Check for updates

Research Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
July-August 2021: 1–11
© The Author(s) 2021
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/1729881420925283
journals.sagepub.com/home/arx
SAGE

Utilization of multilayer perceptron for determining the inverse kinematics of an industrial robotic manipulator

Sandi Baressi Šegota[1], Nikola Anđelić[1], Vedran Mrzljak[1], Ivan Lorencin[1], Ivan Kuric[2] and Zlatan Car[1]

Abstract

Inverse kinematic equations allow the determination of the joint angles necessary for the robotic manipulator to place a tool into a predefined position. Determining this equation is vital but a complex work. In this article, an artificial neural network, more specifically, a feed-forward type, multilayer perceptron (MLP), is trained, so that it could be used to calculate the inverse kinematics for a robotic manipulator. First, direct kinematics of a robotic manipulator are determined using Denavit–Hartenberg method and a dataset of 15,000 points is generated using the calculated homogenous transformation matrices. Following that, multiple MLPs are trained with 10,240 different hyperparameter combinations to find the best. Each trained MLP is evaluated using the R^2 and mean absolute error metrics and the architectures of the MLPs that achieved the best results are presented. Results show a successful regression for the first five joints (percentage error being less than 0.1%) but a comparatively poor regression for the final joint due to the configuration of the robotic manipulator.

Keywords

Artificial intelligence, artificial neural network, inverse kinematics, machine learning, multilayer perceptron, robotic manipulator

Date received: 11 August 2020; accepted: 14 June 2021

Topic Area: Robot Manipulation and Control
Topic Editor: Andrey Savkin
Associate Editor: Martin Grossman

Introduction

Determining kinematic properties of a robotic manipulator is a crucial step in any work relating to the use of the robotic manipulator. Before any further calculations can be performed, direct and inverse kinematic equations need to be determined. Direct kinematic equations allow the transformation from the joint variable space into the tool configuration space, that is, calculation of the position of tool in workspace from predefined joint rotation values.[1] Inverse kinematic equations allow the opposite transformation from the tool configuration space to the joint variable space, that is, if we know the position in the workspace that we are trying to achieve, we can calculate the joint values necessary to position the tool at that location. While the determination of the robotic manipulator direct kinematics is relatively straightforward, and there are methods such as Denavit–Hartenberg (D-H) that allow for the simple determination, determining inverse kinematics is a more

[1] Department of Engineering, University of Rijeka, Rijeka, Croatia
[2] Department of Mechanical Engineering, University of Žilina, Žilina, Slovakia

Corresponding author:
Zlatan Car, Department of Engineering, University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia.
Email: zlatan.car@riteh.hr

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).

complex process.[2] Determining the inverse kinematic equations for complex robots has high algebraic complexity.[3] Determining the solution numerically is an option, but it takes a comparatively long time compared to using a direct solution, such as an equation.

In this article, a method is proposed in which multilayer perceptron (MLP) is used to regress the equations for the inverse kinematic equations for each joint. ANN is an artificial intelligence or machine learning algorithm[4] that can be used for solving various regression and classification tasks.[5,6] Artificial intelligence methods such as this have a wide variety of applications. ANNs and other artificial intelligence methods, such as evolutionary computing algorithms,[7–9] have a wide variety of uses in robotics in areas such as computer vision[10,11] and path optimization[12–14] and navigation.[15,16] ANN emulates the human neural system using a structure of nodes, referred to as neurons, that are interconnected with weighted connections.[17,18] By adjusting this weight depending on the data, ANNs can provide the ability to classify or regress the input data to a solution with high precision.[18,19] Robotic manipulator used is modeled after a realistic six degrees of freedom (6-DOF) manipulator ABB IRB 120.

State-of-the-art

Villegas et al.[20] use the dataset adaptation technique to solve the inverse kinematic issue, by ANN retraining using data points with higher error values. This approach shows promising results in rising the accuracy of the trained ANN. Nemeth et al.[21] show the use of machine learning methods, such as classification trees and clustering, in the process of detecting failures in production systems. The authors show a successful detection of failures using these methods, using data collected from control program's logged files. Ghafil et al.[22] attempt to solve inverse kinematics of 3-DOF robotic manipulator using Levenber–Marquardt, Bayesian regularization, and scaled conjugate gradient learning algorithms. Results show that the best results are provided when using the Bayesian regularization algorithm. Demby et al.[23] show the use of a reinforcement learning method, specifically ANNs and adaptive neurofuzzy inference systems. While their methods show some ability to regress, the authors conclude that the results are not accurate enough for use. Ghasemi et al.[24] show the use of an ANN for the determination of inverse kinematics on a simple 3-DOF RRL robotic manipulator. The proposed solution shows that ANNs have the ability to be used for kinematic determination for a simpler robotic manipulator. Dash et al.[25] attempt to regress the inverse kinematic equations using the ANN. The authors conclude that the results achieved are not satisfactory, as they have a very low accuracy especially for the second and sixth joint of the manipulator, most probably caused due to a low number of training data and no hyperparameter variation. Zarrin et al.[26] show the application of the ANN for the direct control of a soft robotic system. This article shows a successful application of the ANN in use with robotic manipulators. Takatani et al.[27] use a neural network to achieve determination of joint values. The authors propose the use of a complex, multi-ANN model to learn the inverse kinematics without the use of evaluation functions. The results show that, for a 3-DOF, robotic manipulator solution like the proposed one can provide satisfactory results. Dereli and Köker[28] propose a metaheuristic artificial intelligence solution for inverse kinematic calculation. Multiple algorithms are compared by authors (quantum behaved particle swarm, firefly algorithm, particle swarm optimization, and artificial bee colony) and best results are shown when using quantum behaved particle swarm. Lim and Lee[29] discuss the number of points necessary for the inverse kinematic solutions. Their findings show that regression can be learned with as little as 125 experimentally obtained data points.

Hypotheses

This article is trying to prove three hypotheses and these are as follows:

- a MLP ANN can be used to determine inverse kinematics of a realistic 6-DOF manipulator instead of a more complicated method,
- determine the best possible configuration of the MLP for determining inverse kinematics of each joint, and
- a dataset virtually generated from direct kinematics can be used for ANN training.

The novelty of this article is twofold. First, it lays in the fact that an artificially generated dataset has been used to train the neural network. Using an artificially generated dataset, training of inverse kinematics networks for a new robotic manipulator can be performed significantly faster than when an experimental dataset is used (provided that the results are satisfactory). This would allow for training of the neural networks in such a case where the robotic manipulator is not readily available to the researchers to perform a large number of experimental measurements upon. The second part lays in the fact that only the x, y, and z spatial positions of the end effector are used as inputs without the orientations of the end effector. In case that satisfactory result can be achieved this way, the method would show the ability to regress the inverse kinematics problem without the need for measurement of the Euler angles, which define the orientation, simplifying training of such neural networks.

In continuation, first, the methodology will be presented. The method used for determining direct kinematics and the dataset generation will be explained. After that, a description of MLP ANNs is given, along with the methods for hyperparameter determination and solution quality

estimation. Finally, results are presented and discussed along with the drawn conclusions.

Methodology

In this article, the methodology used in our research will be presented. First, the process of determining the direct kinematic equations of the robotic manipulator will be described, followed by the description of a way those equations were used to generate a used dataset. The overview of the ANN will be given, along with the description of chosen activation functions and hyperparameters used during the grid search process. Finally, the way authors evaluated the quality of the obtained solutions will be presented.

Direct kinematic equation determination

Direct kinematic equations are determined using the D-H algorithm. In the D-H algorithm, each joint is assigned a sequential number i from 1 to n, where 1 depicts the first joint of the robotic manipulator nearest to the base and n depicts the last joint of the robotic manipulator.[30] Then, the recursive D-H algorithm is started. In the first iterative part, each joint is assigned a right orthonormal coordinate system L_k, where $k \in [0, n$ and $k = i - 1$ for each joint.[31] At the start of the algorithm, the base joint of the robotic manipulator is assigned the coordinate system L_0 with the axes x^0, y^0, and z^0. For the remaining joints, the axes of the coordinate system for each joint are placed in such a way that axis z^k equates the joint axis of the associated joint. Axis x^k is placed in the cross-section of the axes z^k and z^{k-1}. If the axes z^k and z^{k-1} are not perpendicular, the axis x^k is placed in such a way that it is perpendicular to the axes z^k and z^{k-1}. The remaining axis y^k for each joint k is placed in such a way that the right oriented orthonormal coordinate system is formed.[32,33] Once all the coordinate systems have been formed, points b_k for each joint k are placed. Point b_k for each joint is placed at the intersection of axes x_k and z^{k+1}. If those axes are not perpendicular, point b_k is placed at the intersection of axis x^k and axis perpendicular to both axes x_k and z^{k+1}.[34] With the coordinate system L and points b placed, the kinematic parameters can be determined. There are four kinematic parameters that need to be determined for each joint k: d_k, a_k, α_k, Θ_k.[32,35] These parameters represent

- θ_k: the joint angle defined as the angle of rotation around axis z^{k-1} that makes axes x^{k-1} and x^k parallel;
- α_k: the angle of rotation around axes x^k that makes axes z^{k-1} and z^k parallel;
- d_k: translation along axis z^{k-1} needed to achieve intersection of axes x^{k-1} and x^k; and
- x_k: translation along axis x^{k-1} needed to achieve intersection of axes z^{k-1} and z^k.

For a rotational joint, θ_k is treated as a variable, and for a translational joint, d_k is treated as a variable.[36] Values for each joint are placed in the matrix

$$T_{k-1}^{k} = \begin{bmatrix} C_{\Theta_k} & -C_{\alpha_k}S_{\Theta_k} & S_{\alpha_k}S_{\Theta_k} & a_kC_{\Theta_k} \\ S_{\Theta_k} & C_{\alpha_k}C_{\Theta_k} & -S_{\alpha_k}C_{\Theta_k} & a_kS_{\Theta_k} \\ 0 & S_{\alpha_k} & C_{\alpha_k} & d_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Because of the size of the equations, the shortened trigonometric format is used. When using this format, the trigonometric functions are written using only the first letter of their name, so sine is written as S and cosine as C. In addition, the arguments of the functions are written as indexes. The kinematics matrix for the entire robotic manipulator, connecting the base of the robotic manipulator and the tool, is calculated as a product of the homogenous transformation matrices T_{k-1}^{k} of each joint. The resultant matrix $T_{\text{base}}^{\text{tool}}$ is formatted as[35,37]

$$T_{\text{base}}^{\text{tool}}(q) = \begin{bmatrix} R(q) & p(q) \\ v_1^T & \sigma \end{bmatrix} \quad (2)$$

in which v_1^T represents the perspective vector, with the usual value of [0 0 0], σ represents the scaling coefficient with the usual value of 1. The vector $p(q)$ (3×1) represents the final tool position in the workspace.[34] Matrix $R(q) = [r^1\ r^2\ r^3]$ (3×3) represents the tool orientation matrix, where[33,38]

- r^1 is the perpendicular vector,
- r^2 is the movement vector, and
- r^3 is the approach vector.

Robotic manipulator tool positioning can be defined using

$$w = \begin{bmatrix} w^1 \\ w^2 \end{bmatrix} = \begin{bmatrix} [x\ y\ z]^T & T \end{bmatrix} \quad (3)$$

with x, y, and z being robotic manipulator end-effector positions in the tool configuration space, and Φ, Θ, and Ψ being Euler angles (spin, nutation, and precession).[33,39] Vectors w^1 and w^2 can be defined using the homogenous transformation matrix of the robotic manipulator

$$w^1 = p(q) \quad (4)$$

and

$$w^2 = r^3 \quad (5)$$

where $p(q)$ and r^3 are members of the homogenous transformation matrix $T_{\text{base}}^{\text{tool}}(q)$, as described in equation (3).

Dataset generation

To generate the dataset, first, the possible joint values need to be defined. Figure 1 shows the robotic manipulator in

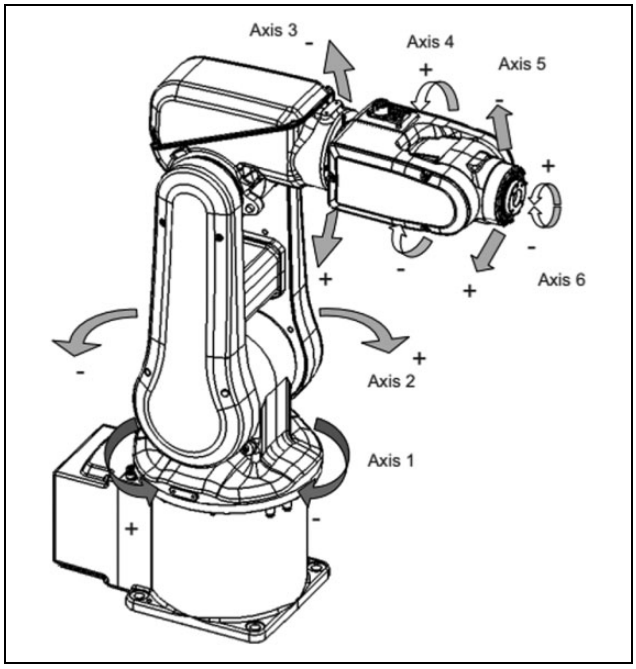

Figure 1. Diametric view of the robotic manipulator with axes displayed[40] (Axes 1–6: Rotational axes of the robotic manipulator).

Table 1. The upper and lower limits of each joint of the robotic manipulator for ABB IRB 120 robotic manipulator, obtained from the manufacturer's product specification.[40]

Joint	Lower limit (°)	Upper limit (°)	Range (°)
1	−165	165	330
2	−110	110	220
3	−70	110	180
4	−160	160	320
5	−120	120	240
6	0	360	360

diametric view with joint axes displayed. The limits of joint angle values for each joint are given in Table 1. On the figure, axes 1 through 6 represent the rotational axis of the corresponding joint of the robotic manipulator, with the arrows representing the directions in which the given joint is capable of rotating.

A random value is generated for each joint within its given range. These values are generated with a uniform random distribution. Then, the direct kinematic equations are used to calculate the x, y, and z coordinates of the robotic manipulator end effector. Doing this yields a dataset of 15,000 data points, which is a large enough amount to attempt a regression analysis using artificial intelligence techniques.[29] Furthermore, analysis of the values in the dataset shows that all of the input and output values are unique, allowing for unique mapping from the inputs to outputs. While this dataset could have been generated experimentally, through robot positioning and measurement of joint angles, this would have taken an extremely long time and would have added the problem of measurement imprecision and errors. Generating dataset in the manner described in this article was chosen by the authors because of the wish to test the possibility of using an artificially generated dataset for ANN training.

Multilayer perceptron regressor

MLP is a type of a feed-forward ANN, consisting of an input layer, output layer, and at least one hidden layer,[41] consisting of one or more artificial neurons.[42] The value of each neuron starting with inputs is propagated to the neurons in the next layer over connections. Each connection has a certain weight, which signifies the importance of the value of that neuron to the value of the neuron it is connected to. The sum of values of all neurons of the previous layer connected to the current neuron is then transformed using the activation function of the neuron in question and the new value of that neuron is calculated.[43] The value of the input neurons is weighted, summed, and transformed over each neuron in the first hidden layer, and this is repeated for each following layer until the final, output, layer is reached and the values of neurons are weighted and summed giving the output value.[44,45]

From the above, it can be seen that the weights of the connections play an important part in value determined as the output. Because of this property, it is important to set connection weights as accurately as possible, which is done in the training stage during which the data stored in the dataset are used to adjust the connection weights.[43,45]

Each of the inputs x_i has a corresponding weight θ_n stored in vector $\Theta_0 = [\theta_1 \quad \theta_2 \quad \theta_3 \cdots \theta_n]^T$.[46] Each layer in neural network has its own weight vector Θ in which the number of elements represents the number of connections from that layer to the following one.[42] The input values of the neural network can be defined as[43]

$$z = X \cdot \Theta_0 \tag{6}$$

where X is the vector of inputs described in equation (7) and Θ_0 is the set of weights for the connections connecting the input layer of the neural network to the first hidden layer.

The output value of the neurons is calculated as

$$f(X_k) = \mathcal{F}(X_k \cdot \Theta) = \mathcal{F}(x_1 \cdot \theta_1 + x_2 \cdot \theta_2 + \cdots + x_n \cdot \theta_n) \tag{7}$$

where $\mathcal{F}$ is the activation function of neuron and k is the number of layers in ANN, with $k = 0$ for input layer.[43,47] In other words, neuron output value is the value of its activation function for the weighted sum of its inputs. The value of each neuron is calculated this way, with appropriate weights, until the output layer is reached and its value is calculated.[44] Initially, the weights of the neurons are set to randomly selected values.[44,45] These randomly selected values, in most cases, do not provide satisfactory results and need to be adjusted.[43,48]

The error of the neural network is propagated back through the network (meaning, in the direction from the output neuron to the input neurons), with the goal of adjusting the connection weights.[49,50] Gradients are used to allow the adjustment of values depending on their difference from the goal value, where error $\mathcal{E} = 0$.[51] The connections that have a higher difference from the goal value will have a higher adjustment value compared to those that are close to the goal. Once the final gradient Θ' (at inputs) is calculated, the gradient is updated using

$$\Theta_{\text{new}} = \Theta_{\text{old}} - \frac{\alpha}{m} \cdot \frac{\partial \frac{\sum_{i=0}^{n} (y_i - \hat{y}_i)^2}{2n}}{\partial \Theta} \tag{8}$$

where α represents the learning rate of the ANN[52,53] and m is a total number of data points. The higher this rate is, the faster the weights Θ will get adjusted,[54] but a too large learning rate can cause issues with the backpropagation diverging instead of converging toward $\mathcal{E} = 0$.[55,56]

In addition to value of neurons, there is another value taken into account in each layer. This value is referred to as bias and marked with b_i. Bias is the value that allows the adjustment of the origin point of the activation function.[43,57] This enables neural networks to be adjusted in a way that allows solving problems, which have the solutions near the origin point of the coordinate system of the solution space.[58–60]

Hyperparameter determination

In the previous sections, a number of values have been mentioned, such as learning rate α, hidden layers, and the number of neurons in each as well as the activation functions. These values are parameters that describe the neural network itself, or more precisely, its architecture. To differentiate these parameters from parameters contained within the neural network, namely the values of weights Θ, these parameters are named hyperparameters.[61,62]

Values of hyperparameters determine how well the ANN will perform the task it was designed to. The hyperparameters of the neural network varied in this article are hidden layers (number of layers and neurons per each layer[63]), activation function of the neurons, solver, initial learning rate, type of learning rate adjustment, and regularization parameter L2.[51,64,65] The values of hyperparameters used in the research are given in Table 2.

The total number of combinations, that is, ANN architectures, can be calculated as a product of numbers of possible values for each hyperparameter, which means that the total of ANN architectures tested is 10,240.

The hyperparameters are tested using the grid search method.[67] In grid search method, the combination of each of the above parameters is given, and a neural network with those hyperparameters is calculated.[68,69] Additionally, each combination of parameters is executed 10 times due to the cross-validation process described in the following section.

Table 2. Possible values for each hyperparameter used in grid search for determining the best ANN architecture.[a]

Hyperparameter	Possible values	Number
Hidden layers	(3), (3,3,3,3), (3,3),(4,3,3,4), (6,6,6,6), (6,6), (7,6,6,7),(6,4,6), (6,12,12,6), (10,10,10,10,10), (12,12,12), (10,20,20,10), (4,4), (4), (6), (100), (100,100,100), (100,100,100,100), (30), (8,6,8)	20
Activation function	Identity, ReLU, Tan-H, Logistic	4
Solver	Adam, LBFGS	2
α	0.5, 0.1, 0.01, 0.00001	4
α Adjustment	constant, adaptive, inverse scaling	4
L2	0.1, 0.001, 0.0001, 0.01	4

[a]First column gives hyperparameter values of which are defined in the second column. Third column provides a total number of possible parameter values. For the "hidden layers" row, each tuple represents an ANN setup, where each hidden layer is represented by an integer equaling the number of neurons it contains.[66]

Solution quality estimation

After the training using the parameters obtained using an iteration of grid search, the other part of the dataset is used as the testing set. In testing, the trained MLP attempts to determine the value of the set inputs, which are compared to the real outputs. The difference between this process and training is that no adjustment is done in this part—there is no backpropagation, just the forward. This enables the quality of the neural network to be determined, as the values it predicts are compared to the values that have not been used in training and as such did not have an effect on the trained weights of the neural network. Once the predicted set $\hat{y}^{\text{train}}$ is calculated, it can be compared to the actual result values of the training set y^{train}.[43]

Two values are used to determine the quality of the solution: coefficient of determination (R^2) and mean absolute error (MAE).

The coefficient of determination is defined as the proportion of variance in the dependent variable predictable from the independent variables.[70,71] This provides information on how much of the variance in the data is explained by the predicted data.[72,73] Let vector $y = [y_1 y_2 \cdots y_m]$ contain the real data points y_i, and vector $\hat{y} = [\hat{y}_1 \hat{y}_2 \cdots \hat{y}_m]$ contain the predicted data, where each prediction $\hat{y}_i$ corresponds to the real data y_i for each i.[73–75]

With these two values, the coefficient of determination R^2 is defined by[76–78]

$$R^2 = 1 - \frac{\sum_i^m (y_i - \hat{y}_i)^2}{\sum_i^m (y_i - \bar{y})^2} \tag{9}$$

MAE is defined as[79]

$$\text{MAE} = \frac{1}{n} \sum_i^n |y_i - \hat{y}_i| \tag{10}$$

and it provides the information of what is the average difference between real values in vector y and corresponding predicted values in vector $\hat{y}$.

R^2 is defined in range 0, 1, with values closer to 1 representing better quality solution,[80] that is, solution that predicts values, which have less unexplained variance.[81] Solutions with an R^2 value of 0.9 or higher are taken in consideration where available. In ideal solution, that is, a perfect regressor, MAE would equal 0.[82,83] With this in mind, solutions with lower MAE are taken as better during evaluation.

To assure the stability of the ANN models across various sets of data, cross validation has been applied. Tenfold cross validation has been used. This process splits the dataset into 10 parts (folds) randomly. Then, the training is repeated 10 times. In each of these iterations, one of these folds is used as a testing set, while the other nine are used as a training set. This allows for the entirety of the dataset to be used for testing, determining the quality of the achieved solution across the entirety of the dataset.

Results

This section represents the results obtained by the authors during the research. First, the kinematic parameters and direct kinematic equations obtained are presented, followed by the results obtained from ANN training.

Direct kinematic properties

Performing D-H method on the ABB IRB 120 robotic manipulator yields the schematic shown in Figure 2. On the schematic, each joint of the robotic manipulator is represented with a dotted circle. L_0 through L_5 represent the origins of coordinate system of each joint, while x^0 through x^5, y^0 through y^5, and z^0 through z^5 represent the axes of each coordinate system. With L_6 being the end-effector's coordinate system, and x^6, y^6, and z^6 being the normal, sliding and approach and vectors, respectively. Finally, symbols b_1 through b_6 represent the intersections of the current coordinate systems x axis and the following coordinate systems z axis.

For the observed robotic manipulator, the values of kinematic parameters are presented in Table 3. By inserting values from Table 3 into the homogenous transformation, matrices of each joint using equation (1) matrices shown in equations (24) to (29) are obtained.

$$T_0^1 = \begin{bmatrix} \cos(q_1) & 0 & -\sin(q_1) & 0 \\ \sin(q_1) & 0 & \cos(q_1) & 0 \\ 0 & -1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11)$$

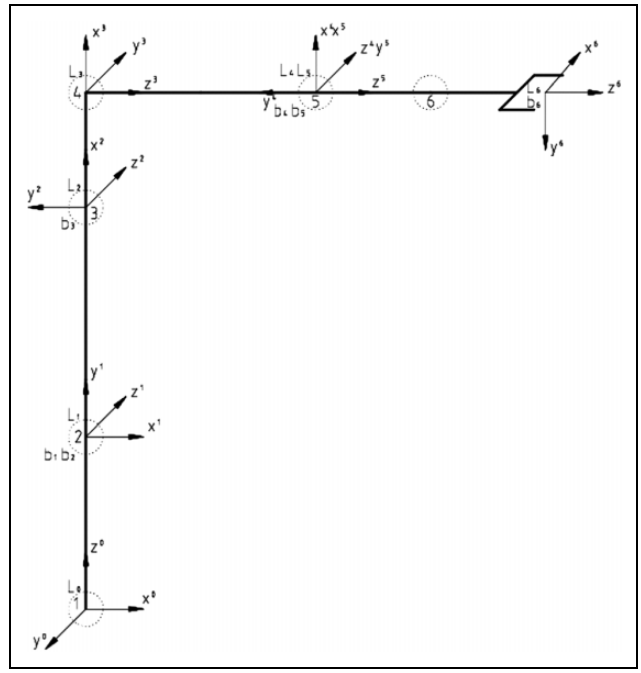

Figure 2. Simplified schematic of the modeled robotic manipulator with coordinate systems determined using D-H method (Dotted circle: manipulator joint; L0–L6: coordinate system origins; x0–x6, y0–y6, z0–z6: coordinate system axes; b1–b6: *x* and *z* axes intersection points). D-H: Denavit–Hartenberg.

Table 3. Kinematic parameters of the robotic manipulator determined using the D-H method.

k	1	2	3	4	5	6
θ_k	q_1	q_2	q_3	q_4	q_5	q_6
a_k (m)	0	l_2	l_3	0	0	0
d_k (m)	l_1	0	0	l_4	0	l_6
α_k (rad)	$-\frac{\pi}{2}$	0	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	$-\frac{\pi}{2}$	0

$$T_1^2 = \begin{bmatrix} \cos(q_2) & -\sin(q_1) & 0 & l_2 * \cos(q_2) \\ \sin(q_2) & \cos(q_1) & 0 & l_2 * \sin(q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (12)$$

$$T_2^3 = \begin{bmatrix} \cos(q_3) & 0 & -\sin(q_3) & l_3 \cos(q_3) \\ \sin(q_3) & 0 & \cos(q_3) & l_3 * \sin(q_3) \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

$$T_3^4 = \begin{bmatrix} \cos(q_4) & 0 & -\sin(q_4) & 0 \\ \sin(q_4) & 0 & \cos(q_4) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (14)$$

$$T_4^5 = \begin{bmatrix} \cos(q_5) & 0 & -\sin(q_5) & 0 \\ \sin(q_5) & 0 & \cos(q_5) & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

and

$$T_5^6 = \begin{bmatrix} \cos(q_6) & -\sin(q_6) & 0 & 0 \\ \sin(q_6) & \cos(q_6) & 0 & 0 \\ 0 & 0 & 1 & 0.147 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}$$

Multiplication of this matrix per equation (2) yields the homogenous transformation matrix of the robotic manipulator as follows

$$T_0^6 = \begin{bmatrix} T_{00} & T_{01} & T_{02} & T_{04} \\ T_{10} & T_{11} & T_{12} & T_{14} \\ T_{20} & T_{21} & T_{22} & T_{24} \\ T_{30} & T_{31} & T_{32} & T_{34} \end{bmatrix} \tag{17}$$

where

$$T_{00} = ((S_1S_4 + C_1C_4C_{23})C_5 - S_5S_{23}C_1)C_6 + (S_1C_4 - S_4C_1C_{23})S_6 \tag{18}$$

$$T_{10} = ((S_1C_4C_{23} - S_4C_1)C_5 - S_1S_5S_{23})C_6 - (S_1S_4C_{23} + C_1C_4)S_6 \tag{19}$$

$$T_{20} = -(S_5C_{23} + S_{23}C_4C_5)C_6 + S_4S_6S_{23} \tag{20}$$

$$T_{30} = 0 \tag{21}$$

$$T_{01} = (-(S_1S_4 + C_1C_4C_{23})C_5 + S_5S_{23}C_1)S_6 + (S_1C_4 - S_4C_1C_{23})C_6 \tag{22}$$

$$T_{11} = (-S_1C_4C_{23} + S_4C_1)C_5 + S_1S_5S_{23})S_6 - (S_1S_4C_{23} + C_1C_4)C_6 \tag{23}$$

$$T_{21} = (S_5C_{23} + S_{23}C_4C_5)S_6 + S_4S_{23}C_6 \tag{24}$$

$$T_{31} = 0 \tag{25}$$

$$T_{02} = -(S_1S_4 + C_1C_4C_{23})S_5 - S_{23}C_1C_5 \tag{26}$$

$$T_{12} = (-S_1C_4C_{23} + S_4C_1)S_5 - S_1S_{23}C_5 \tag{27}$$

$$T_{22} = S_5S_{23}C_4 - C_5C_{23} \tag{28}$$

$$T_{32} = 0 \tag{29}$$

$$T_{03} = -l_4S_1S_4S_5 - l_4S_5C_1C_4C_{23} - l_4S_{23}C_1C_5 - l_1S_{23}C_1 + l_3C_1C_2 + l_5C_1C_{23} \tag{30}$$

$$T_{13} = -l_4S_1S_5C_4C_{23} - l_4S_1S_{23}C_5 - l_1S_1S_{23} + l_3S_1C_2 + l_5S_1C_{23} + l_4S_4S_5C_1 \tag{31}$$

$$T_{23} = -l_3S_2 + l_4S_5S_{23}C_4 - l_5S_{23} - l_4C_5C_{23} - l_1C_{23} + l_2 \tag{32}$$

and

$$T_{33} = 1 \tag{33}$$

Equations (31) to (46) represent each of the elements of the matrix T_0^6, defined in equation (30). Each of the elements l_1 through l_6 represent a length of a given link of the robotic manipulator, starting from the base to the end effector. Elements S and C represent a shortened form of trigonometric functions sine and cosine. The indexes of those elements represent angle of the joints the trigonometric function was used on. For example, $\sin(q_1)$ becomes S_1, or $\cos(q_2)$ becomes C_2. When more than one number is indexed, it represents a sum of those angles. For example, $\cos(q_1 + q_2 + q_3)$ becomes C_{123}.

Multilayer perceptron regression

The results for the most successful models for each joint are selected. As mentioned, models R^2 and MAE performance metrics on the training set are observed as the determination of quality. The results are given in Table 5. Where possible, a simpler architecture with comparable results was chosen due to a simpler architecture having better training times. Shorter training times are apparent in the simpler neural networks (where simpler means a lower number of neurons) due to the lower number of connection gradients and weights that need to be calculated during the backpropagation stage in training. The ratio of MAE and the total range for the joint as given in Table 1 are also presented in Table 4.

Discussion

The obtained results demonstrate a successful regression of the inverse kinematics problem. All the solutions achieve the MAE, which is smaller than 1% of the possible joint angle range. Only model which has an error larger than 1° or 0.1% of the joint angle range is the q_6. The larger error of the q_6 can be explained by the fact that the dataset does not contain the tool orientation data, which has a large influence on the joint in question. From Figures 1 and 2, it can be seen that joint 6 is the final torsion joint of the robotic manipulator, meaning its direct influence is not exhibited on the X, Y, and Z positioning of the end-effector position. R^2 scores achieved by the networks are above 0.9 in all

Table 4. Percentage error for each joint in regards to its maximum movement range.

Joint	$\overline{\text{MAE}}$ (°)	σ	$\frac{\text{MAE}}{\text{Range (\%)}}$
q_1	0.06784	0.01493	0.02056
q_2	0.04062	0.01280	0.01846
q_3	0.04883	0.01132	0.02713
q_4	0.12098	0.03151	0.03781
q_5	0.02125	0.00656	0.00885
q_6	1.56557	0.04419	0.43489

MAE: mean absolute error.

Table 5. The best architecture found during the grid search process.[a]

Joint	Hidden layers	Activation function	Solver	α	α Adjustment	L2	R^2	σ	MAE	σ
q_1	(100, 100, 100, 100)	Tanh	Adam	0.1	Adaptive	0.001	0.98	0.02	0.07	0.01
q_2	(100, 100, 100, 100)	Tanh	LBFGS	0.5	Adaptive	0.1	0.99	0.01	0.04	0.01
q_3	(100, 100, 100)	ReLU	Adam	0.1	Adaptive	0.01	0.90	0.04	0.04	0.01
q_4	(100, 100, 100, 100)	Tanh	LBFGS	1e-05	Adaptive	0.01	0.94	0.02	0.12	0.03
q_5	(100, 100, 100, 100)	Tanh	LBFGS	0.5	Inverse scaling	0.001	0.99	0.00	0.02	0.01
q_6	(10, 10, 10, 10, 10)	Identity	Adam	0.1	Constant	0.1	0.7	0.00	1.57	0.04

MAE: mean absolute error.

[a]For each joint best hyperparameters and achieved performance metrics are given.

cases except for the q_6 output, which can be explained in the same manner as the high MAE for the regressed angle value of the sixth joint. Except for the q_6, the worst performing network in terms of R^2 score is the regressor for the third joint q_3, with R^2 score of 0.9. Still, even with a comparatively low R^2 score, it achieves a low MAE of 0.04°. Except for the q_6, in terms of the MAE, the worst performing regression model is q_4, with MAE of 0.12°, still a relatively high R^2 score is displayed (0.94) by the regression model in question. Standard errors across the 10 folds of the algorithms are low for both metrics used, indicating stable solutions for all the regression goals.

Observing the hyperparameters, it can be seen that the number of the hidden layers and neurons per layer tend toward the higher end of the tested hyperparameters. Four of the best performing networks (for the regression goals q_1, q_2, q_4, q_5) have the largest tested number of neurons—four hidden layers with 100 neurons each. It should be noted that all four of those networks used the hyperbolic tangent (Tanh) activation function inside the neurons and the LBFGS solver. Learning rate tended toward higher end of the tested parameters, except for the q_4, which used a comparatively low starting value. Adaptive learning rate yielded the best results except for q_5 and q_6, which used inverse scaling and constant learning rates, respectively. L2 regularization parameter varies between the networks. In the case of a higher L2 parameter being selected, it can be concluded that those models (q_2 and q_6) had certain inputs, which had a high influence on the outputs, and that influence needed to be lowered, which was achieved through the regularization.

Conclusion

This article presents the application of MLP ANN for the goal of inverse kinematics modeling. Grid search algorithm has been applied and the best hyperparameters for each of the six output goals have been determined. The scores achieved are satisfactory, with all models achieving an error below 1% of the appropriate output range. The models of the first five joints tended toward the largest tested networks. This indicates that higher quality results, if such are needed, may possibly be achieved with even larger networks. The last joint having a lower regression score indicates a lack of information, due to it only being affected by the orientation of the end-effector data, which was not used in the dataset. This implies that modeling of similarly configured robotic manipulators using the described method may require that data if a high precision is needed for the final joint.

Future work will include the testing of other methods, which may yield simpler to use models, such as symbolic regression, and the expansion of the dataset include obstacle data.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed the receipt of following financial support for the research, authorship, and/or publication of this article: This research was (partly) supported by the CEEPUS network CIII-HR-0108, European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS), project CEKOM under the grant KK.01.2.2.03.0004, CEI project "COVIDAi" (305.6019-20), University of Rijeka scientific grant uniri-tehnic-18-275-1447 and the project VEGA 1/0504/17.

ORCID iD

Zlatan Car https://orcid.org/0000-0003-2817-9252

References

1. Xu W, Liang B, Li C, et al. Path planning of free-floating robot in cartesian space using direct kinematics. *Int J Adv Robot Syst* 2007; 4(1): 17–26.
2. Wang J, Li Y, and Zhao X. Inverse kinematics and control of a 7-DOF redundant manipulator based on the closed-loop algorithm. *Int J Adv Robot Syst* 2010; 7(4): 37.
3. Božek P and Lozhkin A. The precision calculating method of robots moving by the plane trajectories. *Int J Adv Robot Syst* 2019; 16(6). DOI: 1729881419889556.
4. Wei Y, Nie X, Hiraga M, et al. Developing end-to-end control policies for robotic swarms using deep Q-learning. *J Adv Comput Intell Intell Inform* 2019; 23(5): 920–927.
5. Sassi MA, Otis MJ, and Campeau-Lecours A. Active stability observer using artificial neural network for intuitive physical

human–robot interaction. *Int J Adv Robot Syst* 2017; 14(4). DOI: 1729881417727326.
6. Tang SH, Ang CK, Ariffin MKABM, et al. Predicting the motion of a robot manipulator with unknown trajectories based on an artificial neural network. *Int J Adv Robot Syst* 2014; 11(10): 176.
7. Baressi Šegota S, Anđelić N, Lorencin I, et al. Path planning optimization of six-degree-of-freedom robotic manipulators using evolutionary algorithms. *Int J Adv Robot Syst* 2020; 17(2). DOI: 1729881420908076.
8. Wei Y, Hiraga M, Ohkura K, et al. Autonomous task allocation by artificial evolution for robotic swarms in complex tasks. *Artif Life Robot* 2019; 24(1): 127–134.
9. Car Z and Mikac T. Evolutionary approach for solving cell-formation problem in cell manufacturing. *Adv Eng Inform* 2006; 20(3): 227–232.
10. Srinivas S, Sarvadevabhatla RK, Mopuri KR, et al. A taxonomy of deep convolutional neural nets for computer vision. *Front Robot AI* 2016; 2: 36.
11. Jia W, Mou S, Wang J, et al. Fruit recognition based on pulse coupled neural network and genetic Elman algorithm application in apple harvesting robot. *Int J Adv Robot Syst* 2020; 17(1).
12. Joshy P and Supriya P. Implementation of robotic path planning using ant colony optimization algorithm. In: *2016 International conference on inventive computation technologies (ICICT)* (ed Y Robinson), volume 1. Coimbatore, India, 26–27 August 2016, pp. 1–6. IEEE.
13. Sun C, Li G, and Xu J. Adaptive neural network terminal sliding mode control for uncertain spatial robot. *Int J Adv Robot Syst* 2019; 16(6). DOI: 1729881419894065.
14. Sapietová A, Saga M, Kuric I, et al. Application of optimization algorithms for robot systems designing. *Int J Adv Robot Syst* 2018; 15(1). DOI: 1729881417754152.
15. Zhang P, Xiong C, Li W, et al. Path planning for mobile robot based on modified rapidly exploring random tree method and neural network. *Int J Adv Robot Syst* 2018; 15(3): DOI: 1729881418784221.
16. Wenhui Z, Hongsheng L, Xiaoping Y, et al. Adaptive robust control for free-floating space robot with unknown uncertainty based on neural network. *Int J Adv Robot Syst* 2018; 15(6). DOI: 1729881418811518.
17. El_Jerjawi NS and Abu-Naser SS. Diabetes prediction using artificial neural network. *J Adv Sci* 2018; 124: 1–10.
18. Lorencin I, Anđelić N, Mrzljak V, et al. Genetic algorithm approach to design of multi-layer perceptron for combined cycle power plant electrical power output estimation. *Energies* 2019; 12(22): 4352.
19. Briones JC, Flores B, and Cruz-Cano R. Multi-mode radar target detection and recognition using neural networks. *Int J Adv Robot Syst* 2012; 9(5): 177.
20. Villegas R, Yang J, Ceylan D, et al. Neural kinematic networks for unsupervised motion retargetting. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (eds D Plummer and I Torwick), Salt Lake City, UT, USA, 18–23 June 2018, pp. 8639–8648. IEEE.
21. Nemeth M, Nemethova A, and Michalconok G. Determination issues of data mining process of failures in the production systems. In: *Computer science on-line conference* (ed R Silhavy), Zlin, Czech Republic, 24 April 2019, pp. 200–207. Cham: Springer.
22. Ghafil HN, László K, and Jármai K. Investigating three learning algorithms of a neural networks during inverse kinematics of robots. In: *Solutions for sustainable development: proceedings of the 1st international conference on engineering solutions for sustainable development (ICESSD 2019)* (eds K Szita Tóthné, K Jármai, and K Voith), 3–4 October 2019, Miskolc, Hungary, p. 33. CRC Press.
23. Demby's J, Gao Y, and DeSouza G. A study on solving the inverse kinematics of serial robots using artificial neural network and fuzzy neural network. In: *2019 IEEE international conference on fuzzy systems (FUZZ-IEEE)* (ed J Garibaldi), New Orleans, LA, USA, 23–26 June 2019, pp. 1–6. IEEE.
24. Ghasemi J, Moradinezhad R, and Hosseini M. Kinematic synthesis of parallel manipulator via neural network approach. *Int J Eng* 2019; 30(9): 1319–1325.
25. Dash KK, Choudhury B, and Senapati S. Inverse kinematics solution of a 6-dof industrial robot. In: *Soft computing in data analytics*. Singapore: Springer, 2019, pp. 183–192.
26. Zarrin A, Azizi S, and Aliasghary M. A novel inverse kinematics scheme for the design and fabrication of a five degree of freedom arm robot. *Int J Dyn Control* 2020; 8: 1–11.
27. Takatani H, Araki N, Sato T, et al. Neural network-based construction of inverse kinematics model for serial redundant manipulators. *Artif Life Robot* 2019; 24: 1–7.
28. Dereli S and Köker R. A meta-heuristic proposal for inverse kinematics solution of 7-dof serial robotic manipulator: quantum behaved particle swarm algorithm. *Artif Intell Rev* 2020; 53: 949–964.
29. Lim DW and Lee YK. On the number of training samples for inverse kinematics solutions by artificial neural networks. In: *2019 16th International conference on ubiquitous robots (UR)* (ed K Minjun), Jeju, Korea (South), 24–27 June 2019, pp. 61–64. IEEE.
30. Valayil TP, Selladurai V, and Ramaswamy NR. Kinematic modeling of a serial robot using Denavit-Hartenberg method in Matlab. *(TAGA) J graph technol* 2018; 14: 2347–2445. http://www.tagajournal.com/gallery/v14.230.pdf (accessed 18 February 2020).
31. Spong MW and Vidyasagar M. *Robot dynamics and control*. Chichester: John Wiley & Sons, 2008.
32. Siciliano B and Khatib O. *Springer handbook of robotics*. Berlin: Springer, 2016.
33. Tsai LW. *Robot analysis: the mechanics of serial and parallel manipulators*. New York, NY: John Wiley & Sons, 1999.
34. Ramos PB, Medina JM, Salcedo MC, et al. Application of the Denavit-Hartenberg method to estimate the positioning errors of an automated XYZ cartesian table. *Contemp Eng Sci* 2018; 11: 3483–3493.

35. Craig JJ. *Introduction to robotics: mechanics and control, 3/E*. Chennai: Pearson Education India, 2009.
36. Li N and Ping X. Research on DH parameter modeling methods. *Int J Comput Sci Control Eng* 2019; 7(1): 8–16.
37. Becerra Y, Arbulu M, Soto S, et al. A comparison among the Denavit-Hartenberg, the screw theory, and the iterative methods to solve inverse kinematics for assistant robot arm. In: *International conference on swarm intelligence*. Chiang Mai, Thailand, 26–30 July 2019, pp. 447–457. New York: Springer.
38. Azhar MW. *Kinematics modelling of robot manipulator using solidworks*. PhD Thesis, Universitas Muhammadiyah Surakarta, Indonesia, 2019.
39. Ying SJ, Dubey R, and Sundarrao S. Gyroscope for robot to sense the balance. *Int J Robot Eng* 2018; 3(2): 012.
40. ABB, Affolternstrasse 44 8050 Zurich Switzerland. *Product Specification IRB-120*, 1st ed. Zurich, Switzerland: ABB Group, 2019.
41. Lorencin I, Anđelić N, Španjol J, et al. Using multi-layer perceptron with Laplacian edge detector for bladder cancer diagnosis. *Artif Intell Med* 2019; 102: 101746.
42. Alom MZ, Taha TM, Yakopcic C, et al. A state-of-the-art survey on deep learning theory and architectures. *Electronics* 2019; 8(3): 292.
43. Goodfellow I, Bengio Y, and Courville A. *Deep learning*. Cambridge, MA: MIT Press, 2016.
44. Hastie T and Friedman . *Elements of statistical learning: data mining, inference, and prediction*) Berlin, Germany: Springer, 2003.
45. Bishop CM. *Pattern recognition and machine learning*. Berlin: Springer Science + Business Media, 2006.
46. Shamsolmoali P, Zareapoor M, Jain DK, et al. Deep convolution network for surveillance records super-resolution. *Multimed Tools Appl* 2019; 78(17): 23815–23829.
47. Nagarajan HP, Mokhtarian H, Jafarian H, et al. Knowledge-based design of artificial neural network topology for additive manufacturing process modeling: a new approach and case study for fused deposition modeling. *J Mech Des* 2019; 141(2): 021705.
48. Ravanelli M, Brakel P, Omologo M, et al. A network of deep neural networks for distant speech recognition. In: *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, New Orleans, LA, USA, 5–9 March 2017, pp. 4880–4884.
49. Pineda FJ. Generalization of back-propagation to recurrent neural networks. *Phys Rev Lett* 1987; 59(19): 2229.
50. Bhattacherjee A, Roy S, Paul S, et al. Classification approach for breast cancer detection using back propagation neural network: a study. In: *Deep learning and neural networks: concepts, methodologies, tools, and applications*. Pennsylvania: IGI Global, 2020, pp. 1410–1421.
51. Loshchilov I and Hutter F. Fixing weight decay regularization in Adam. 2017.
52. Ranganathan V and Natarajan S. A new backpropagation algorithm without gradient descent. 2018; *arXiv:1802.00027*, 1–15.
53. Zhou X. Understanding the convolutional neural networks with gradient descent and backpropagation. *J Phys Conf Series* 2018; 1004: 012028.
54. Smith LN. A disciplined approach to neural network hyper-parameters: part 1–learning rate, batch size, momentum, and weight decay. 2018; *arXiv:1803.09820*, 1–21.
55. Takase T, Oyama S, and Kurihara M. Effective neural network training with adaptive learning rate based on training loss. *Neural Netw* 2018; 101: 68–78.
56. Li H, Xu Z, Taylor G, et al. Visualizing the loss landscape of neural nets. In: *Advances in neural information processing systems*. Montréal, Canada, December 2018, pp. 6389–6399.
57. Alvi M, Zisserman A, and Nellåker C. Turning a blind eye: explicit removal of biases and variation from deep neural network embeddings. In: *Proceedings of the european conference on computer vision (ECCV)* (eds A Vedaldi, H Vedaldi, Th Brox, and J Frahm), Munich, Germany, 8–14 September 2018. Berlin, Germany: Springer.
58. Rahaman N, Baratin A, Arpit D, et al. On the spectral bias of neural networks 2018; *arXiv:1806.08734v3*, 1–23.
59. Gunasekar S, Lee JD, Soudry D, et al. Implicit bias of gradient descent on linear convolutional networks. In: *Advances in neural information processing systems*. Montréal, Canada, 2018, pp. 9461–9471.
60. Yang H, Su J, Zou Y, et al. Layout hotspot detection with feature tensor generation and deep biased learning. *IEEE Trans Comput-Aided Des Integ Circuits Syst* 2018; 38(6): 1175–1187.
61. Dauphin Y, De Vries H, and Bengio Y. Equilibrated adaptive learning rates for non-convex optimization. In: *Advances in neural information processing systems* (eds M Jordan, Y LeCun, and S Solla), Montreal, Quebec, Canada, 7–12 December 2015, pp. 1504–1512.
62. Baressi Šegota S, Lorencin I, Anđelić N, et al. Improvement of marine steam turbine conventional exergy analysis by neural network application. *J Marine Sci Eng* 2020; 8(11): 884.
63. Wu R, Huang H, Qian X, et al. A L-BFGS based learning algorithm for complex-valued feedforward neural networks. *Neural Process Lett* 2018; 47(3): 1271–1284.
64. Car Z, Baressi Šegota S, Anđelić N, et al. Modeling the spread of COVID-19 infection using a multilayer perceptron. *Comput Math Method Med* 2020; 2020: 1–10.
65. Sysoev O and Burdakov O. A smoothed monotonic regression via l2 regularization. *Knowl Inform Syst* 2019; 59(1): 197–218.
66. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: machine learning in python. *J Mach Learn Res* 2011; 12: 2825–2830.
67. Yasin H, Caraka RE, Hoyyi A, et al. Prediction of crude oil prices using support vector regression (SVR) with grid search-cross validation algorithm. *Glob J Pure Appl Math* 2016; 12(4): 3009–3020.
68. Syarif I, Prugel-Bennett A, and Wills G. SVM parameter optimization using grid search and genetic algorithm to

improve classification performance. *Telkomnika* 2016; 14(4): 1502.

69. Li L, Tan Y, He C, et al. Case study of the effect of grid-search objective function in microseismic source location. In: *International conference and exhibition* (ed M Francis), Barcelona, Spain, 3–6 April 2016, pp. 168–168. Society of Exploration Geophysicists and American Association of Petroleum.
70. Kaplan S and Gürcan EK. Comparison of growth curves using non-linear regression function in Japanese quail. *J Appl Anim Res* 2018; 46(1): 112–117.
71. Piepho HP. A coefficient of determination (r2) for linear mixed models. *Biometrical Journal* 2018; 61(4): 860–872.
72. Yang SJ, Lu OH, Huang AY, et al. Predicting students' academic performance using multiple linear regression and principal component analysis. *J Inform Process* 2018; 26: 170–176.
73. Zeinadini Meymand A, Bagheri Bodaghabadi M, Moghimi A, et al. Modeling of yield and rating of land characteristics for corn based on artificial neural network and regression models in southern Iran. *Desert* 2018; 23(1): 85–95.
74. Dershem R, Chu X, Wood G, et al. Response to 'regression to the mean, apparent data errors, and biologically extraordinary results'. *Int J Obes* 2018; 42(4): 951.
75. Ingrassia S and Punzo A. Cluster validation for mixtures of regressions via the total sum of squares decomposition. *J Classif* 2020; 37: 526–547.
76. Dong Z, Xie L, Yang Y, et al. Local sensitivity analysis of kinetic models for cellulose pyrolysis. *Waste Biomass Valori* 2019; 10(4): 975–984.
77. Kovaleva E, Dolomatov M, Latypov Đ, et al. Possibility of predicting activation energy for viscous flow in five-membered naphthenes by means of structural descriptors. *Am J Phys Chem* 2019; 8(1): 26–31.
78. Kasuya E. On the use of r and r squared in correlation and regression. *Ecol Res* 2019; 34(1): 235–236.
79. Camero A, Toutouh J, and Alba E. A specialized evolutionary strategy using mean absolute error random sampling to design recurrent neural networks. 2019; *arXiv:1909.02425*, 1–10
80. Kertész Á, Hlaváčová Z, Vozáry E, et al. Relationship between moisture content and electrical impedance of carrot slices during drying. *Int Agrophys* 2015; 29(1): 61–66.
81. Schober P, Boer C, and Schwarte LA. Correlation coefficients: appropriate use and interpretation. *Anesth Analg* 2018; 126(5): 1763–1768.
82. Vimala S and Vivekanandan K. A novel biclustering with mean absolute difference similarity measure for collaborative filtering recommender system. *Int J Pure Appl Math* 2018; 118(20): 1–7.
83. Langdon WB, Dolado J, Sarro F, et al. Exact mean absolute error of baseline predictor, marp0. *Inform Softw Technol* 2016; 73: 16–18.