Mathematical modeling

Математическое моделирование

UDC 004.89
https://doi.org/10.32362/2500-316X-2022-10-2-59-74

RESEARCH ARTICLE

Application of bioinspired global optimization algorithms to the improvement of the prediction accuracy of compact extreme learning machines

Liliya A. Demidova [@], **Artyom V. Gorchakov**

MIREA – Russian Technological University, Moscow, 119454 Russia
[@] *Corresponding author, e-mail: demidova.liliya@gmail.com*

Abstract

Objectives. Recent research in machine learning and artificial intelligence aimed at improving prediction accuracy and reducing computational complexity resulted in a novel neural network architecture referred to as an extreme learning machine (ELM). An ELM comprises a single-hidden-layer feedforward neural network in which the weights of connections among input-layer neurons and hidden-layer neurons are initialized randomly, while the weights of connections among hidden-layer neurons and output-layer neurons are computed using a generalized Moore–Penrose pseudoinverse operation. The replacement of the iterative learning process currently used in many neural network architectures with the random initialization of input weights and the explicit computation of output weights significantly increases the performance of this novel machine learning algorithm while preserving good generalization performance. However, since the random initialization of input weights does not necessarily guarantee optimal prediction accuracy, the purpose of the present work was to develop and study approaches to intelligent adjustment of input weights in ELMs using bioinspired algorithms in order to improve the prediction accuracy of this data analysis tool in regression problems.

Methods. Methods of optimization theory, theory of evolutionary computation and swarm intelligence, probability theory, mathematical statistics and systems analysis were used.

Results. Approaches to the intelligent adjustment of input weights in ELMs were developed and studied. These approaches are based on the genetic algorithm, the particle swarm algorithm, the fish school search algorithm, as well as the chaotic fish school search algorithm with exponential step decay proposed by the authors. By adjusting input weights with bioinspired optimization algorithms, it was shown that the prediction accuracy of ELMs in regression problems can be improved to reduce the number of hidden-layer neurons to reach a high prediction accuracy on learning and test datasets. In the considered problems, the best ELM configurations can be obtained using the chaotic fish school search algorithm with exponential step decay.

Conclusions. The obtained results showed that the prediction accuracy of ELMs can be improved by using bioinspired algorithms for the intelligent adjustment of input weights. Additional calculations are required to adjust the weights; therefore, the use of ELMs in combination with bioinspired algorithms may be advisable where it is necessary to obtain the most accurate and most compact ELM configuration.

Keywords: neural networks, extreme learning machine, bioinspired algorithms, genetic algorithm, particle swarm optimization algorithm, fish school search algorithm, machine learning, regression analysis

• **Submitted:** 29.11.2021 • **Revised:** 22.12.2021 • **Accepted:** 01.03.2022

For citation: Demidova L.A., Gorchakov A.V. Application of bioinspired global optimization algorithms to the improvement of the prediction accuracy of compact extreme learning machines. *Russ. Technol. J.* 2022;10(2):59–74. https://doi.org/10.32362/2500-316X-2022-10-2-59-74

Financial disclosure: The authors have no a financial or property interest in any material or method mentioned.

The authors declare no conflicts of interest.

НАУЧНАЯ СТАТЬЯ

Применение биоинспирированных алгоритмов глобальной оптимизации для повышения точности прогнозов компактных машин экстремального обучения

Л.А. Демидова [@], **А.В. Горчаков**

МИРЭА – Российский технологический университет, Москва, 119454 Россия
[@] *Автор для переписки, e-mail: demidova.liliya@gmail.com*

Резюме

Цели. В результате современных исследований в машинном обучении, направленных на повышение точности и снижение вычислительной сложности алгоритмов анализа данных, была предложена новая архитектура искусственной нейронной сети – машина экстремального обучения. Это нейронная сеть прямого распространения с единственным скрытым слоем. В этой сети веса соединений между входными нейронами и нейронами скрытого слоя инициализируются случайно, а веса соединений между нейронами скрытого слоя и выходными нейронами вычисляются с использованием операции псевдообращения Мура – Пенроуза. Замена итерационного процесса обучения, присущего многим архитектурам нейронных сетей, на случайную инициализацию одной части весов и вычисление другой части делает рассматриваемый инструмент существенно более производительным, с сохранением хорошей обобщающей способности. Однако случайная инициализация входных весов не гарантирует оптимальной точности прогнозов. Цель работы – разработка и исследование подходов к интеллектуальной настройке входных весов в машинах экстремального обучения биоинспирированными алгоритмами для повышения точности прогнозов этого инструмента анализа данных в задачах восстановления регрессии.

Методы. Использованы методы теории оптимизации, теории эволюционных вычислений и роевого интеллекта, теории вероятностей и математической статистики, системного анализа.

Результаты. Разработаны и исследованы подходы к интеллектуальной настройке входных весов в машинах экстремального обучения, основанные на применении генетического алгоритма, алгоритма роя частиц, алгоритма поиска косяком рыб, алгоритма хаотического поиска косяком рыб с экспоненциальным убыванием шага, предложенного авторами настоящего исследования. Выявлено, что применение биоинспирированных алгоритмов способно улучшить точность прогнозов машин экстремального обучения в задачах восстановления регрессии, причем машине экстремального обучения с уточненными биоинспирированными алгоритмами весами требуется меньшее число нейронов на скрытом слое для достижения высокой точности прогнозов на тренировочных и тестовых наборах данных. С помощью хаотического алгоритма поиска косяком рыб

с экспоненциальным убыванием шага могут быть получены наилучшие конфигурации машин экстремального обучения в рассмотренных задачах.

Выводы. Полученные результаты показывают, что точность прогнозов машин экстремального обучения может быть улучшена посредством применения биоинспирированных алгоритмов интеллектуальной настройки входных весов. Для выполнения настройки весов требуются дополнительные вычисления, поэтому использование машин экстремального обучения в сочетании с биоинспирированными алгоритмами может быть целесообразно в тех областях, где необходимо получение наиболее точной и компактной конфигурации машины экстремального обучения.

Ключевые слова: нейронные сети, машины экстремального обучения, биоинспирированные алгоритмы, генетический алгоритм, алгоритм роя частиц, алгоритм поиска косяком рыб, машинное обучение, регрессионный анализ

• **Поступила:** 29.11.2021 • **Доработана:** 22.12.2021 • **Принята к опубликованию:** 01.03.2022

Для цитирования: Демидова Л.А., Горчаков А.В. Применение биоинспирированных алгоритмов глобальной оптимизации для повышения точности прогнозов компактных машин экстремального обучения. *Russ. Technol. J.* 2022;10(2):59–74. https://doi.org/10.32362/2500-316X-2022-10-2-59-74

Прозрачность финансовой деятельности: Авторы не имеют финансовой заинтересованности в представленных материалах или методах.

Авторы заявляют об отсутствии конфликта интересов.

INTRODUCTION

Due to the digitalization of the economy, an increasing number of enterprises are integrating intelligent modules into their products and enterprise information systems in order to automate and accelerate business processes. These modules often comprise decision support systems, expert systems, and prediction systems that use machine learning algorithms. Such algorithms are used to automate the discovery of hidden relationships in datasets to make predictions without human intervention. Due to the increasing amounts of data being processed by intelligent systems, there is a need to increase the efficient performance of machine learning algorithms while preserving the accuracy of automated decisions.

In studies of classification, regression analysis, and prediction of time series, a variety of efficient machine learning approaches have been proposed, e.g., *k*-nearest neighbors [1], support vector machines [2], various hybrid versions [3], random forests [4], and artificial neural networks [5], which allow a high decision accuracy to be achieved.

A widely used and efficient supervised machine learning approach uses artificial neural networks (ANN), which can automate decision-making via the evolution of a complex nonlinear system [6]. Numerous ANN architectures and training methods have been developed, which demonstrate high efficiency in solving classification problems using deep learning, convolutional layers, and dropout layers [7], as well as solving regression layers using hybrids of population- and gradient-based algorithms of ANN learning [8, 9]. To find the optical hyperparameters of ANNs, bio-inspired algorithms are often used. For example, the particle swarm optimization algorithm was used to optimize the hyperparameters of a recurrent network with long short-term memory to predict time series [10].

The ANN training problem reduces to a problem of minimizing a certain ANN loss function on a training dataset. Researchers and practitioners conventionally train ANNs using an iterative backpropagation method and algorithms based on stochastic gradient descent [11, 12]. However, with increasing amounts of analyzed data, the time required for the convergence of gradient methods increases, especially when using deep ANN architectures. In order to accelerate ANN training, Google researchers proposed a distributed stochastic gradient descent algorithm [13], in which subsets of the training dataset are placed at several slave nodes of the computational network. At each iteration, slave computational nodes compute multidimensional gradient matrices for the proper data subset. A master node receives the computed multidimensional gradient matrices from the slave nodes, subtracts the gradients from the ANN weight matrix, and then sends the updated multidimensional weight matrix to the slave nodes, after which the process is repeated until the stopping criterion is satisfied. However, even when using distributed ANN training methods, the training may take a long time: from several hours to several days.

To accelerate ANN training, Huang et al. [14] proposed a novel ANN architecture referred to as an extreme learning machine (ELM). An ELM comprises a single-hidden-layer ANN in which the weights of connections among input-layer neurons and hidden-layer neurons, as well as the hidden-layer shift vector, are initialized randomly, whereas

the weights of connections among hidden-layer neurons and output-layer neurons are computed using a generalized Moore–Penrose pseudoinverse operation [15]. Besides, the hidden-layer activation function should be infinitely differentiable [14]. By using an ELM, it is possible to eliminate the iterative process of optimizing the ANN loss function from the process of preparation of an ANN model capable of making accurate decisions, thus significantly reducing the computational effort to train the ANN. The absence of an iterative training process does not prevent the ELM from demonstrating good generalization performance in a number of applied problems [16–18].

However, the random initialization of the weights of connections among input-layer and hidden-layer neurons along with the hidden-layer shift vector does not guarantee the optimal configuration of an ELM. Instead, the problems of intelligent refinement of input weights and hidden-layer shift vector of ELMs may be solved using bioinspired optimization algorithms [16, 19]. Such algorithms are heuristic methods of global optimization that process several solutions at each iteration without using information on the derivative of the function for their optimization; this simplifies the launch of these algorithms in parallel and distributed modes [20]. Widely used bioinspired algorithms included the genetic algorithm (GA) [21], the particle swarm optimization (PSO) [22], the fish school search (FSS) [23, 24], and others. Cai et al. [16] used particle swarm optimization to obtain the optimal ELM configuration for traffic flow forecasting. Song et al. [19] carried out a comparative analysis of the classical realization of ELM and the ELM, in which the intelligent selection of input weights was performed using the genetic algorithm, demonstrating the high efficiency of the ELM configuration obtained via GA.

In the present work, the efficiencies of various bioinspired algorithms were studied, including the genetic algorithm, the particle swarm algorithm, the standard fish school search algorithm, as well as a chaotic fish school search algorithm with exponential step decay, which were used in the problem of choosing the optimal weights of connections among input-layer and hidden-layer neurons, as well as the hidden-layer shift vector. Regression estimation problems were considered on three datasets; the generalization performances were compared between the classical ELM [14] and the ELMs in which the intelligent adjustment of input weights was made using bioinspired algorithms. Additionally, landscapes optimized using bioinspired loss function algorithms were studied in the course of the refinement of the input weights and shifts in the ELMs.

EXTREME LEARNING MACHINE

In a supervised machine learning problem, a set of objects, $X = X_L \cup X_T$, is given, where X_L is a learning dataset, and X_T is a test dataset; also given are a set of possible answers, Y, and an unknown objective function, f: $X \to Y$, which maps the set of objects to the set of possible answers. The f values are known for each object in the set X. The set X_L is used during the training of the model, and the trained model quality is evaluated on the set X_T. The learning dataset X_L has the form $\{\vec{x}_1, \vec{x}_2, ..., \vec{x}_s\}$, where s is the number of objects in X_L; each (ith) object $\vec{x}_i \in X_L$ is represented as a set of features, $\vec{x}_i = (h_1, h_2, ..., h_n)$, which characterize the object $\vec{x}_i$; and the $\vec{y}_i$ values are known for each (ith) object $\vec{x}_i$ and are equal to $f(\vec{x}_i)$. In the course of the training, the machine learning algorithm constructs the function a: $X \to Y$ to sufficiently approximate the unknown objective function f on $X = X_L \cup X_T$.

An extreme learning machine (ELM) is a supervised machine learning algorithm, which, similarly to an ANN, can make decisions by automatically configuring a complex nonlinear system for a certain problem. It comprises a single-hidden-layer feedforward neural network with an infinitely differentiable activation function in the hidden layer [14] where the weights of connections among input-layer neurons and hidden-layer neurons, as well as hidden-layer shifts, are initialized randomly, whereas the weights of connections among hidden-layer neurons and output-layer neurons are computed. Figure 1 presents the structure of the ELM.

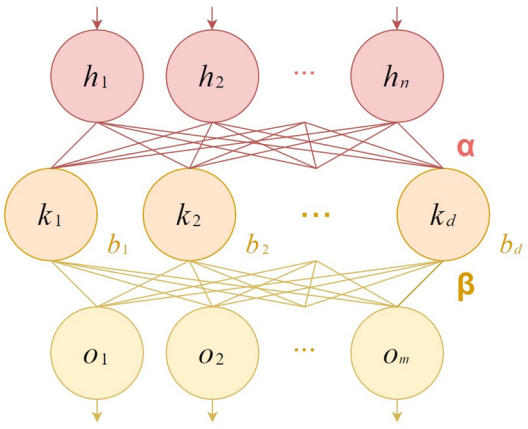


Fig. 1. Extreme learning machine

Elements of the matrix $\boldsymbol{\alpha}$ of the weights of connections among input-layer neurons $h_1, h_2, \ldots h_n$ and hidden-layer neurons $k_1, k_2, \ldots k_d$, having the form $\mathbb{R}^{d \times n}$, where n is the number of the input-layer neurons, and d is the number of the hidden-layer neurons,

are initialized randomly; shifts b_1, b_2, … b_d of the hidden-layer neurons are also initialized randomly, while the vector of the shifts has the form $\mathbb{R}^d$. The matrix $\boldsymbol{\beta}$ of the weights of connections among the hidden-layer neurons k_1, k_2, … k_d and output-layer neurons o_1, o_2, … o_m, which has the form $\mathbb{R}^{d\times m}$, where m is the number of the output-layer neurons, is calculated as

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{Y}_{\mathrm{L}}, \text{ where } \mathbf{H}^{\dagger} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}, \ \mathbf{H} = g(\mathbf{X}_{\mathrm{L}}\boldsymbol{\alpha}^{\mathrm{T}} + \mathbf{b}). \quad (1)$$

Here, $\mathbf{X}_{\mathrm{L}}$ is a matrix having the form $\mathbb{R}^{s\times n}$, in which the rows code s objects in the learning dataset X_{L}, with each object being represented as a set of n features $\{h_1, h_2, \ldots h_n\}$. $\boldsymbol{\alpha}^{\mathrm{T}}$ is the transposed matrix of the weights of connections among neurons of the form $\mathbb{R}^{n\times d}$, where n is the number of the input-layer neurons and d is the number of the hidden-layer neurons. $\mathbf{b}$ is a matrix of the form $\mathbb{R}^{s\times d}$ that is obtained by the transformation of the vector of shifts of the hidden-layer neurons of the form $\mathbb{R}^d$ to a matrix of the form $\mathbb{R}^{1\times d}$ in which the first row is repeated s times. g is an infinitely differentiable activation function, which is applied element-by-element to each element of the matrix. $\mathbf{H}$ is the output matrix of the hidden-layer neurons of the form $\mathbb{R}^{s\times d}$. $\mathbf{H}^{\dagger}$ is the Moore–Penrose pseudoinverse of the matrix $\mathbf{H}$ [15] of the form $\mathbb{R}^{d\times s}$. $\mathbf{Y}_{\mathrm{L}}$ is the matrix of answers of the form $\mathbb{R}^{s\times m}$, in which the row code answers correspond to objects in the learning dataset X_{L}, each answer having the form $\{o_1, o_2, \ldots o_m\}$. Thus, the matrix $\boldsymbol{\beta}$ of the weights of connections among d hidden-layer neurons and m output-layer neurons has the form $\mathbb{R}^{d\times m}$. For the hidden-layer activation function g in ELMs, the sigmoid activation function is often used [25]:

$$\sigma(x) = \frac{1}{1-\mathrm{e}^{-x}}. \quad (2)$$

Here, x is an element of the hidden-layer output matrix of the form $\mathbb{R}^{s\times d}$, which is obtained by the multiplication of the matrix $\mathbf{X}_{\mathrm{L}}$ and the matrix $\boldsymbol{\alpha}^{\mathrm{T}}$ with the subsequent addition of the matrix $\mathbf{b}$.

The number d of the hidden-layer neurons, which is an ELM hyperparameter, should be adjusted to the problem to be solved. The number n of features of each object in the learning dataset is determined according to the specificity of the domain of the problem being solved. In solving the regression estimation problem using the ELM considered in this work, the number m of output-layer neurons is taken to be 1.

INTELLIGENT SELECTION OF INPUT WEIGHTS USING BIOINSPIRED ALGORITHMS

The random initialization of the weights $\boldsymbol{\alpha}$ of connections among input-layer neurons and hidden-layer neurons, as well as the hidden-layer shifts $\mathbf{b}$, does not guarantee the optimal ELM configuration [16, 19]. Studies have been carried out in which particle swarm optimization was used to obtain the optimal ELM configuration in traffic flow forecasting problems [16], with a genetic algorithm being applied in regression estimation problems [19]. In these works, the root mean square error (RMSE) was selected as the objective function for the bioinspired algorithms:

$$\mathrm{RMSE} = \sqrt{\frac{1}{s}\sum_{i=1}^{s}(a(\vec{x}_i) - y_i)^2}, \quad (3)$$

where $\vec{x}_i$ is the ith object in the learning dataset X_{L} comprising s objects, y_i is the answer for the ith object, and $a(\vec{x}_i)$ is the prediction of the ELM for the ith object $\vec{x}_i$.

The generalization performance of a trained ANN model is often evaluated using the mean absolute error (MAE) function:

$$\mathrm{MAE} = \frac{1}{s}\sum_{i=1}^{s}\left|a(\vec{x}_i) - y\right|_i, \quad (4)$$

where $\vec{x}_i$ is the ith object in the learning dataset X_{L} comprising s objects, y_i is the answer for the ith object, and $a(\vec{x}_i)$ is the prediction of the ELM for the ith object. The mean absolute error function was used [16] to evaluate the ELM model quality.

When solving the problem of selecting the optimal values of the input weights given by the matrix $\boldsymbol{\alpha}$ of the form $\mathbb{R}^{d\times n}$ and the hidden-layer shift vector $\{b_1, b_2, \ldots, b_d\}$ of the form $\mathbb{R}^d$ (Fig. 1), each (ith) agent in the population of bioinspired algorithms can be represented as the vector $\{\alpha^i_{11}, \alpha^i_{12}, \ldots, \alpha^i_{1n}, \alpha^i_{21}, \alpha^i_{22}, \ldots, \alpha^i_{2n}, \ldots, \alpha^i_{d1}, \alpha^i_{d2}, \ldots, \alpha^i_{dn}, b^i_1, b^i_2, \ldots, b^i_d\}$.

The genetic algorithm used in the problem of intelligent selection of the input weights and hidden-layer shifts in the ELM [19] is a heuristic population optimization algorithm inspired by evolutionary processes occurring in biological nature. Algorithm 1 determines the pseudocode of the genetic algorithm. The selection, crossover, and mutation operators are sequentially applied to the agents in the population at each iteration using various selection strategies, e.g., tournament selection and truncation selection [21].

The particle swarm optimization (PSO) algorithm used in the problem of selecting the optimal values of the

input weights and hidden-layer shifts in the ELM [16] is the well-known global optimization algorithm proposed by Kennedy and Eberhart [22], which was inspired by the coordinated flight of bird flocks. Algorithm 2 determines the pseudocode of the particle swarm algorithm.

The fish school search (FSS) algorithm is a global optimization algorithm inspired by the behavior of fish schools swimming in search of food. The algorithm, which was proposed by Bastos Filho et al. [23], has found application in solving numerous problems, including image reconstruction [26] and distribution of weights in ANN [27].

The modification of the fish school search algorithm proposed by Demidova and Gorchakov [24], which converges faster than FSS, is called tent map-based fish school search with exponential step decay (ETFSS). The ETFSS pseudocode is determined by Algorithm 3.

In Algorithm 3, uniformly distributed random numbers for the vectors $\vec{r}_1$ and $\vec{r}_2$ are generated by the mapping tent using a dynamic system in a chaotic state. The dynamic system is described by time set T, state set S, and mapping M: $T \times S \to S$, which characterize the evolution of the dynamic system. The mapping tent is defined as

$$y_{n+1} = \mu \min(y_n, 1 - y_n), \text{ where } \mu = 1.9999. \quad (5)$$

Here, the number μ is a bifurcation parameter, and y_n determines the state of the dynamic system at time t.

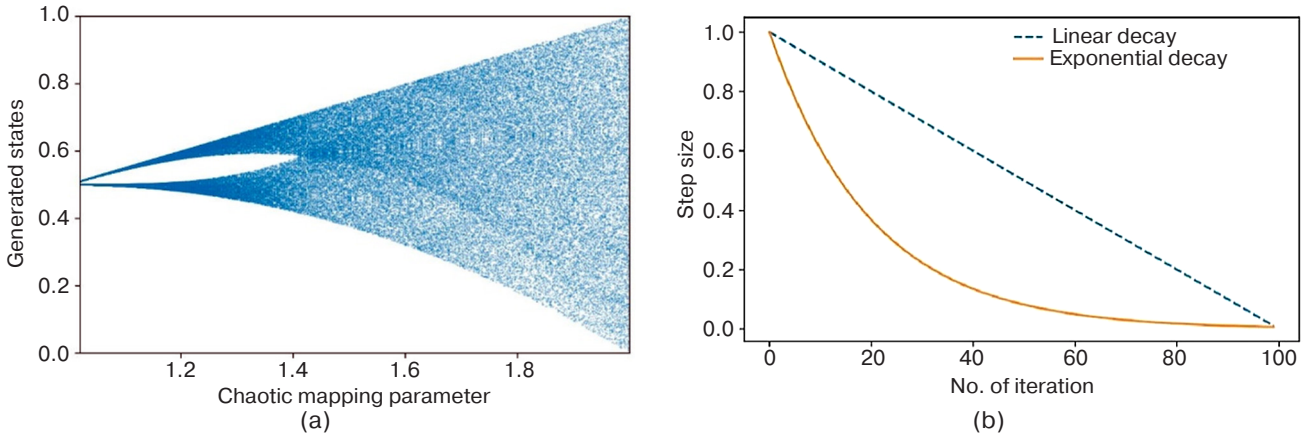


Fig. 2. (a) Bifurcation diagram of the dynamic system of mapping (5) and (b) the comparison of the linear and exponential step decays in FSS and ETFSS, respectively

Algorithm 1. Genetic algorithm

Start: $P_{\text{crossover}}$ is the crossover probability, P_{mutation} is the mutation probability.

1: define fitness function (f)
2: set the number of a generation to be 0 ($t = 0$)
3: randomly generate agents in the initial population P_t
4: calculate the f value for each agent in P_t
5: **until** the stopping criterion is satisfied, **do**
6: $t = t + 1$
7: *select* agents for the population P_t from the population P_{t-1}
8: change agents in P_t using the *crossover* operator with the probability $P_{\text{crossover}}$
9: change agents in P_t using the *mutation* operator with the probability P_{mutation}
10: calculate the f value for each agent in P_t
11: **complete cycle**
12: **return** the best evolved solution

Algorithm 2. Particle swarm algorithm*

Start: w, c_1, c_2, $v_{\max}$

1: define fitness function (f)
2: set the number of a generation to be 0 ($t = 0$)
3: randomly generate agents in the initial population P_t

4: randomly generate the velocities V_t of agents in the range $[-v_{max}, v_{max}]$
5: calculate the f value for each agent in P_t
6: select agent $\vec{p}_{gbest} \in P_t$, such that $\forall \vec{p}_{i,t} \in P_t : f(\vec{p}_{gbest,t}) \le f(\vec{p}_{i,t})$
7: for each particle $\vec{p}_{i,t} \in P_t$, set the best position $\vec{p}_{pbest,i}$ to be equal to $\vec{p}_{i,t}$
8: **until** the stopping criterion is satisfied, **do**
9: $t = t + 1$
10: **for each** agent $\vec{p}_{i,t} \in P_t$, **do**
11: randomly generate $\vec{r}_1$ and $\vec{r}_2$, the components of which $\in [0,1]$
12: $\vec{v}_{i,t} = w\vec{v}_{i,t-1} + c_1\vec{r}_1(\vec{p}_{gbest} - \vec{p}_{i,t}) + c_2\vec{r}_2(\vec{p}_{pbest,i} - \vec{p}_{i,t})$
13: restrict $\vec{v}_{i,t}$ to the range $[-v_{max}, v_{max}]$
14: $\vec{p}_{i,t} = \vec{p}_{i,t-1} + \vec{v}_{i,t}$
15: calculate the f value for the agent $\vec{p}_{i,t}$
16: **if** $f(\vec{p}_{i,t}) < f(\vec{p}_{pbest,i})$, **then** $\vec{p}_{pbest,i} = \vec{p}_{i,t}$
17: **complete cycle**
18: select agent $\vec{p}_{gbest,t} \in P_t$, such that $\forall \vec{p}_{i,t} \in P_t : f(\vec{p}_{gbest,t}) \le f(\vec{p}_{i,t})$
19: **if** $f(\vec{p}_{gbest,t}) < f(\vec{p}_{gbest})$, **then** $\vec{p}_{gbest} = \vec{p}_{gbest,t}$
20: **complete cycle**
21: **return** the best found solution $\vec{p}_{gbest}$

*gbest is the global best solution at iteration t in terms of particle swarm optimization algorithm and pbest is the best solution found by a certain agent (personal best) at iteration t in terms of particle swarm optimization algorithm.

Algorithm 3. Chaotic fish school search algorithm with exponential step decay*

Start: $step_{ind,initial}$, $step_{vol,initial}$, $\gamma = 5$

1: define fitness function (f)
2: set the number of a generation to be 0 ($t = 0$)
3: randomly generate agents in the initial population P_t
4: calculate the f value for each agent in P_t
5: select agent $\vec{p}_{gbest} \in P_t$, such that $\forall \vec{p}_{i,t} \in P_t : f(\vec{p}_{gbest,t}) \le f(\vec{p}_{i,t})$
6: **until** the stopping criterion is satisfied, **do**
7: $t = t + 1$
8: $step_{ind,t} = step_{ind,initial} e^{\frac{-\gamma t}{iter_{max}}}$
9: $step_{vol,t} = step_{vol,initial} e^{\frac{-\gamma t}{iter_{max}}}$
10: **for each** agent $\vec{p}_{i,t} \in P_t$, **do**
11: randomly generate $\vec{r}_1$ and $\vec{r}_2$, the components of which $\in [0,1]$
12: $\vec{p}_{i,t} = \vec{p}_{i,t-1} + step_{ind,t}\vec{r}_1$
13: calculate the f value for $\vec{p}_{i,t}$

14: **if** $f(\vec{p}_{i,t}) \geq f(\vec{p}_{i,t-1})$, **then** $\vec{p}_{i,t} = \vec{p}_{i,t-1}$

15: $w_{i,t} = w_{i,t-1} + \dfrac{\Delta f_{i,t}}{\max(\Delta f_{i,t+1})}$

16: **complete cycle**

17: $\vec{I} = \dfrac{\sum_{i=1}^{n}(\vec{p}_{i,t} - \vec{p}_{i,t-1})\Delta f_{i,t}}{\sum_{i=1}^{n}\Delta f_{i,t}}$

18: **for each** agent $\vec{p}_{i,t} \in P_t$, **do** $\vec{p}_{i,t} = \vec{p}_{i,t} + \vec{I}_t$

19: $\vec{B} = \dfrac{\sum_{i=1}^{n}\vec{p}_{i,t}w_{i,t}}{\sum_{i=1}^{n}\vec{p}_{i,t}}$

20: **for each** agent $\vec{p}_{i,t} \in P_t$, **do**

21: $\vec{p}_{i,t} = \vec{p}_{i,t} \pm step_{\text{vol},t}\vec{r}_2 \dfrac{\vec{p}_{i,t} - \vec{B}_{t+1}}{\left|\vec{p}_{i,t} - \vec{B}_{t+1}\right|}$; the sign is minus, if $\sum_{i=1}^{n} w_{i,t} > \sum_{i=1}^{n} w_{i,t-1}$; otherwise, the sign is plus

22: **complete cycle**

23: select agent $\vec{p}_{\text{gbest},t}$, such that $\forall \vec{p}_{i,t} \in P_t : f(\vec{p}_{\text{gbest},t}) \leq f(\vec{p}_{i,t})$

24: **if** $f(\vec{p}_{\text{gbest},t}) < f(\vec{p}_{\text{gbest}})$, **then** $\vec{p}_{\text{gbest}} = \vec{p}_{\text{gbest},t}$

25: **complete cycle**

26: **return** the best found solution $\vec{p}_{\text{gbest}}$

*$step_{\text{ind}}$ is the maximum step size of the individual movement in terms of the fish school optimization algorithm, $step_{\text{vol}}$ is the maximum step size of the collective-volitive movement in terms of the fish school optimization algorithm, *iter* is the number of iterations, and $iter_{max}$ is the maximum allowable number of iterations in the algorithm.

Figure 2a presents the bifurcation diagram of system (5).

Figure 2b compares the exponential step decay used in the ETFSS algorithm and the linear step decay used in the original fish school search algorithm. The efficiency of using a chaotic pseudorandom number generator (5) and exponential step decay (Fig. 2b) in the fish school search algorithm has been previously shown [24].

COMPUTATIONAL EXPERIMENT

To study the efficiencies of the algorithms GA, PSO, FSS, and ETFSS in the problem of distribution of input weights and shifts in the ELM and compare the obtained ELM configurations with the classical realization of ELM, in which the input weights and shifts are initialized randomly, three earlier described [28, 29] open datasets were considered. The first set, which contained Central Processing Unit (CPU) Performance data, comprised 209 rows and 10 columns. Each row was used to code 9 features; these could potentially affect the 10th feature, which quantitatively characterized the processor performance. The second set, which contained auto imports data, comprised 206 rows and 26 columns. Each row coded 25 features of an object, which could affect the 26th feature: car price. The third dataset contained Boston Housing data and comprised 506 rows and 14 columns. Each row coded 13 features of objects, which potentially affected the 14th feature.

To solve the regression estimation problem for the above datasets using the ELM, a sigmoid activation function (2) in the hidden layer was used. In the classical realization of ELM, each of the datasets was divided 10 times into learning dataset X_L, which contained 70% of the total number of objects in the set X, and test dataset X_T, which contained 30% of objects of the set X. The classical realization of ELM was trained 10 times on the set X_L by formula (1) and estimated 10 times on the set X_T by formula (4) in order to select the optimal number of neurons in the hidden layer. Figure 3 illustrates the process of selection of the number of neurons, while the selected numbers of neurons are given in Table 1.

Table 1. Selected number of hidden-layer neurons in the classical ELM

Dataset	CPU Performance	Auto Imports	Boston Housing
Number of neurons	30	20	60

The number of neurons in the ELM trained by the bioinspired algorithms was selected in a similar fashion. It was shown [16, 19] that compact ELM (in which the number of hidden-level neurons is relatively small) can achieve better generalization performance in the case of using bioinspired algorithms for selection of input weights and shifts. For this reason, the upper bound of the number of hidden-level neurons was set according to Table 1. In a preliminary study in which cross validation was carried out over 10 blocks at various numbers of hidden-level neurons in the ELM, with the input weights and the hidden-level shifts being distributed by the bioinspired algorithms, it was determined that models with the best generalization performance can be obtained using ETFSS. Table 2 and Fig. 4 present the results of the selection of the number of neurons for the ELM optimized by ETFSS (ETFSS-ELM).

Table 2. Selected number of neurons in the ELM adjusted by the ETFSS algorithm

Dataset	CPU Performance	Auto Imports	Boston Housing
Number of neurons	10	10	30

As Figs. 3 and 4 show, if the selected number of neurons is too small or too large, the ELM prediction accuracy decreases as a consequence of underfitting or overfitting of the model, respectively. The ETFSS-ELM requires fewer neurons to achieve better accuracy on the test data than the classical ELM.

Figure 5 presents the convergence curves of the GA, PSO, and ETFSS algorithms in the optimization of root-mean-square loss function (3) of the compact ELM on learning datasets at the values of the parameters of the bioinspired algorithms that are given in Table 3. Table 2 presents the numbers of hidden-layer neurons in the compact bioinspired ELM.

Table 3. Values of the parameters of the bioinspired algorithms

Algorithm	Selected parameter values
GA	$P_{\text{crossover}} = 0.9$, $P_{\text{mutation}} = 0.1$
PSO	$v_{\max} = 5$, $c_1 = 0.8$, $c_2 = 0.5$, $w = 0.8$
FSS, ETFSS	$step_{\text{ind,initial}} = 0.7$, $step_{\text{vol,initial}} = 0.7$
All algorithms	300 iterations, 100 population agents

It is evident from the curves in Fig. 5 that the ETFSS algorithm, which is an improved version of the FSS algorithm, can find better solutions to the problem of searching for the optimum of loss function (3) in comparison with the GA, PSO, and FSS algorithms.

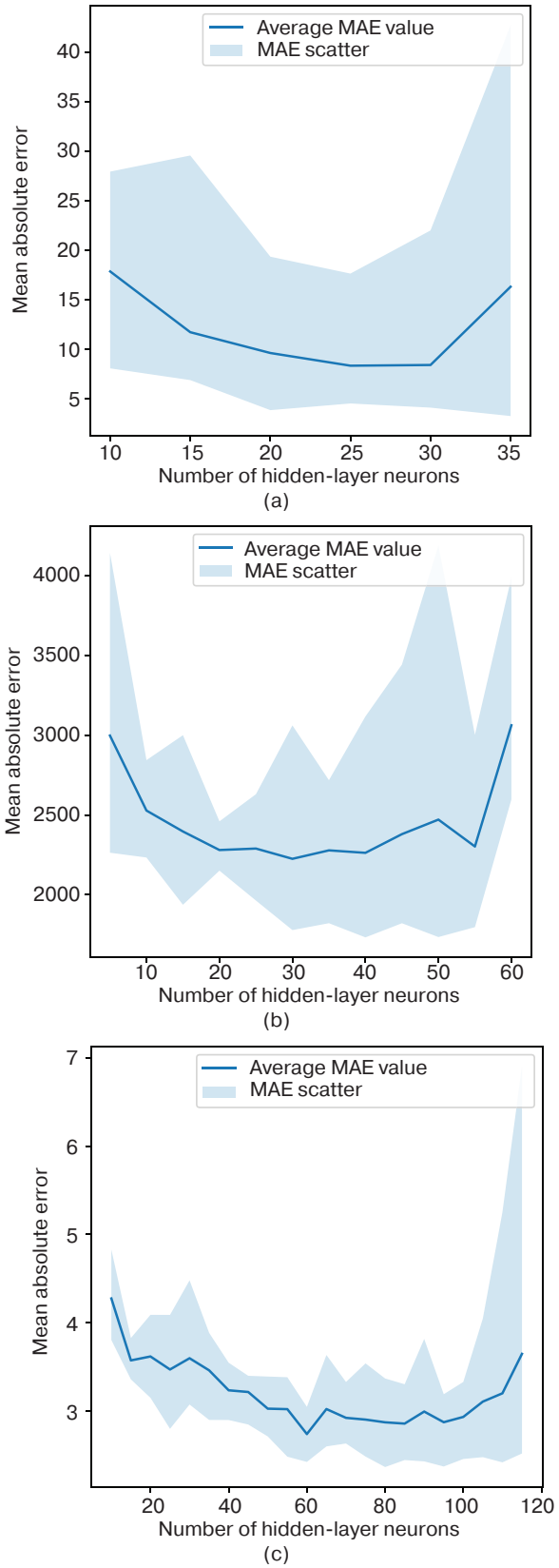


Fig. 3. Selection of the optimal number of neurons in ELM. The shaded areas represent the scatter of the values of function (4) on X_T in 10-block cross-validation; the lines represent the averaged values of function (4) for the (a) CPU Performance, (b) Auto Imports, and (c) Boston Housing datasets

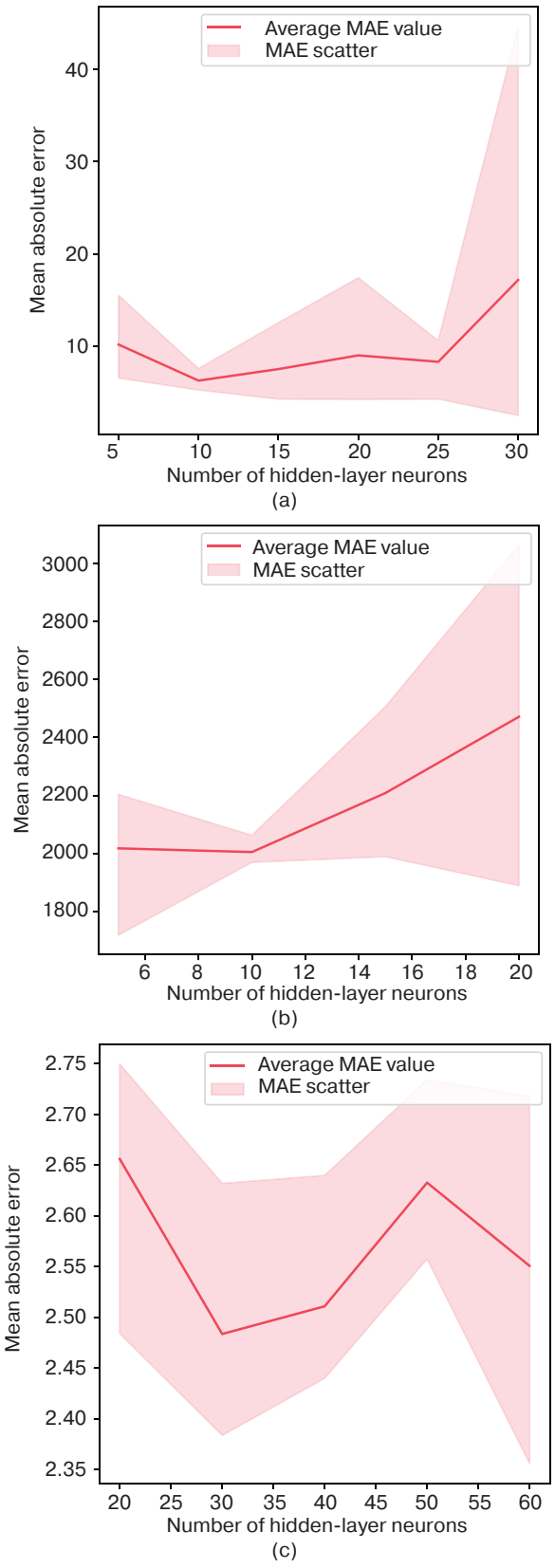


Fig. 4. Selection of the number of neurons in the ELM in which the input weights and shifts were adjusted by the ETFSS algorithm. The shaded areas represent the scatter of the values of function (4) on X_T; the lines represent the averaged values of function (4) for the (a) CPU Performance, (b) Auto Imports, and (c) Boston Housing datasets

The generalization performances of the models GA-ELM, PSO-ELM, FSS-ELM, and ETFSS-ELM obtained by refining the weights by the algorithms GA, PSO, FSS, and ETFSS, respectively, were estimated using the box-and-whisker plots of the values of mean average error (MAE) function (4) (Fig. 6). Table 4 presents the average MAE values on the test subsets of the considered datasets. The comparison also included the classical realizations of ELM with random initialization of input weights and shifts with the numbers of neurons given in Table 1 and 2.

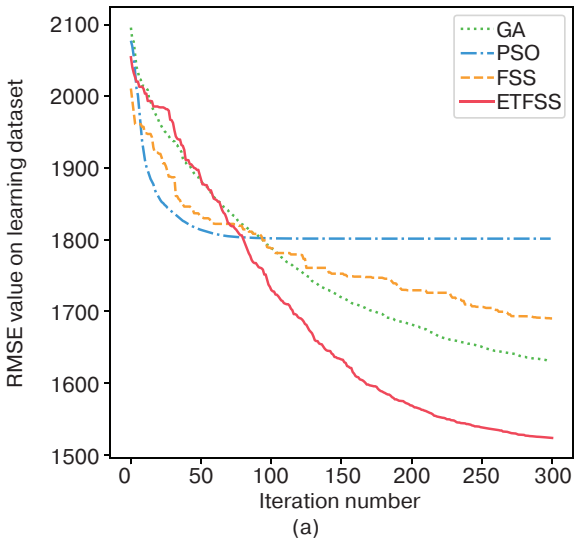
Table 4. Average values of MAE (4) on the test subsets of the considered datasets, which are obtained by 10-block cross validation of ELM-1 with the numbers of hidden-level neurons given in Table 2; ELM-2 with the numbers of hidden-level neurons given in Table 1, GA-ELM, PSO-ELM, FSS-ELM, and ETFSS-ELM

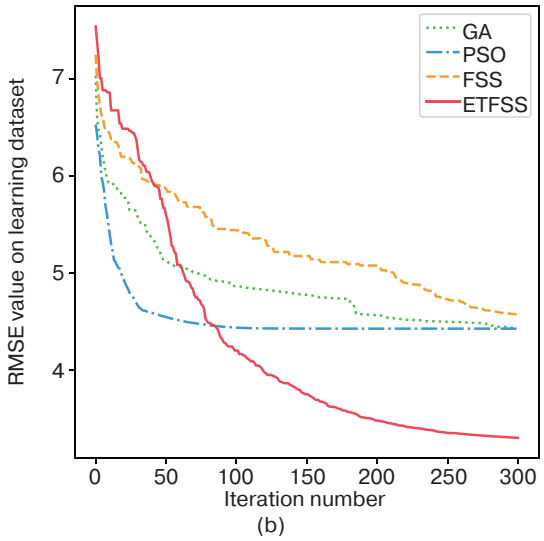
Dataset	ELM-1	ELM-2	GA-ELM	PSO-ELM	FSS-ELM	ETFSS-ELM
CPU Performance	13.2	10.4	9.4	8.8	10.3	**6.2**
Auto Imports	2657	2354	2155	2199	2522	**2124**
Boston Housing	3.47	2.89	2.69	2.86	3.11	**2.64**

Figure 6 and Table 4 show that the ETFSS-ELM model has the best generalization performance among the studied models. GA-ELM, PSO-ELM, FSS-ELM, and ETFSS-ELM, in which intelligent adjustment of weights and shifts was carried out, demonstrate the best generalization performance for all the datasets in comparison with the classical ELM with the same number of hidden-layer neurons. The ELM with the increased numbers of hidden-layer neurons as given in Table 1 is superior to the compact classical ELM and inferior to the compact ELMs that were adjusted by the bioinspired algorithms and have the numbers of hidden-layer neurons as indicated in Tables 2 and 4.

VISUALIZATION OF THE LANDSCAPES OF THE OPTIMIZED LOSS FUNCTION

Landscapes of objective function (3) were visualized to illustrate the process of the search for the optimal values of input weights $\boldsymbol{\alpha}$ of dimension $\mathbb{R}^{d\times n}$, where d is the number of hidden-layer neurons in the ELM, and n is the number of input-layer neurons and shifts of dimension $\mathbb{R}^{d}$. Objective function (3) in this problem of intelligent adjustment of the input weights and shifts in the ELM inputs multidimensional; vector $\vec{w}$ of the form $\mathbb{R}^{q}$, where $q = d \times n + d$. Table 5 presents the q values for each of the considered datasets.





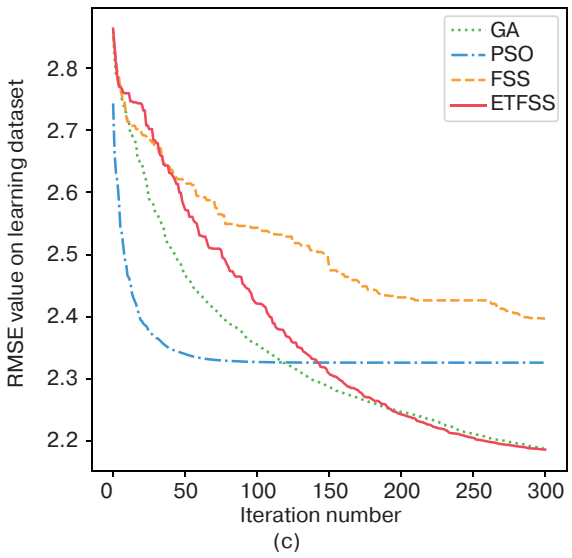
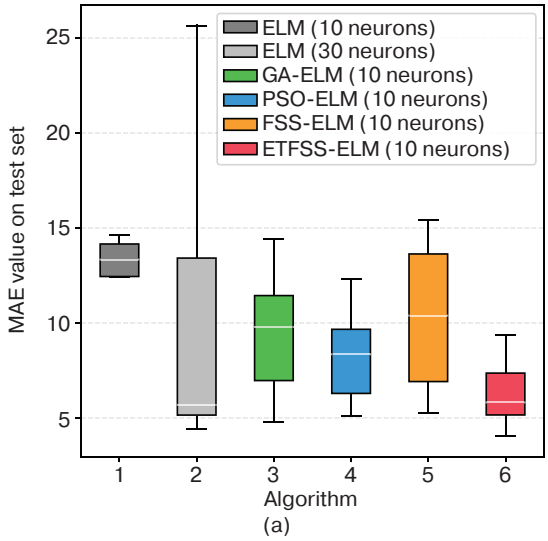
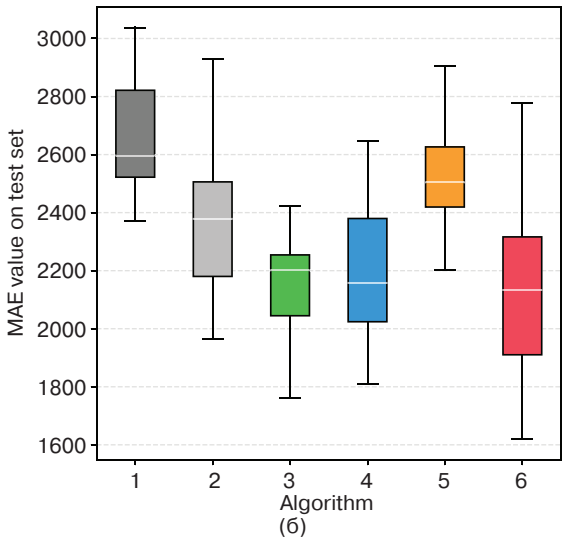


Fig. 5. Convergence of the bioinspired algorithms in the optimization of function (3) to select the input weights and shifts in the ELM for the (a) CPU Performance, (b) Auto Imports, (c) Boston Housing datasets





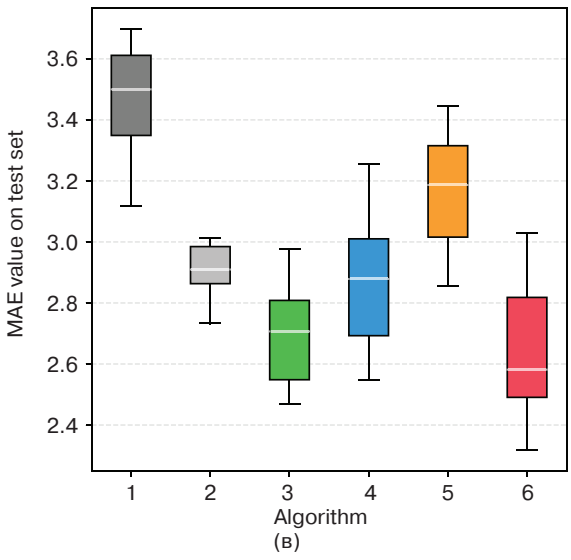


Fig. 6. Box-and-whisker plots of the values of MAE (4) on test data for the classical and bioinspired ELMs with various numbers of hidden-layer neurons for the (a) CPU Performance, (b) Auto Imports, and (c) Boston Housing sets

Table 5. Dimensions of the vector $\vec{w}$ of weights and shifts in the ELM

Dataset	CPU Performance	Auto Imports	Boston Housing
Dimension of w	100	260	420

As Table 5 shows, the dimension of objective function (3) is high enough in each of the analyzed problems, but the landscapes can be visualized only for two-dimensional functions. There are a number of approaches to visualizing multidimensional functions in a three-dimensional rectangular coordinate system. For example, it was proposed [30] to define two orthogonal vectors, $\vec{a}$ and $\vec{b}$, the dimension of which coincides with the dimension of $\vec{w}$, and then define function u to be visualized as

$$u(\alpha,\beta) = f(\vec{w} + \alpha\vec{a} + \beta\vec{b}). \quad (5)$$

Here, f is the initial function, $\vec{w}$ is a multidimensional vector, α and β are scalar parameters, $\vec{a}$ and $\vec{b}$ are orthogonal unit vectors the dimension of which coincides with that of vector

In the problem under consideration, the function f is defined by formula (3); the vector $\vec{w}$ contains the input weights and shifts in the optimal ELM configuration; $\vec{a} = \{0,1,0,1,...\}$; $\vec{b} = \{1,0,1,0,...\}$; and the dimensions of the vectors $\vec{a}, \vec{b}$, and $\vec{w}$ coincide. For each of the considered datasets, the landscapes of function (3) near the found optimum $\vec{w}$ were visualized (Figs. 7, 8). The scalar parameters α and β were varied during the visualization in the range [−1, 1] at an interval of 0.02.

To visualize the process of convergence of the ETFSS algorithm in spaces containing many local extrema shown in Figs. 7 and 8, the changes in the position of a randomly selected agent of the ETFSS algorithm at every 25th iteration were determined. For each position $\vec{p}$ of the agent, the nearest point was chosen in the grid constructed during the landscape visualization by varying the scalars α and β. The proximity of the vector $\vec{p}$ to the point $\{\alpha_i, \beta_i\}$ was determined by calculating the Manhattan distance with a shift with respect to the vector $\vec{w}$, near which the visualization is performed:

$$dist(\vec{p},\alpha_i,\beta_j) = \left|(\vec{p} - \vec{w}) - (\alpha_i\vec{a} + \beta_j\vec{b})\right| = \\ = \sum_{k=1}^{n}\left|(p_k - w_k) - (\alpha_i a_k + \beta_j b_k)\right|. \quad (6)$$

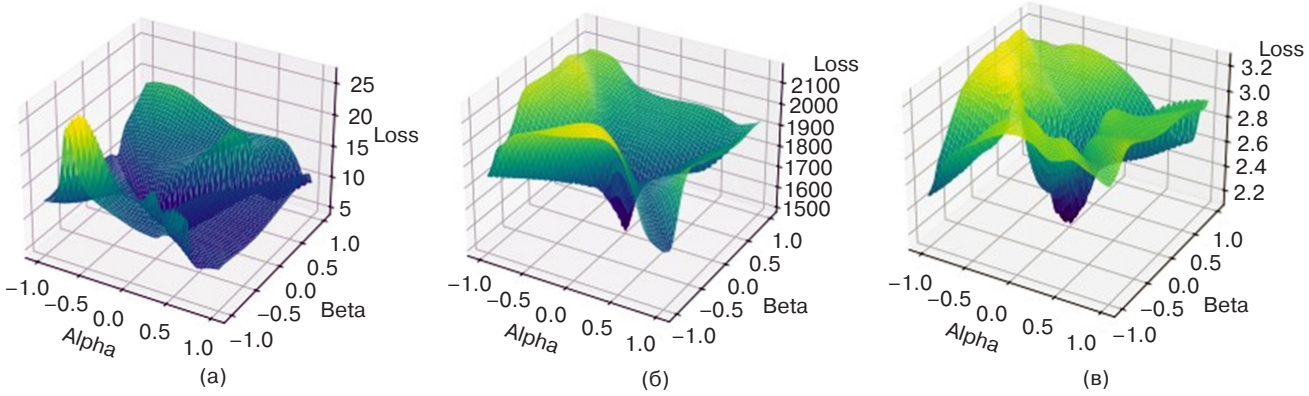


Fig. 7. Visualization of landscapes of multidimensional loss functions near the found optimum for the (a) CPU Performance, (b) Auto Imports, and (c) Boston Housing datasets

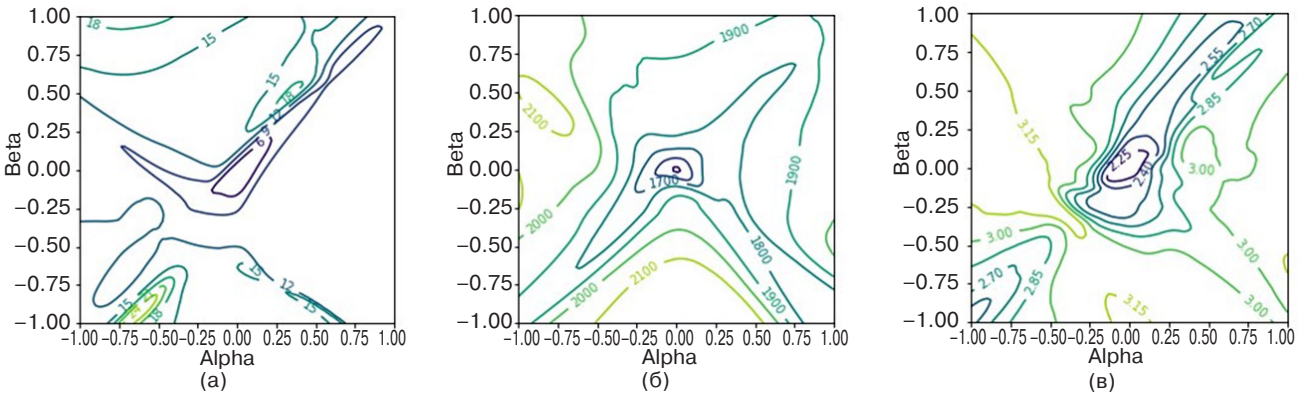


Fig. 8. Visualization of contour lines of multidimensional loss functions near the found optimum for the (a) CPU Performance, (b) Auto Imports, and (c) Boston Housing datasets

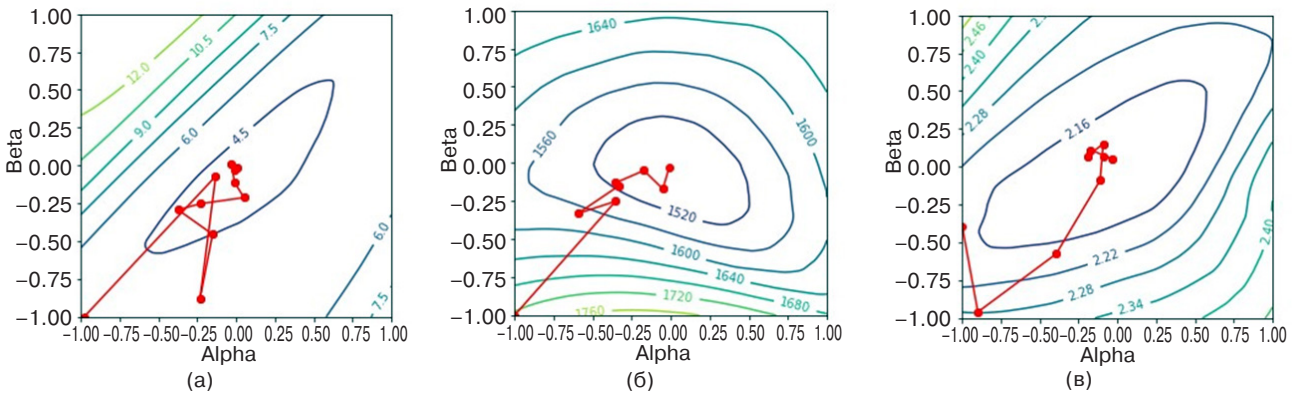


Fig. 9. Changes in the position of a randomly selected agent of the ETFSS algorithm near the found solution for the (a) CPU Performance, (b) Auto Imports, and (c) Boston Housing datasets

Here, n is the dimension of the agent position vector $\vec{p}$; this dimension coincides with that of the vector $\vec{w}$, which codes the input weights and shifts in the found optimal ELM coordination. The dimension of the agent position vector $\vec{p}$ also coincides with that of the mutually orthogonal vectors $\vec{a}$ and $\vec{b}$, $\vec{a} = \{0,1,0,1,...\}$, $\vec{b} = \{1,0,1,0,...\}$, where α_i and β_i are the coordinates of a visualization grid point.

Figure 9 presents the visualizations of the change in the position $\vec{p}$ of the randomly selected agent of the ETFSS algorithm at every 25th iteration in the process of the search for the optimal values of the input weights and shifts in the ELM by optimizing loss function (3) near the found solution .

It can be seen from Figure 7 and 8 that function (3) in the considered problems has a large number of extrema, which are successfully passed by the bioinspired optimization algorithms in the search for the optimal values of the input weights and hidden-layer shifts in the ELM. The trajectories in Fig. 9 suggest that the ETFSS algorithm improves the solutions obtained by the population of agents at each iteration until the completion of the execution of the algorithm.

CONCLUSIONS

In this work, the efficiency of bioinspired algorithms was studied in the problem of intelligent selection of the weights of connections among input-layer neurons and hidden-layer neurons and also the hidden-layer shifts in extreme learning machines in regression estimation problems.

Generalization performances between the classical, compact classical, and compact ELMs were compared, in which the intelligent adjustment of input weights and hidden-layer shifts was made using the genetic algorithm (GA-ELM), the particle swarm optimization (PSO-ELM), the fish school search (FSS-ELM), and the chaotic fish school search with exponential step decay (ETFSS-ELM). It was determined that the compact ELM in which the input weights are adjusted by the bioinspired algorithms achieves a better generalization performance using fewer hidden-layer neurons. The chaotic fish school search with exponential step decay (ETFSS) [24] can ensure the best ELM configurations in the considered problems.

The constructed visualizations of the multidimensional loss function landscapes in the three-dimensional rectangular coordinate system near the found solution demonstrate the presence of numerous extrema, which makes it expedient to use bioinspired algorithms in the search for the global optimum. The obtained visualization of the trajectory of a randomly selected agent of the ETFSS algorithm shows that, iteration by iteration, the algorithm improves the solutions found by the population.

Further studies may be aimed at exploring the possibility of using ETFSS to refine the weights of an online extreme learning machine capable of further learning as new data are received without iterative retraining [31] Analysis of the dependence of the results obtained using bioinspired algorithms on their hyperparameters is also a promising research direction.

Authors' contribution

L.A. Demidova—conceptualization, guidance, supervision, validation, and original draft preparation.

A.V. Gorchakov—software, resources, visualization, testing, and original draft preparation.

All authors have read and agreed to the published version of the manuscript.

REFERENCES

1. Wu Y., Ianakiev K., Govindaraju V. Improved *k*-nearest neighbor classification. *Pattern Recognition*. 2002;35(10):2311–2318. https://doi.org/10.1016/S0031-3203(01)00132-7
2. Noble W.S. What is a support vector machine? *Nat. Biotechnol*. 2006;24(12):1565–1567. https://doi.org/10.1038/nbt1206-1565
3. Demidova L.A. Two-stage hybrid data classifiers based on SVM and kNN algorithms. *Symmetry*. 2021;13(4):615. https://doi.org/10.3390/sym13040615
4. Lin W., Wu Z., Lin L., Wen A., Li J. An ensemble random forest algorithm for insurance Big Data analysis. *IEEE Access*. 2017;5:16568–16575. https://doi.org/10.1109/ACCESS.2017.2738069
5. Deng L., Hinton G., Kingsbury B. New types of deep neural network learning for speech recognition and related applications: An overview. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013:8599–8603. https://doi.org/10.1109/ICASSP.2013.6639344
6. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*. 1958;65(6):386–408. https://doi.org/10.1037/h0042519
7. Affonso C., Debiaso Rossi A.L., Antunes Vieira F.H., Ponce de Leon Ferreira de Carvalho A.C. Deep learning for biological image classification. *Expert Systems with Applications*. 2017;85:114–122. https://doi.org/10.1016/j.eswa.2017.05.039
8. Chen N., Xiong C., Du W., Wang C., Lin X., Chen Z. An improved genetic algorithm coupling a back-propagation neural network model (IGA-BPNN) for water-level predictions. *Water*. 2019;11(9):1795. https://doi.org/10.3390/w11091795
9. Such F.P., Madhavan V., Conti E., Lehman J., Stanley K.O., Clune J. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv preprint arXiv:1712.06567. 2017. https://arxiv.org/abs/1712.06567
10. Shao B., Li M., Zhao Y., Bian G. Nickel price forecast based on the LSTM neural network optimized by the improved PSO algorithm. *Mathematical Problems in Engineering*. 2019;2019(2):1934796. https://doi.org/10.1155/2019/1934796
11. Ruder S. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747. 2016. https://arxiv.org/abs/1609.04747
12. Kulikov A.A. The structure of the local detector of the reprint model of the object in the image. *Russ. Technol. J.* 2021;9(5):7–13. https://doi.org/10.32362/2500-316X-2021-9-5-7-13
13. Dean J., Corrado G.S., Monga R., Chen K., Devin M., Le Q.V., Mao M.Z., Ranzato M.A., Senior A., Tucker P., Yang K., Ng A.Y. Large scale distributed deep networks. *Advances in Neural Information Processing Systems*. 2012;25:1223–1231.
14. Huang G.B., Zhu Q.Y., Siew C.K. Extreme learning machine: theory and applications. *Neurocomputing*. 2006;70(1-3):489–501. https://doi.org/10.1016/j.neucom.2005.12.126
15. Rao C.R. Generalized inverse of a matrix and its applications. In: *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability*. 1972. V. 1. *Theory of Statistics*. 1972:601–620. https://doi.org/10.1525/9780520325883-032
16. Cai W., Yang J., Yu Y., Song Y., Zhou T., Qin J. PSO-ELM: A hybrid learning model for short-term traffic flow forecasting. *IEEE Access*. 2020;8:6505–6514. https://doi.org/10.1109/ACCESS.2019.2963784
17. Liu Y., Loh H.T., Tor S.B. Comparison of extreme learning machine with support vector machine for text classification. In: *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Innovations in Applied Artificial Intelligence*. 2005;3533:390–399. http://doi.org/10.1007/11504894_55
18. Li G.X. Application of extreme learning machine algorithm in the regression fitting. In: *2016 International Conference on Information System and Artificial Intelligence* (*ISAI*). 2016:419–422. https://doi.org/10.1109/ISAI.2016.0095
19. Song S., Wang Y., Lin X., Huang Q. Study on GA-based training algorithm for extreme learning machine. In: *2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics*. *IEEE*. 2015;2:132–135. https://doi.org/10.1109/IHMSC.2015.156
20. Nikonov V.V., Gorchakov A.V. Train machine learning models using modern containerization and cloud infrastructure. *Promyshlennye ASU i kontrollery = Industrial Automated Control Systems and Controllers*. 2021;6:33–43 (in Russ.). https://doi.org/10.25791/asu.6.2021.1288
21. Eremeev A.V. A genetic algorithm with tournament selection as a local search method. *J. Appl. Ind. Math*. 2012;6(3):286–294. https://doi.org/10.1134/S1990478912030039
22. Kennedy J., Eberhart R. Particle swarm optimization. In: *Proceedings of ICNN'95-International Conference on Neural Networks*. 1995;4:1942–1948. https://doi.org/10.1109/ICNN.1995.488968
23. Bastos Filho C.J.A., de Lima Neto F.B., Lins A.J.C.C., Nascimento A.I.S., Lima M.P. A novel search algorithm based on fish school behavior. In: *2008 IEEE Int. Conference on Systems, Man and Cybernetics*. 2008:2646–2651. https://doi.org/10.1109/ICSMC.2008.4811695
24. Demidova L.A., Gorchakov A.V. A study of chaotic maps producing symmetric distributions in the fish school search optimization algorithm with exponential step decay. *Symmetry*. 2020;12(5):784. https://doi.org/10.3390/sym12050784
25. Cao W., Gao J., Ming Zh., Cai Sh. Some tricks in parameter selection for extreme learning machine. *IOP Conf. Ser.: Mater. Sci. Eng*. 2017;261(1):012002. https://doi.org/10.1088/1757-899X/261/1/012002
26. Dos Santos W., Barbosa V., de Souza R., Ribeiro R., Feitosa A., Silva V., Ribeiro D., Covello de Freitas R., Lima M., Soares N. Image reconstruction of electrical impedance tomography using fish school search and differential evolution. In: *Critical Developments and Applications of Swarm Intelligence*. IGI Global; 2018. P. 301–338. https://doi.org/10.4018/978-1-5225-5134-8.ch012

27. Demidova L.A., Gorchakov A.V. Application of chaotic Fish School Search optimization algorithm with exponential step decay in neural network loss function optimization. *Procedia Computer Science*. 2021;186(6):352–359. https://doi.org/10.1016/j.procs.2021.04.156
28. Harrison D. Jr., Rubinfeld D.L. Hedonic housing prices and the demand for clean air. *J. Environ. Econ. Manag*. 1978;5(1):81–102. https://doi.org/10.1016/0095-0696(78)90006-2
29. Kibler D., Aha D.W., Albert M.K. Instance-based prediction of real-valued attributes. *Comput. Intell*. 1989;5(2):51–57. https://doi.org/10.1111/j.1467-8640.1989.tb00315.x
30. Li H., Xu Z., Taylor G., Studer C., Goldstein T. Visualizing the loss landscape of neural nets. In: *NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 2018:6391–6401. https://arxiv.org/abs/1712.09913v3
31. Dai B., Gu C., Zhao E., Zhu K., Cao W., Qin X. Improved online sequential extreme learning machine for identifying crack behavior in concrete dam. *Adv. Struct. Eng*. 2019;22(2):402–412. https://doi.org/10.1177/1369433218788635

About the authors

Liliya A. Demidova, Dr. Sci. (Eng.), Professor, Professor, ERP Systems Department, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: demidova.liliya@gmail.com. Scopus Author ID 56406258800, ResearcherID R-6077-2016. https://orcid.org/0000-0003-4516-3746

Artyom V. Gorchakov, Postgraduate Student, ERP Systems Department, Institute of Information Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: worldbeater-dev@yandex.ru. Scopus Author ID 57215001290, ResearcherID ABC-8911-2021. https://orcid.org/0000-0003-1977-8165

Об авторах

Демидова Лилия Анатольевна, д.т.н., профессор, профессор кафедры корпоративных информационных систем Института информационных технологий ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: demidova.liliya@gmail.com. Scopus Author ID 56406258800, ResearcherID R-6077-2016. https://orcid.org/0000-0003-4516-3746

Горчаков Артём Владимирович, аспирант кафедры корпоративных информационных систем Института информационных технологий ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: worldbeater-dev@yandex.ru. Scopus Author ID 57215001290, ResearcherID ABC-8911-2021. https://orcid.org/0000-0003-1977-8165

Translated by V. Glyanchenko
Edited for English language and spelling by Dr. David Mossop