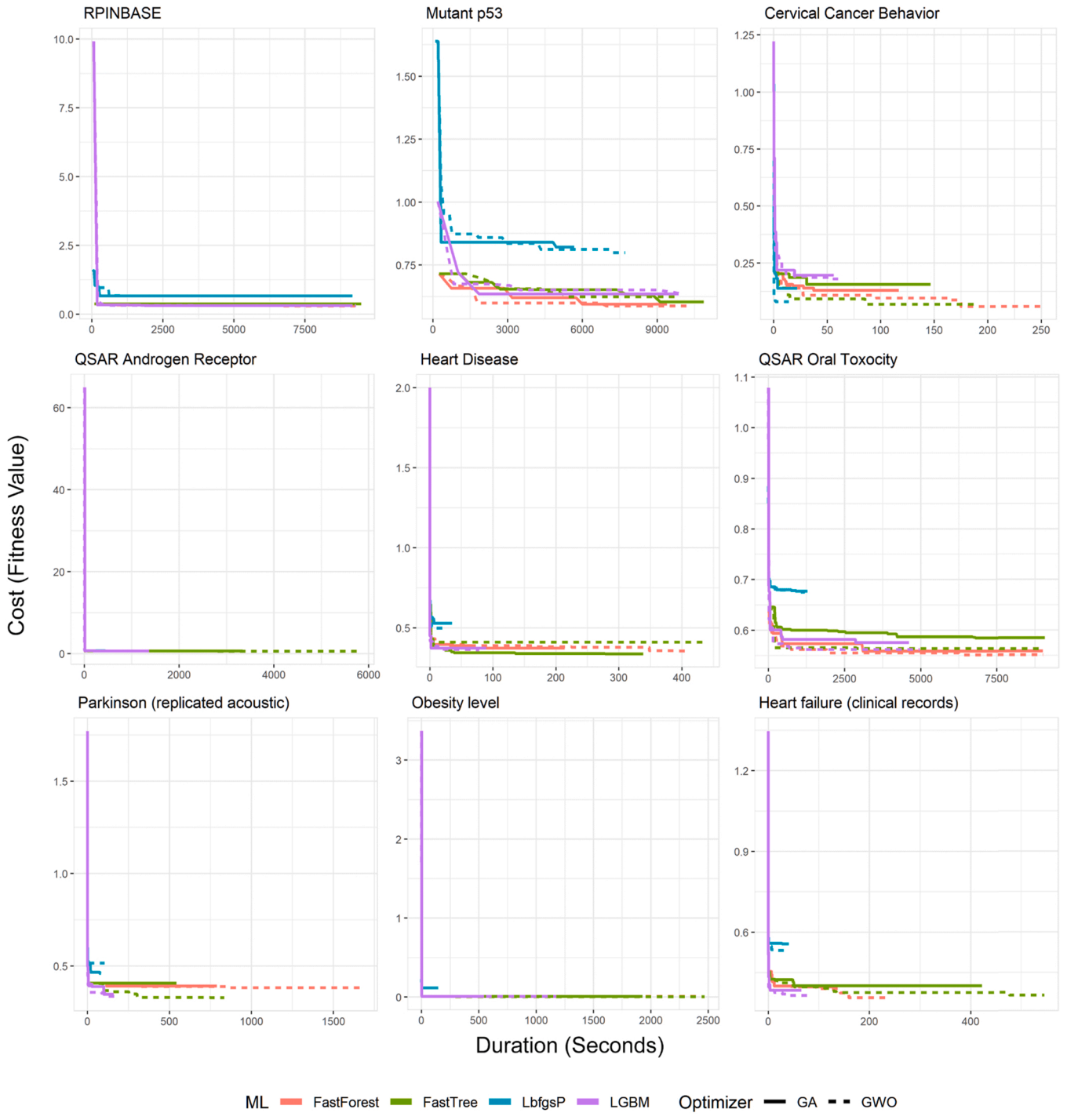


Fig. 10. Convergence curves of regression.

the optimization. First is a number that specifies the size of the maximum iteration. If the iteration counter reaches the size of the maximum iteration, the optimizer stops. The second condition is a deadline that works based on elapsed seconds. The motivation of this condition is that the duration of training for each dataset depends on the composition and content of the dataset and the machine learning algorithm that tries to fit a model. Therefore, the duration is more reasonable rather than the iteration number.

Fig. 11 shows the convergence curves of GA, GWO, and EGS for multi-class classification. Each point in the Y-axis represents the best cost of the algorithm at the time. The gaps at the beginning of lines in some plots originate from delays that the algorithm should wait to complete its action. For example, if an Exhaustive Grid contains 500 rows, all 500 states should be inspected to obtain the best one. To compare the performances of training and tuning algorithms, an increasing number of EGS rows has been generated. Corresponding to the average number of fitness calls of GA and GWO, which are shown in Figs. 5 and 6, four fractions of 25%, 50%, 75%, and 100% have been built. For example, a set of 500 fitness calls of metaheuristics corresponds to the 125, 250, 375, and 500 fitness calls of EGS.

Any search agent in any iteration calls the fitness function once. Also, this fitness performs a train utilizing received hyperparameters from optimizer algorithm or Exhaustive Grid. Note that the number of EGS rows rarely fits the exact desired number, because the total rows` count

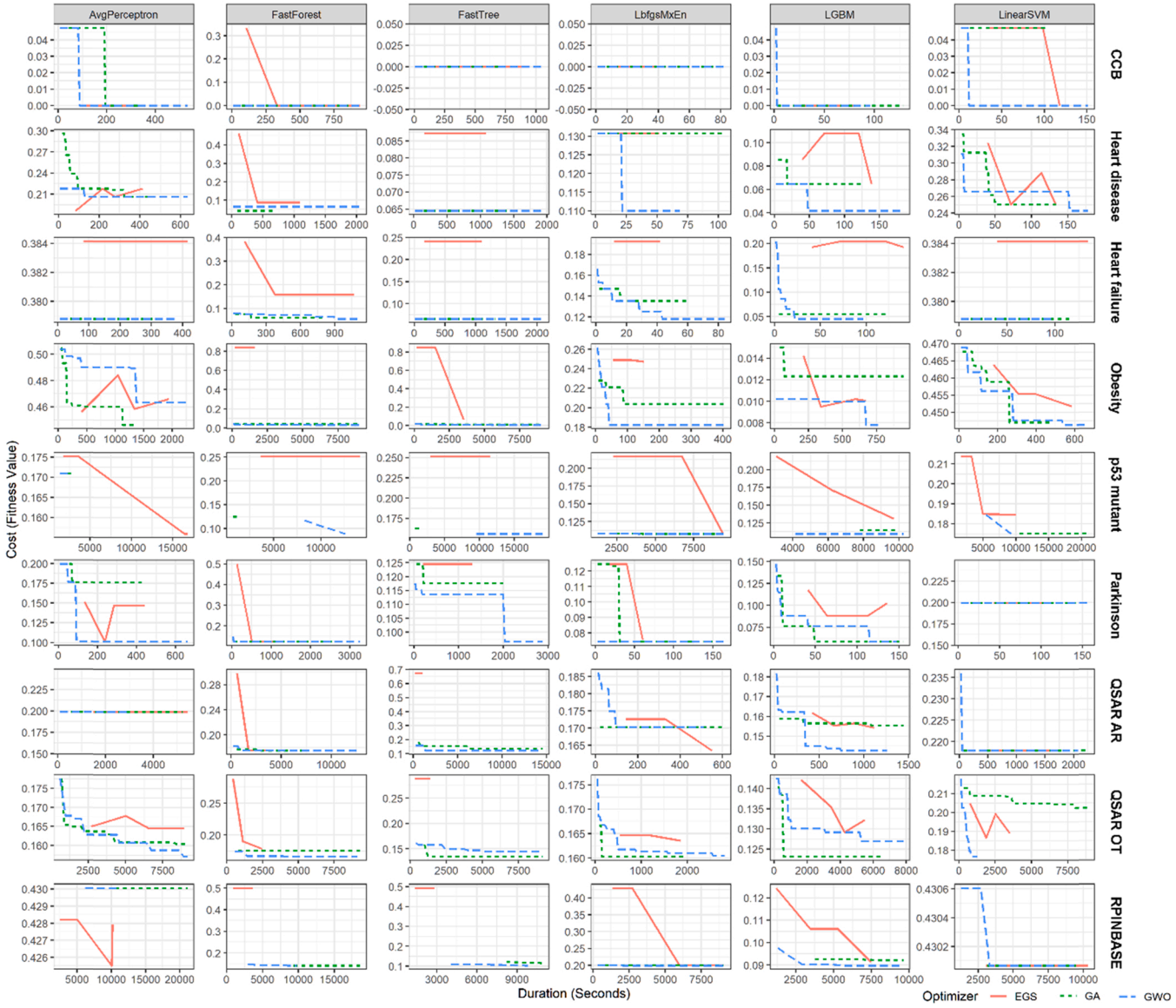


Fig. 11. Presentation of multi-class classification convergence curves between EGS, GA, and GWO.

equals the product of all possible states of hyperparameters. As a result, we considered the closest approximations of EGS to the metaheuristic approaches. Besides, Fig. 12 demonstrates the convergence curves for regression. Furthermore, the composition and values of hyperparameters can change the duration of training. So, the number of fitness calls does not always suit the same durations. Please notice that in some EGS plots, increasing the accuracy (number of fractions) leads to decreasing performance. The reason is that the pivot misses the optimum values of the hyperparameter. For example, suppose a hyperparameter with the boundary of 0 and 1 and the optimum value of 0.5. In this case, a three-split (0, 0.5, 1) gives a better result than a four-split (0, 0.33, 0.66, 1).

Detailed results of all runs of multi-class classification and regression in GA and GWO are available in Supplementary file 1. Moreover, the p-value of the t-test between whole GWO and GA costs is 2.6E-5. Fig. 13(a) illustrates the correlation clustering of multi-class classification and Fig. 13(b) shows the correlations between regression algorithms. Moreover, in all cases of 180 experiments of this study, the results have been improved. The general performance of multiclass classification is given in Table 3. Also, Table 4 shows the performances of regression experiments. In both tables, the bolded numbers demonstrate the best result for each dataset. Also, the italic font represents the shared best performances between multiple algorithms.

Finally, in image classification using DNN, two collections of images were selected. The first set consists of Covid19 X-rays and the second contains dermatoscopic images of common pigmented skin lesions. Similar to above, GA and GWO optimizers try to tune four architectures of DNN. The overall result of 32 metric groups (four architectures by two optimizers by two datasets by two fitness equations) of training and validation are given in Table 5. While Eq. (7) calculates the fitness using accuracy and cross-entropy of train and validation sets, Eq. (8) focuses on validation accuracy. For a better sense, the values of Eq. (8) in Table 5 are converted from cost to accuracy score.

4. Discussion

In this section, the obtained results are analyzed and discussed. Typically, hyperparameters play a crucial role to build models in machine learning problems. These parameters boost the discrimination power in classifiers and the performance of fitting curves in regression. All machine learning algorithms in this study receive a series of arguments and by default, they use some predefined values. Although some

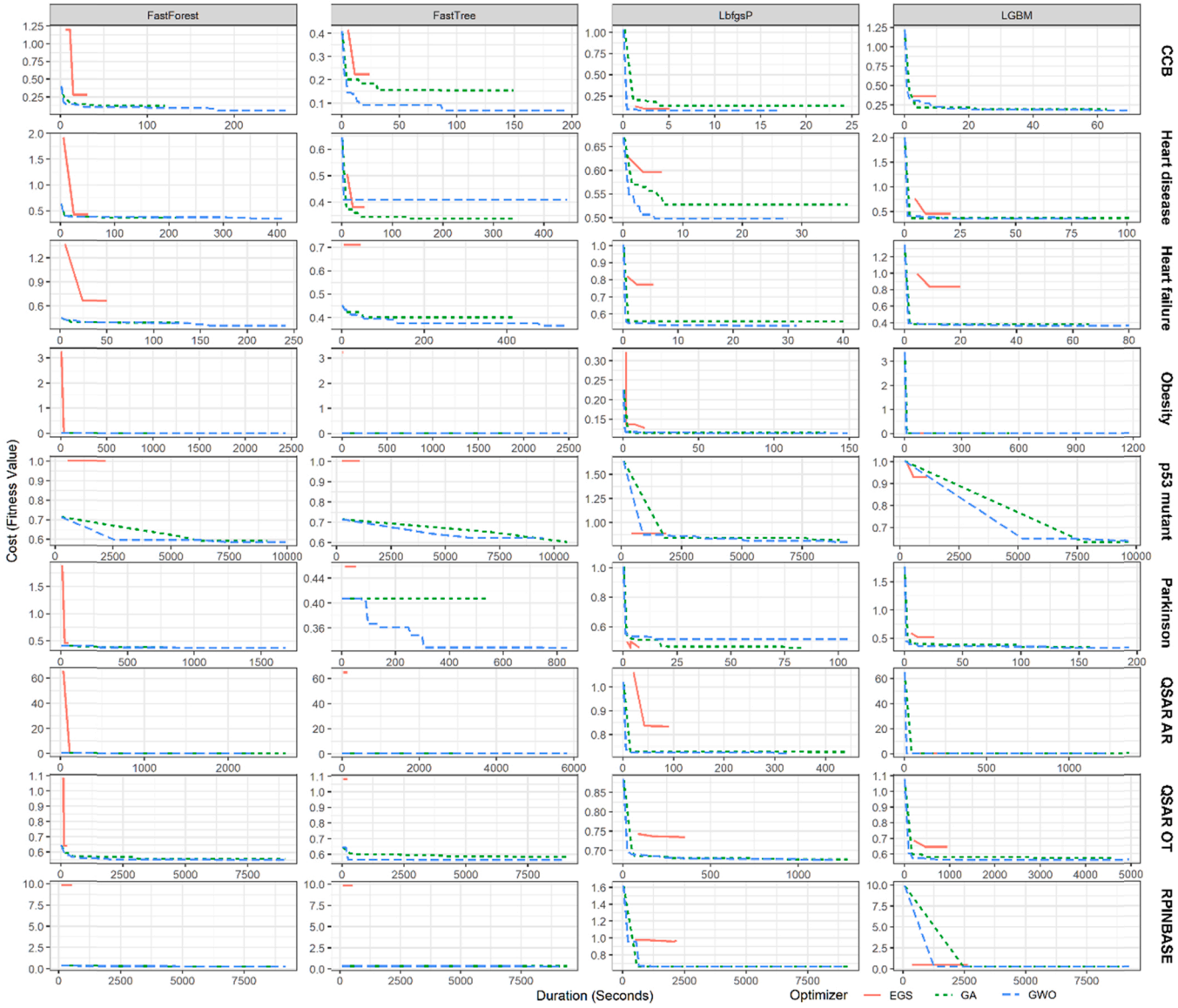
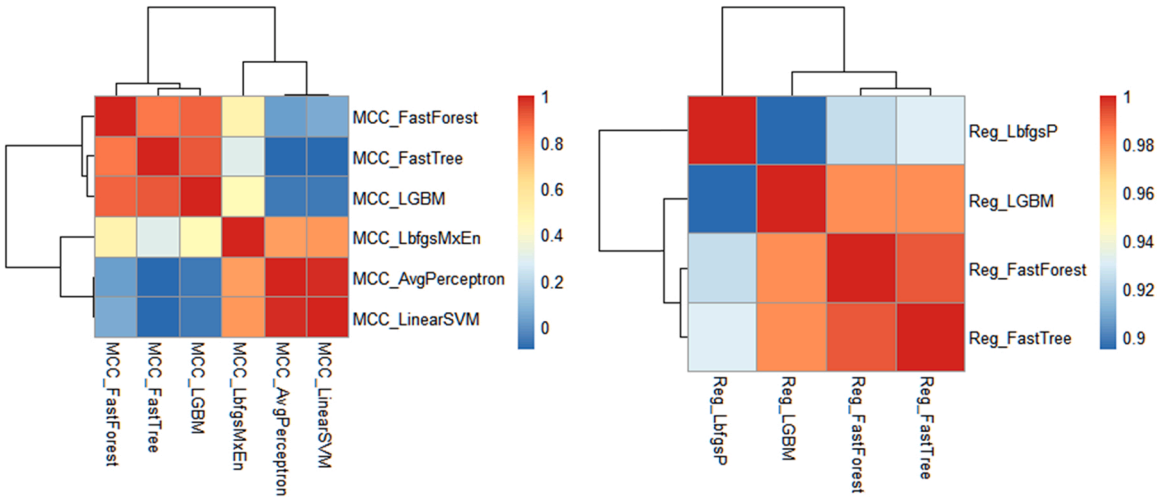


Fig. 12. Presentation of regression convergence curves between EGS, GA, and GWO.

algorithms would not express a good quality with default arguments, they can increase their performances by tuning arguments. As it is seen in most plots of Fig. 7, while some algorithms such as averaged perceptron and linear SVM behave more conservative with a narrow variance in different populations of optimizers, some other algorithms like Fast Tree, Fast Forest, and LGBM are more flexible and vary wider. In more conservative algorithms, the default metrics, which are shown by red points, are close to the best and worst cases. Furthermore, the performances of various datasets in Fig. 7 show that none of the algorithms are better than others deterministically. It depends on data content and features. So, examining different algorithms for different datasets makes sense especially for non-experts in statistics and data science algorithms. These inferences are reasonable for regression in Fig. 8. For example, the LGBM gives the worst performances by default arguments, but it can be improved dramatically by tuning its parameters and arguments.

Generally, metaheuristics are used to find optima in several problems. Here, they are employed to reduce the cost and increase the performance of training algorithms. The convergence curves of different algorithms for datasets with different scales are given in Figs. 9 and 10. Despite there is no guarantee for optimizers to find the best analytical optima, but after some iterations and time, the costs are decreased gradually. The time and iterations depend on the scales of datasets, algorithms, and their arguments. By choosing a suitable number of resources such as hardware and time, curves can be converged more quickly. Also, as an alternative strategy, by observing the performances of several algorithms, the winner algorithm in the first iterations can be focused and preempt the resources to exploit better. Also, by tracking improvements in starting times of curves and because initializing members of the optimizer are generated randomly, this method acts better than blind and random approaches.

Since the EGS has no strategy to navigate the results toward the optima, it can be considered a blind approach. Fig. 11 suggests that in only 7% of instances of multi-class classification (4 out of 54), the EGS converges faster than metaheuristic approaches. Similarly, in 7% of plots, the results are close. In the remaining 86% of training algorithms and datasets, metaheuristic algorithms (GA and GWO) converge faster and perform better than the EGS. Also, Fig. 12 implies that in about 92% of training algorithms and datasets (33 out of 36), metaheuristic algorithms converge quicker than the EGS. While none of the metaheuristic algorithms perform ascending convergence lines, there are some fluctuations in EGS. This undesired situation happens when splitting pivots skips optima hyperparameters. Therefore, increasing the number of



a) Correlation clustering of multiclass classification b) Correlation clustering of regression

Fig. 13. Correlation clustering of multiclass classification and regression algorithms.

Table 3
Performances of classifiers.

Classification training algorithm / Dataset		RPINBASE	p53	CCB	QSAR AR	Heart Disease	QSAR OT	Parkinson	Obesity	Heart Failure
AvgPerceptron	Default	56.6	82.9	75.2	77.3	70.3	82.2	77.0	37.1	62.1
	Best GA	57.0	82.9	***100***	80.1	79.3	84.0	82.4	55.4	62.1
	Best GWO	57.0	82.9	***100***	80.1	79.3	84.3	89.9	53.7	62.1
	Best EGS	57.6	84.4	***100***	80.1	79.3	83.2	85.3	59.1	61.6
FastForest	Default	84.3	87.5	80.5	81.5	87.9	79.8	82.4	95.6	92.0
	Best GA	85.8	88.1	***100***	82.6	**95.8**	82.5	87.5	96.1	93.9
	Best GWO	85.8	**91.1**	***100***	82.6	93.5	83.4	87.5	96.9	94.5
	Best EGS	50.0	74.8	***100***	82.5	91.3	82.2	87.5	16.1	84.2
FastTree	Default	87.9	83.8	94.4	81.6	78.9	83.8	83.9	98.3	93.5
	Best GA	88.4	83.8	***100***	86.4	93.5	86.5	88.2	99.0	93.5
	Best GWO	89.7	84.4	***100***	**87.9**	93.5	85.6	90.3	**99.5**	93.5
	Best EGS	50.8	74.8	***100***	32.5	91.3	71.2	87.5	15	75.9
LbfgsMxEn	Default	57.3	86.2	***100***	81.4	83.5	81.9	80.3	68.5	84.2
	Best GA	80.0	89.2	***100***	83.0	86.9	84.0	92.6	79.6	86.4
	Best GWO	80.1	89.2	***100***	83.0	89.0	83.9	92.6	81.8	88.2
	Best EGS	80.0	89.1	***100***	83.7	87	83.7	92.4	75.3	80.8
LGBM	Default	88.3	87.6	89.6	82.0	80.0	85.4	85.5	97.7	86.9
	Best GA	90.8	88.5	***100***	84.5	93.5	**87.7**	***94.1***	98.8	94.5
	Best GWO	**91.0**	89.1	***100***	85.7	**95.8**	87.3	***94.1***	99.2	**95.5**
	Best EGS	90.9	86.9	***100***	84.6	93.5	86.8	89.7	99.0	80.8
LinearSVM	Default	56.3	75.8	75.2	70.1	62.5	76.4	77.0	48.5	62.1
	Best GA	57.0	82.5	95.3	78.2	75.0	79.8	80.0	55.3	62.1
	Best GWO	57.0	82.5	***100***	78.2	75.7	82.3	80.0	55.4	62.1
	Best EGS	57.0	82.5	***100***	78.2	73.2	80.7	80.0	55.0	61.6

Table 4
Performances of regression training algorithms.

Regression training algorithm / Dataset		RPINBASE	p53	CCB	QSAR AR	Heart Disease	QSAR OT	Parkinson	Obesity	Heart Failure
FastForest	Default	0.371	0.715	0.408	0.724	0.644	0.646	0.408	0.007	0.454
	Best GA	0.317	0.594	0.129	0.646	0.373	0.559	0.392	0.006	0.397
	Best GWO	**0.305**	**0.588**	0.059	0.635	0.355	**0.551**	0.382	0.005	**0.357**
	Best EGS	9.826	1.003	0.281	0.666	0.431	0.640	0.456	0.007	0.671
FastTree	Default	0.371	0.715	0.408	0.724	0.644	0.646	0.408	0.007	0.454
	Best GA	0.369	0.603	0.156	0.662	**0.338**	0.584	0.408	0.006	0.402
	Best GWO	0.330	0.624	**0.068**	**0.616**	0.410	0.563	**0.329**	**0.004**	0.366
	Best EGS	9.826	1.003	0.225	65.035	0.381	1.082	0.459	3.230	0.712
LbfgsP	Default	1.619	1.639	1.033	1.021	0.669	0.884	1.007	0.226	1.006
	Best GA	0.666	0.821	0.138	0.727	0.528	0.677	0.459	0.116	0.556
	Best GWO	0.666	0.799	0.079	0.722	0.498	0.675	0.516	0.114	0.532
	Best EGS	0.974	0.889	0.099	0.834	0.596	0.735	0.459	0.128	0.773
LGBM	Default	9.917	1.003	1.222	65.035	2.000	1.079	1.771	3.371	1.347
	Best GA	0.313	0.635	0.195	0.668	0.372	0.575	0.346	0.006	0.384
	Best GWO	0.310	0.639	0.177	0.628	0.361	0.562	0.336	0.005	0.365
	Best EGS	0.468	0.930	0.363	0.741	0.462	0.647	0.520	0.007	0.837

Table 5
Performances of DNN training algorithm in optimizers.

	Dataset: Covid19 X-rays			Dataset: HAM10000		
	Default Fitness	Best Fitness	Improvement	Default Fitness	Best Fitness	Improvement
Eq. (6)	0.347 ± 1.3E-3	0.327 ± 3E-4	0.02	0.3532 ± 1E-4	0.3514 ± 1E-4	0.0018
Validation accuracy	91.12 ± 0.6 %	92.87 ± 0.96%	1.75 %	90.08 ± 0.4 %	91.33 ± 0.88%	1.25 %

splitting pivots in EGS does not improve the performance always.

To compare the GA and GWO, a t-test is held for total multi-class classifiers and regression. The null hypothesis for t-tests is represented by the following sentence: the best cost obtained from GWO is equal or greater than GA. The p-value is 2.6E-5 and it is less than 0.05. As a result, the null hypothesis is rejected and GWO is significantly better than GA in our experiments on employed datasets in this study. Therefore, a tuner utilizing GWO adequately covers the optimization part of the proposed method. Besides, to perform a faster tuning, instead of using randomly generated initialization search agents, previous experiences can be used. Furthermore, Table 3 suggests that the best performances can be obtained by all training algorithms. However, Averaged Perceptron and LinearSVM have a lower chance. Also, Table 4 shows that FastForest and FastTree are winners in regression.

The clustered and paired algorithms in Fig. 13 are useful when due to limitations, to reduce the number of candidate algorithms, eliminating similar algorithms is preferred. In multi-class classification in Fig. 13 (a), the FastTree and LGBM behave similarly. On the other hand, Averaged Perceptron and Linear SVM are grouped as the second highly correlated cluster. Finally, Fast Forest and Lbfgs classifiers correlated with a lower score. Moreover, the Lbfgs algorithm is different from the other three algorithms. Moreover, Fig. 13 (b) illustrates the correlation matrix of regression Performances. The FastTree and the FastForest behave similarly. This analysis suggests that if the number of training algorithms should be limited, a training algorithm can be elected between the most correlated training algorithms. The improvement of the DNN classifier for both Eqs. (7) and (8) is given in Table 5. In Eq. (7), optimizers try to decrease training and validation entropies and to increase the training and validation accuracies simultaneously. In Eq. (8), the goal is to improve the validation accuracy. For a better understanding of the readers, the values are recalculated and transformed into overall accuracy. Reducing the entropy causes a reduction in the confidence level of the model. It is useful for unseen samples for future predictions.

Most machine learning-based applications call prediction functions more frequently than training processes. So, it is a smart approach to train models as best as possible and consuming a reasonable number of resources at the training phase is ignorable. Also, in routine problems, the experience of successful models such as the combination of arguments can be used for new datasets in model updating processes.

5. Conclusion and future works

Machine learning algorithms take place among the most important algorithms in computer science. They learn using previous observations and predict new incoming samples. The accuracy metric explains how a model is successful to learn and generalize. Besides, some other metrics and parameters such as entropy and training duration can be taken into consideration. Once a model is trained well, it can be used repeatedly. Generally, machine learning users try to generate a reliable model by increasing train and validation accuracy and diminishing the confidence level by decreasing entropy. In addition to the nature of features in datasets, the performance of the training phase in machine learning problems highly depends on how they discriminate samples. Some internal parameters like learning rate, which are called hyperparameters, control the learning process. Considering the wide range of features in different applications, despite estimating proper values for parameters would be difficult for ordinary users, these have an important role in the efficiency and performance of models. This research proposes a method to tune controlling hyperparameters of machine learning algorithms to obtain a better performance. The idea is that hyperparameters are mapped to dimensions of an optimizer population and the optimizer tries to minimize the cost of each algorithm. The best member of the optimizer contains the best configuration of hyperparameters among explored combinations. Totally in this study, hyperparameters of 10 algorithms of machine learning are optimized using GA, GWO, and EGS utilizing nine different datasets. Also, the 11th algorithm (DNN image classification algorithm using four architectures) is used in this study to classify two image datasets. Our results show that in all cases except a little portion of simple datasets that already obtain the best result with default parameters, the method improves performances. In most cases of this study, the metaheuristic algorithms perform better and converge faster than EGS. This auto tuner method is strongly suggested for machine learning users that do not deal with algorithms in depth. The most proper algorithm dataset and its hyperparameters configuration for a specific can be returned as the output of this method.

Investigating a wider range of several machine learning training algorithms and different metaheuristics have been defined as the most important future works for this study. Also, providing a ready-to-use library and packages for different platforms and also an online web tool is in plans for the future. Furthermore, different internal parameters of deep learning such as kernel and filter sizes can be the point of interest for future studies.

Consent for publication

Not applicable.

Ethics approval and consent to participate

Not applicable

Funding

No funding.

CRediT authorship contribution statement

Sajjad Nematzadeh: Conceptualization, Methodology, Visualization, implementation, Formal analysis, Investigation, Writing – review & editing the manuscript. **Farzad Kiani**: Conceptualization, Investigation, Writing – review & editing the manuscript. **Mahsa Torkamanian-Afshar**: Conceptualization, Investigation. **Nizamettin Aydin**: Administration, Conceptualization, Supervision, Writing – review & editing, and revising the manuscript.

Competing interests

The authors have no competing interests.

Data Availability

Supplementary File 1. Detailed results of all runs of multi-class classification and regression in GA and GWO – (MLTunerResults.zip) – (https://github.com/sajjad-nematzadeh/MLTuner).

Acknowledgements

Not applicable.

Appendix A. Supporting information

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.compbiolchem.2021.107619.

References

Abe, S., 2010a. Two-Class Support Vector Machines (Support Vector Machines for Pattern Classification), in Two-Class Support Vector Machines (Support Vector Machines for Pattern Classification), Springer, London, pp. 21–112. doi: 10.1007/978-1-84996-098-4_2.

Abe, S., 2010b. Multiclass Support Vector Machines (Support Vector Machines for Pattern Classification), in Multiclass Support Vector Machines (Support Vector Machines for Pattern Classification), Springer, London, pp. 113–161. doi: 10.1007/978-1-84996-098-4_3.

Abe, S., 2010c. Support Vector Machines for Pattern Classification. Springer, London.

Ahmed, Z., et al., 2019. Machine Learning at Microsoft with ML.NET, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 2448–2458, May, doi: 10.1145/3292500.3330667.

Alawad, W., Zohdy, M., Debnath, D., 2018. Tuning hyperparameters of decision tree classifiers using computationally efficient schemes, in: Proceedings of the 2018 1st IEEE International Conference on Artificial Intelligence and Knowledge Engineering, AIKE 2018, Nov., pp. 168–169. doi: 10.1109/AIKE.2018.00038.

Bacanin, N., Bezdan, T., Tuba, E., Strumberger, I., Tuba, M., 2020. Optimizing convolutional neural network hyperparameters by enhanced swarm intelligence metaheuristics. Algorithms 13 (3), 67. https://doi.org/10.3390/a13030067 (Mar.).

Ballabio, D., Grisoni, F., Consonni, V., Todeschini, R., 2019. Integrated QSAR models to predict acute oral systemic toxicity. Mol. Inform. 38 (8–9), 1800124 https://doi.org/10.1002/minf.201800124 (Aug.).

Canziani, A., Paszke, A., Culurciello, E., 2016. An Analysis of Deep Neural Network Models for Practical Applications, May. [Online]. Available: https://arxiv.org/abs/1605.07678v4. (Accessed: Aug. 10, 2021).

Chamasemani, F.F., Singh, Y.P., 2011. Multi-class Support Vector Machine (SVM) classifiers - an application in hypothyroid detection and classification, in: Proceedings of the 2011 6th International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2011, pp. 351–356. doi: 10.1109/BIC-TA.2011.51.

Chicco, D., Jurman, G., 2020. Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. BMC Med. Inform. Dec. Mak. 20 (1), 16. https://doi.org/10.1186/s12911-020-1023-5 (Feb.).

Danziger, S.A., et al., 2009. Predicting positive p53 cancer rescue regions using Most Informative Positive (MIP) active learning. PLoS Comput. Biol. 5 (9), 1000498 https://doi.org/10.1371/journal.pcbi.1000498 (Sep.).

Darwish, A., Ezzat, D., Hassanien, A.E., 2020. An optimized model based on convolutional neural networks and orthogonal learning particle swarm optimization algorithm for plant diseases diagnosis. Swarm Evolut. Comput. 52, 100616 https://doi.org/10.1016/j.swevo.2019.100616 (Feb.).

Detrano, R., et al., 1989. International application of a new probability algorithm for the diagnosis of coronary artery disease. Am. J. Cardiol. 64 (5), 304–310. https://doi.org/10.1016/0002-9149(89)90524-9 (Aug.).

Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., Cosar, A., 2019. A survey on new generation metaheuristic algorithms. Comput. Indus. Eng. 137, 106040 https://doi.org/10.1016/j.cie.2019.106040 (Nov.).

Duarte, E., Wainer, J., 2017. Empirical comparison of cross-validation and internal metrics for tuning SVM hyperparameters. Pattern Recognit. Lett. 88, 6–11. https://doi.org/10.1016/j.patrec.2017.01.007 (Mar.).

Fernandes, K., Cardoso, J.S., Fernandes, J., 2017. Transfer learning with partial observability applied to cervical cancer screening, in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 10255 LNCS, pp. 243–250. doi: 10.1007/978-3-319-58838-4_27.

Fuchs, C., Spolaor, S., Nobile, M.S., Kaymak, U., 2019. A Swarm Intelligence Approach to Avoid Local Optima in Fuzzy C-Means Clustering, in IEEE International Conference on Fuzzy Systems, Jun., vol. 2019-June. doi: 10.1109/FUZZ-IEEE.2019.8858940.

Godínez-Bautista, A., Padierna, L.C., Rojas-Domínguez, A., Puga, H., Carpio, M., 2018. Bio-inspired metaheuristics for hyper-parameter tuning of support vector machine classifiers, in Studies in Computational Intelligence, vol. 749, Springer Verlag, pp. 115–130. doi: 10.1007/978-3-319-71008-2_10.

Y. Goldberg and M. Elhadad, Learning Sparser Perceptron Models, Acl, 2011, Accessed: Aug. 10, 2021. [Online]. Available: ⟨http://www.cs.bgu.ac.il/~yoavg/publications/acl2011sparse.pdf⟩.

Grisoni, F., Consonni, V., Ballabio, D., 2019. Machine learning consensus to predict the binding to the androgen receptor within the CoMPARA project. J. Chem. Inform. Model. 59 (5), 1839–1848. https://doi.org/10.1021/acs.jcim.8b00794 (May).

Halim, A.H., Ismail, I., Das, S., 2021. Performance assessment of the metaheuristic optimization algorithms: an exhaustive review. Artif. Intell. Rev. 54 (3), 2323–2409. https://doi.org/10.1007/s10462-020-09906-6 (Mar.).

Hussain, K., Mohd Salleh, M.N., Cheng, S., Shi, Y., 2019. Metaheuristic research: a comprehensive survey. Artif. Intell. Rev. 52 (4), 2191–2233. https://doi.org/10.1007/s10462-017-9605-z (Dec.).

Hutter, F., Hoos, H., Leyton-Brown, K., 2014. An efficient approach for assessing hyperparameter importance. PMLR 754–762 (Jan. 27).

Kabir Anaraki, A., Ayati, M., Kazemi, F., 2019. Magnetic resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms. Biocybern. Biomed. Eng. 39 (1), 63–74. https://doi.org/10.1016/j.bbe.2018.10.004 (Jan.).

Ke, G., et al., 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree, in Advances in neural information processing systems 30, pp. 3146–3154. [Online]. Available: https://github.com/Microsoft/LightGBM. (Accessed: Sep. 28, 2020).

Khanduja, N., Bhushan, B., 2021. Recent advances and application of metaheuristic algorithms: A survey (2014–2020), in Studies in Computational Intelligence, vol. 916, Springer Science and Business Media Deutschland GmbH, pp. 207–228. doi: 10.1007/978-981-15-7571-6_10.

Lanjanian, H., et al., 2021. High-throughput analysis of the interactions between viral proteins and host cell RNAs. Comput. Biol. Med. 135, 104611 https://doi.org/10.1016/J.COMPBIOMED.2021.104611 (Aug.).

Lee, W.Y., Park, S.M., Sim, K.B., 2018. Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm. Optik 172, 359–367. https://doi.org/10.1016/j.ijleo.2018.07.044 (Nov.).

Lentzas, A., Nalmpantis, C., Vrakas, D., 2019. Hyperparameter tuning using quantum genetic algorithms, in: Proceedings of the International Conference on Tools with Artificial Intelligence, ICTAI, Nov., vol. 2019-Novem, pp. 1412–1416. doi: 10.1109/ICTAI.2019.00199.

Liu, D.C., Nocedal, J., 1989. On the limited memory BFGS method for large scale optimization. Math. Program. 45 (1), 503–528. https://doi.org/10.1007/BF01589116.

Lujan-Moreno, G.A., Howard, P.R., Rojas, O.G., Montgomery, D.C., 2018. Design of experiments and response surface methodology to tune machine learning hyperparameters, with a random forest case-study. Expert Syst. Appl. 109, 195–205. https://doi.org/10.1016/j.eswa.2018.05.024 (Nov.).

Mcdonald, R., Hall, K., Mann, G., 2010. Distributed training strategies for the structured perceptron. doi: 10.5555/1857999.1858068.

Minaee, S., Kafieh, R., Sonka, M., Yazdani, S., Jamalipour Soufi, G., 2020. Deep-COVID: predicting COVID-19 from chest X-ray images using deep transfer learning. Medical Image Analysis 65, 101794. https://doi.org/10.1016/j.media.2020.101794 (Oct.).

Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey wolf optimizer. Adv. Eng. Softw. 69, 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007 (Mar.).

Naranjo, L., Pérez, C.J., Campos-Roca, Y., Martín, J., 2016. Addressing voice recording replications for Parkinson's disease detection. Expert Syst. Appl. 46, 286–292. https://doi.org/10.1016/j.eswa.2015.10.034 (Mar.).

Neary, P.L., 2018. Automatic hyperparameter tuning in deep convolutional neural networks using asynchronous reinforcement learning, in: Proceedings of the 2018 IEEE International Conference on Cognitive Computing, ICCC 2018 - Part of the 2018 IEEE World Congress on Services, Sep., pp. 73–77. doi: 10.1109/ICCC.2018.00017.

Palechor, F.M., de la H. Manotas, A., 2019. Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from Colombia, Peru and Mexico, Data in Brief, vol. 25, p. 104344, Aug., doi: 10.1016/j.dib.2019.104344.

Pan, S.J., Yang, Q., 2010. A survey on transfer learning. IEEE Trans. Knowl. Data Eng. 22 (10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191.

Passos. L.A., Rodrigues, D.R., Papa, J.P., 2018. Fine tuning deep boltzmann machines through meta-heuristic approaches, in: SACI 2018 - IEEE 12th International Symposium on Applied Computational Intelligence and Informatics, Proceedings, Aug., pp. 419–424. doi: 10.1109/SACI.2018.8440959.

Price, M.N., Dehal, P.S., Arkin, A.P., 2009. FastTree: computing large minimum evolution trees with profiles instead of a distance matrix. Mol. Biol. Evolut. 26 (7), 1641–1650. https://doi.org/10.1093/molbev/msp077 (Jul.).

Probst, P., Bischl, B., 2019. Tunability: importance of hyperparameters of machine learning algorithms. J. Mach. Learn. Res. 20, 1–32.

P. Probst, M. N. Wright, and A. Boulesteix, Hyperparameters and tuning strategies for random forest, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 9, no. 3, p. e1301, May 2019, doi: 10.1002/widm.1301.

E.-G. Talbi, Optimization of deep neural networks: a survey and unified taxonomy, Jun. 2020.

Torkamanian-Afshar, M., Nematzadeh, S., Tabarzad, M., Najafi, A., Lanjanian, H., Masoudi-Nejad, A., 2021. In silico design of novel aptamers utilizing a hybrid method of machine learning and genetic algorithm. Mol. Divers. https://doi.org/10.1007/s11030-021-10192-9.

Torkamanian-Afshar, M., et al., 2020. RPINBASE: an online toolbox to extract features for predicting RNA-protein interactions. Genomics 112 (3), 2623–2632. https://doi.org/10.1016/j.ygeno.2020.02.013.

Tsai, C.-W., Fang, Z.-Y., 2021. An effective hyperparameter optimization algorithm for DNN to predict passengers at a metro station. ACM Trans. Internet Technol. 21 (2), 1–24. https://doi.org/10.1145/3410156 (Mar.).

Tschandl, P., 2018. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Harvard Dataverse. https://doi.org/10.7910/DVN/DBW86T.

Wah June, L., Abu Hassan, M., 2005. Modifications of the limited memory BFGS algorithm for large-scale nonlinear optimization - OKAYAMA UNIVERSITY SCIENTIFIC ACHIEVEMENT REPOSITORY. Math. J. Okayama Univ. 47 (1), 175–188 doi: http://doi.org/10.18926/mjou/33602.

Whitley, D., 1994. A genetic algorithm tutorial. Stat. Comput. 4 (2), 65–85. https://doi.org/10.1007/BF00175354 (Jun.).

Wu, J., Chen, X.Y., Zhang, H., Xiong, L.D., Lei, H., Deng, S.H., 2019. Hyperparameter optimization for machine learning models based on Bayesian optimization. J. Electron. Sci. Technol. 17 (1), 26–40. https://doi.org/10.11989/JEST.1674-862X.80904120 (Mar.).

Yang, L., Shami, A., 2020. On hyperparameter optimization of machine learning algorithms: theory and practice. Neurocomputing 415, 295–316. https://doi.org/10.1016/j.neucom.2020.07.061 (Nov.).

Yates, D., Islam, M.Z., 2021. FastForest: increasing random forest processing speed while maintaining accuracy. Inform. Sci. 557, 130–152. https://doi.org/10.1016/j.ins.2020.12.067 (May).

Zhang, J., Huang, Y., Wang, Y., Ma, G., 2020. Multi-objective optimization of concrete mixture proportions using machine learning and metaheuristic algorithms. Constr. Build. Mater. 253, 119208 https://doi.org/10.1016/j.conbuildmat.2020.119208 (Aug.).

Zhou, Y., et al., 2019. Exploring tunable hyperparameters for deep neural networks with industrial ADME data sets. J. Chem. Inform. Model. 59 (3), 1005–1016. https://doi.org/10.1021/acs.jcim.8b00671 (Mar).

Zhou, J., et al., 2021. Optimization of support vector machine through the use of metaheuristic algorithms in forecasting TBM advance rate. Eng. Appl. Artif. Intell. 97, 104015 https://doi.org/10.1016/j.engappai.2020.104015 (Jan.).