Big Data and Information Analytics

https://www.aimspress.com/journal/BDIA

BDIA, 8: 65–79.
DOI:10.3934/bdia.2024004
Received: 17 September 2024
Revised: 14 October 2024
Accepted: 17 October 2024
Published: 30 October 2024

Research article

Multivariate polynomial regression by an explainable sigma-pi neural network

Xiaoxiang Guo[1], **Zuolin Shi**[1] **and Bin Li**[2,*]

[1] School of Mathematics and Statistics, Henan Academy of Big Data, Zhengzhou University, Zhengzhou 450001, China

[2] School of Water Conservancy and Transportation, Zhengzhou University, Zhengzhou 450001, China

* **Correspondence:** Email: binli1123@zzu.edu.cn.

Abstract: Over the years, data-driven regression on univariate functions has been extensively studied. However, fast, effective, and stable algorithms for multivariate function fitting are still lacking. Recently, Kolmogorov-Arnold networks have garnered significant attention among scholars due to their superior accuracy and interpretability compared to multi-layer perceptrons. In this paper, we have demonstrated that the sigma-pi neural network, a form of Kolmogorov-Arnold networks, can efficiently fit multivariate polynomial functions, including fractional-order multivariate polynomials. Three examples were employed to illustrate the regression performance of the designed neural networks. The explainable sigma-pi neural network will lay the groundwork for further development of general tools for multivariate nonlinear function regression problems.

Keywords: data-driven; fitting; interpretability; sigma-pi neural network; multivariate polynomial

1. Introduction

With the rapid development of big data and artificial intelligence, data-driven predictions now permeate nearly every aspect of modern science [1, 2]. Regression analysis, a key component of data-driven science, has wide applications across various domains, including new material design, stock prediction, medical diagnosis, and geological exploration, etc. [3–9]. Besides, machine learning offers numerous methods for regression analysis, such as support vector regression (SVR), decision trees (DTs), multi-layer perceptrons (MLPs), deep neural networks, etc. Although those methods can achieve high accuracy, the "black box" nature of neural networks limits their interpretability.

According to the Weierstrass approximation theorem, continuous functions defined on closed intervals can be uniformly approximated by polynomial functions [10]. Moreover, a polynomial

function has a simpler form compared to other complex functions, making polynomial regression a natural choice for practical applications [11, 12]. Polynomial regression can be divided into univariate function fitting and multi-variate function regression. While univariate function fitting is highly effective supported by popular data analysis software such as Origin, MATLAB, and SPSS based on the least squares algorithm, there remains a lack of fast, effective, and stable algorithms for multivariate function regression.

The theme of this paper is to design a fast, efficient and accurate algorithm to solve the multivariate polynomial regression problems (Eq (1.1)).

$$\begin{aligned} y &= f(x_1, x_2, \cdots, x_n) \\ &= \sum_{i=1}^{n} \alpha_i x_i^{\beta_i} + \sum_{j,k=1, j\neq k}^{n} \gamma_{jk} x_j^{\mu_j} x_k^{\nu_k} + \sum_{s,p,q=1, s\neq p\neq q}^{n} \eta_{spq} x_s^{\sigma_s} x_p^{\lambda_p} x_q^{\tau_q} + \cdots \end{aligned} \tag{1.1}$$

In the regression process, SVR introduces a kernel function to construct the nonlinear relation. The MLP method employs the weighted summation of variables ($\sum_i \omega_{ji} x_i + \theta_j$) to achieve the aggregation of information. As the weighted summation only passes a linear relationship between variables, the nonlinear knowledge, for example the product relation $x_j^{\mu_j} x_k^{\nu_k}$, should be approximated by using an activation function, such as the sigmoid function, $f(x) = 1/(1 + e^{-x})$, the tanh function, $f(x) = (e^x - e^{-x})/(e^x + e^{-x})$, etc. However, the applications of kernel functions in SVR and activation functions in MLPs make it difficult to interpret the machine learning model.

Recently, Kolmogorov-Arnold networks (KANs) have gained attention due to their superior accuracy and interpretability compared to multi-layer perceptrons [13]. MLPs have fixed activation functions on nodes while KANs have learnable activation functions on edges. The Kolmogorov-Arnold representation theorem established that, if f is a multivariate continuous function on a bounded domain, then f can be written as a finite composition of continuous functions of a single variable and the binary operation of addition. The regression model of KANs is in the form of

$$\begin{aligned} y &= f(x_1, x_2, \cdots, x_n) \\ &= \sum_{i_{L-1}=1}^{n_{L-1}} \phi_{L-1,i_L,i_{L-1}} \Big(\sum_{i_{L-2}=1}^{n_{L-2}} \phi_{L-2,i_{L-1},i_{L-2}} \cdots \Big(\sum_{i_2=1}^{n_2} \phi_{2,i_3,i_2} \Big(\sum_{i_1=1}^{n_1} \phi_{1,i_2,i_1} \Big) \Big(\sum_{i_0=1}^{n} \phi_{0,i_1,i_0}(x_{i_0}) \Big) \Big) \Big) \end{aligned} \tag{1.2}$$

where, L represents the number of network layers and ϕ_{k,i_{k+1},i_k} is a continuous functions of signal variable. That is to say, KANs train and select suitable signal variable continuous functions in each network layer from the given basic function library including polynomial functions, exponential functions, logarithmic functions, etc. Although, KANs have higher accuracy and interpretability, the regression of product relation in multivariate polynomial functions still need to transform to the coupling of univariate functions, which will undoubtedly increase the complexity of the regression.

The traditional artificial neural network uses nodes whose information transmission is via a sigma unit, i.e., a linear weighted sum of inputs, $a = \sum_i \omega_i x_i$. A pi unit is constructed by replacing the weighted summation in the activation with a weighted product, $a = \prod_i x_i^{\omega_i}$. Units of this type are also designated "higher order" in that they contain polynomial terms of order greater than one (linear). The simplest way of modeling such a product relation in multivariate polynomial functions is to increase the complexity of the node by a pi unit. Moreover, a sigma-pi unit can formulate a sum of product

terms,

$$y = \sum_{j=1}^{m} c_j a_j = \sum_{j=1}^{m} c_j \prod_{i=1}^{n} x_i^{\omega_{ji}}. \tag{1.3}$$

Inspired by this architecture of a sigma-pi unit, we find the high-order sigma-pi neural network (SPNN) [14, 15] can efficiently achieve the regression of multivariate polynomial functions. The coefficients and exponents of the multivariate polynomial can be determined through the trained neural network weight parameters (c_j and ω_{ji}).

Performing the regression of multivariate polynomial functions by SPNN has the following advantages compared with that of MLPs and KANs.

- Polynomial regression by the SPNN model has better interpretability than that of MLPs, due to the fact that high-order terms are generated by pi units instead of complex activation functions.
- Polynomial regression by the SPNN model has faster learning efficiency compared with that of KANs, owing to the application of a fixed activation function in polynomial form.
- The optimization and control of the parameters in the SPNN model are more convenient than that in MLPs and KANs. Besides, the forms of the SPNN model are more friendly to real applications.

2. Method and modeling

2.1. The high-order sigma-pi neural network

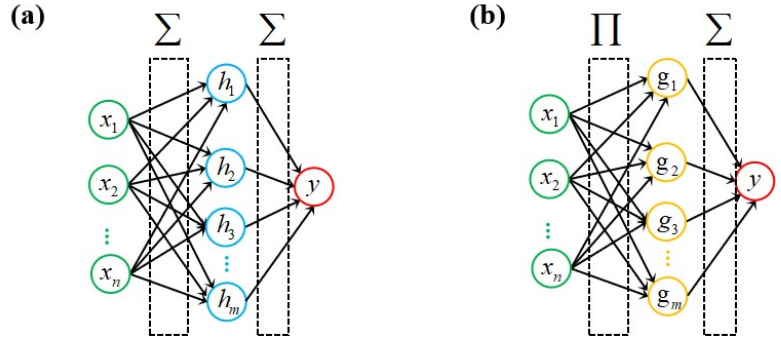


Figure 1. Different structure units in neural networks: (a) sigma-sigma units in the MLP network; (b) sigma-pi units in the high-order network.

In an MLP neural network, information from different input signals is transmitted through weighted summation. For example, in a two-layer forward neural network (see Figure 1(a)), the hidden layer

$$h_i = \sigma_1(\sum_{j=1}^{n} \alpha_{ij} x_j + b_i), \tag{2.1}$$

and the output signal

$$y = \sigma_2(\sum_{i=1}^{m} \beta_i h_i + \theta), \tag{2.2}$$

where σ_1 is the activation function of the hidden layer, σ_2 is the activation function of the output layer, α is the weight of the first layer network, β is the weight of the second layer network, and b and θ are the biases. The information transmission from x_j to h_i, and from h_i to y, are all by weighted summation. So, this kind of structure unit in MLP neural networks can be called sigma-sigma units.

To embed high-order coupled information into a neural network, high-order neural networks are designed. For instance, the sigma-pi neural network (SPNN) enables the embedding of product information into the network. In SPNN, the hidden layer

$$g_i = \sigma_1(\prod_{j=1}^{n} x_j^{\alpha_{ij}} + b_i), \tag{2.3}$$

and the output layer

$$y = \sigma_2(\sum_{i=1}^{m} \beta_i g_i + \theta). \tag{2.4}$$

The information transmission from x_j to g_i is by weighted product, and from g_i to y is by weighted summation, see Figure 1(b).

2.2. Multivariate polynomial regression

As previously mentioned, the basic unit of SPNN is the power function of input signals. We aim to achieve multivariate polynomial regression using the explainable SPNN. In this model, if the activation functions, σ_1 and σ_2, are linear, the output signal

$$\begin{aligned} y &= \sum_{i=1}^{m} \beta_i g_i + \theta \\ &= \sum_{i=1}^{m} \beta_i (\prod_{j=1}^{n} x_j^{\alpha_{ij}} + b_i) + \theta \\ &= \sum_{i=1}^{m} \beta_i (\prod_{j=1}^{n} x_j^{\alpha_{ij}}) + \sum_{i=1}^{m} \beta_i b_i + \theta. \end{aligned} \tag{2.5}$$

Cleary, Eq (2.5) is a multivariate polynomial function, where β_i is the polynomial coefficient, α_{ij} is the polynomial exponent, and $\sum_{i=1}^{m} \beta_i b_i + \theta$ is the constant term. Thus, this explainable SPNN is essentially a type of Kolmogorov-Arnold network.

To illustrate the regression process of a multivariate polynomial function more clearly, we present a simple regression problem as an example:

$$\begin{aligned} y =& f(x_1, x_2, \cdots, x_n) \\ =& a_1 x_1^{b_1} + a_2 x_2^{b_2} + \cdots + a_n x_n^{b_n} + \sum_{i,j=1, i\neq j}^{n} c_{ij} x_i^{p_i} x_j^{q_j}. \end{aligned} \tag{2.6}$$

Here, $\{x_1, x_2, \cdots, x_n\}$ represents the state variables, and y is the target variable. Since Eq (2.6) lacks a constant term, bias parameters are not used in the designed network. A two-layer SPNN is employed to solve this regression problem. The number of neurons in the input layer equals n, while the hidden

layer contains $n + n(n-1) = n^2$ neurons. The neural connections from the input layer to the hidden layer primarily capture the regression of power functions, such as $x_i^{b_i}$ and $x_i^{p_i} x_j^{q_j}$, corresponding to each term in the polynomial function. The network then performs a weighted summation of all terms from the hidden layer to the output layer.

Additionally, to accelerate the parameter optimization, we convert the above sigma-pi neural network to the sigma-sigma neural network, and use the back-propagation algorithm to update the parameters. The input signals are first transformed using logarithmic functions,

$$x_i \Rightarrow u_i = \begin{cases} \log(x_i), x_i > 0; \\ -\infty, x_i = 0; \\ \log(-x_i), x_i < 0. \end{cases} \tag{2.7}$$

Next, a sigma unit operates on the u_i,

$$v_j = \sum_{i=1}^{n} \alpha_{ji} u_i = \sum_{i=1}^{n} \log(|x_i|^{\alpha_{ji}}). \tag{2.8}$$

Here α_{ji} represents the weight from the i^{th} neuron in the input layer to the j^{th} neuron in the hidden layer. A signed exponential function is then applied to v_j,

$$\begin{aligned} h_j &= sig(x_i) \exp(v_j) \\ &= sig(x_i) \exp(\sum_{i=1}^{n} \log(|x_i|^{\alpha_{ji}})) \\ &= \prod_{i=1}^{n} x_i^{\alpha_{ji}}. \end{aligned} \tag{2.9}$$

Therefore, the conversion from the pi unit to the sigma unit can be achieved using logarithmic and exponential operators. The output signal, y, is then obtained by the sigma unit from the hidden layer to the output layer.

$$\begin{aligned} y &= \sum_{j=1}^{m} \beta_j h_j \\ &= \sum_{j=1}^{m} \beta_j \prod_{i=1}^{n} x_i^{\alpha_{ji}} \\ &= \sum_{j=1}^{m} \beta_j sig(x_i) \exp(\sum_{i=1}^{n} \alpha_{ji} \log(|x_i|)). \end{aligned} \tag{2.10}$$

Here, β_j is the weight from the j^{th} neuron in the hidden layer to the output signal. The model error is defined as

$$Err = \frac{1}{2}(y - \hat{y})^2, \tag{2.11}$$

where y is the model output and $\hat{y}$ is the real value. For parameter updates, the gradient descent method is employed,

$$\beta_j = \beta_j - \eta \Delta \beta_j, \alpha_{ji} = \alpha_{ji} - \eta \Delta \alpha_{ji}. \tag{2.12}$$

Here, η is the learn rate,

$$\begin{aligned}\Delta\beta_j &= \frac{\partial Err}{\partial \beta_j} \\ &= \frac{\partial Err}{\partial y} * \frac{\partial y}{\partial \beta_j} \\ &= (y - \hat{y}) * h_j, \end{aligned} \tag{2.13}$$

and

$$\begin{aligned}\Delta\alpha_{ji} = \frac{\partial Err}{\partial \alpha_{ji}} &= \frac{\partial Err}{\partial v_j} * \frac{\partial v_j}{\partial \alpha_{ji}} \\ &= \frac{\partial Err}{\partial y} * \frac{\partial y}{\partial h_j} * \frac{\partial h_j}{\partial v_j} * \frac{\partial v_j}{\partial \alpha_{ji}} \\ &= (y - \hat{y}) * \beta_j * \exp(v_j) * sig(x_i) * \log(|x_i|). \end{aligned} \tag{2.14}$$

The optimized parameters β and α are the corresponding multivariate polynomial regression coefficients and exponents.

2.3. Algorithm pseudo code

The pseudo code of multivariate polynomial regression by our designed neural network is presented in Algorithm 1 below.

Algorithm 1 Multivariate polynomial regression by the explainable sigma-pi neural network

Inputs: the corresponding multi-variables, $x_1, x_2, \cdots, x_n$, the target variable, y, and the desired multi-variable polynomial form, for example, $y = a_1 x_1^{b_1} + a_2 x_2^{b_2} + \cdots + a_n x_n^{b_n} + \sum_{i,j=1, i\neq j}^{n} c_{ij} x_i^{p_i} x_j^{q_j}$.

Outputs: the regression coefficients and exponents, such as, $a_i, b_i, c_{ij}, p_i, q_j$.

Divide the data into the train set, valid set, and test set.
Construct the sigma-pi neural network based on the desired empirical formula.
Variable conversion, $u_i \Leftarrow x_i$.
Let the sigma unit work on u_i, s.t., $u_i \Rightarrow v_j$.
The signed exponentiation effect on v_j, s.t., $v_j \Rightarrow h_j$.
Let the sigma unit work on h_j, s.t., $h_j \Rightarrow y$, compute the regression error, $Err = \frac{1}{2}(y - \hat{y})^2$, set the error threshold as ε, and define the total epochs as N.
while $epoch < N$ **do**
    Parameters update, $\beta_j = \beta_j - \eta\Delta\beta_j, \alpha_{ji} = \alpha_{ji} - \eta\Delta\alpha_{ji}$.
    if $Err < \varepsilon$ **then**
        break;
        output the weight parameters, $\alpha_{ji} \Rightarrow b_i, p_i, q_j$ and $\beta_j \Rightarrow a_i, c_{ij}$.
    else if $Err > \varepsilon$ **then**
        continue;
    end if
end while

3. Examples and application

In this section, we present two examples to demonstrate the feasibility and effectiveness of the proposed model. Moreover, we illustrate the application of our method in analyzing multivariate correlated regression, specifically concerning the maximum stress on concrete pipes under traffic loads.

3.1. Example 1: Integer order multivariate polynomial regression

$$y = 2 * x_1^2 * x_2 + 3 * x_1^3 * x_2 + 4 * x_1^2 * x_2^2. \tag{3.1}$$

In this example, the model is used to determine the regression coefficients and exponents of Eq (3.1) based on generated data. The variable x_1 is linearly sampled from the interval $[1, 3]$, x_2 is randomly sampled from the interval $[1, 4]$, and y is calculated according to Eq (3.1). These generated data are briefly presented in Figure 2, where Figure 2(a) gives the time series of x_1, x_2, and y; Figure 2(b),(c) display scatter plots of y versus x_1, and y versus x_2, respectively.

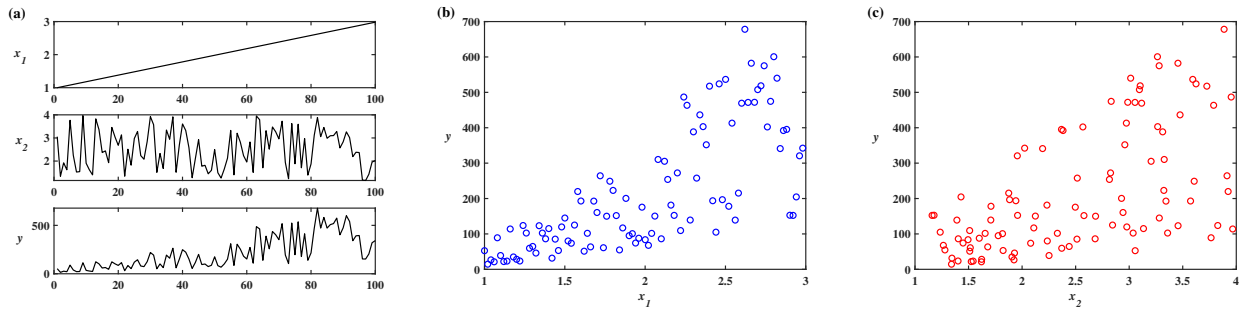


Figure 2. Generated data for integer order polynomial regression: (a) the time series of x_1, x_2, and y; (b) the scatter plot of y versus x_1; (c) the scatter plot of y versus x_2.

The number of neurons in the hidden layer equals 3, i.e., the number of terms of polynomial function Eq (3.1). Let $u = \log(x_1), v = \log(x_2)$, and the signals u and v are fed into the sigma unit as inputs. It should be noted that the variables x_1 and x_2 in this example are restricted to positive values. We use the linear activation function in each architecture, thus, the neurons in the hidden layer satisfy

$$\begin{aligned} h_1 &= \alpha_{11}u + \alpha_{12}v = log(x_1^{\alpha_{11}}) + log(x_2^{\alpha_{12}}), \\ h_2 &= \alpha_{21}u + \alpha_{22}v = log(x_1^{\alpha_{21}}) + log(x_2^{\alpha_{22}}), \\ h_3 &= \alpha_{31}u + \alpha_{32}v = log(x_1^{\alpha_{31}}) + log(x_2^{\alpha_{32}}). \end{aligned}$$

So, by the exponential operation,

$$\begin{aligned} e^{h_1} &= e^{log(x_1^{\alpha_{11}}) + log(x_2^{\alpha_{12}})} = x_1^{\alpha_{11}} * x_2^{\alpha_{12}}, \\ e^{h_2} &= e^{log(x_1^{\alpha_{21}}) + log(x_2^{\alpha_{22}})} = x_1^{\alpha_{21}} * x_2^{\alpha_{22}}, \\ e^{h_3} &= e^{log(x_1^{\alpha_{31}}) + log(x_2^{\alpha_{32}})} = x_1^{\alpha_{31}} * x_2^{\alpha_{32}}. \end{aligned}$$

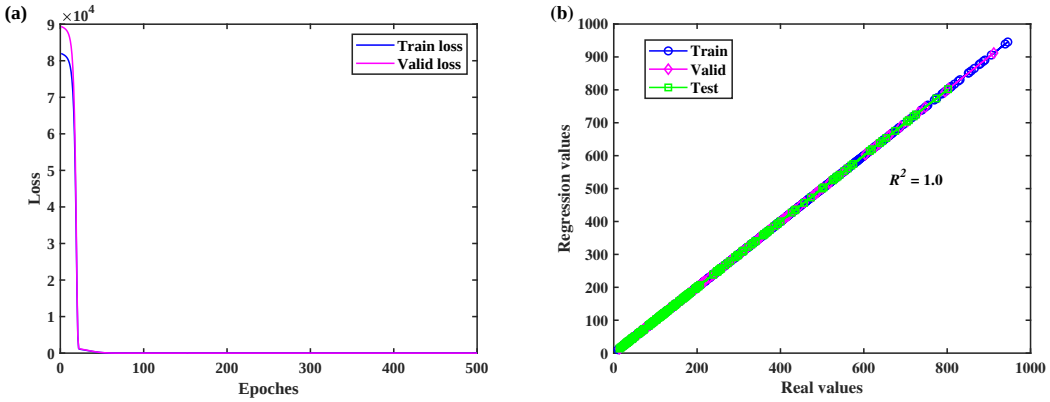
The output signal is the weighted sum of e^{h_1}, e^{h_2}, and e^{h_3} according to the sigma unit from the hidden layer to the output layer.

$$y = \beta_1 e^{h_1} + \beta_2 e^{h_2} + \beta_3 e^{h_3} = \beta_1 x_1^{\alpha_{11}} * x_2^{\alpha_{12}} + \beta_2 x_1^{\alpha_{21}} * x_2^{\alpha_{22}} + \beta_3 x_1^{\alpha_{31}} * x_2^{\alpha_{32}}.$$

Here, β_i represents the corresponding polynomial regression coefficient, and α_{ij} denotes the related polynomial regression exponent. Since we are addressing an integer-order polynomial regression problem, the parameters α_{ij} are constrained to integer values during the parameter iteration process. The training is conducted for 500 epochs, with a maximum learning rate of 0.01. Compared with other optimizers, such as BGD, SGD, and MBGD, the combination of the Adam optimizer with a one-cycle scheduler has faster convergence speed and better modeling performance, so we chose this optimization algorithm in the training process. The regression performance is evaluated using the mean squared error (MSE) and the coefficient of determination (R^2),

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}.$$

Here, y_i is the real value, $\hat{y}_i$ is the regression value, and $\bar{y}$ is the mean value of y. The modeling results are shown in Figure 3, the train loss and valid loss are depicted by the MSE, and the coefficient of determination $R^2 = 1.0$.



(a) (b)

Figure 3. The modeling results: (a) the train loss and valid loss depicted by MSE; (b) the plot of real values versus regression values, the coefficient of determination $R^2 = 1.0$.

The weight parameters of the proposed neural network are listed in Table 1. The parameter α_{ij} is related to the polynomial exponent, and parameter β_i corresponds to the polynomial coefficient.

Table 1. The values of weight parameters of our designed SPNN.

Parameters	α_{11}	α_{12}	α_{21}	α_{22}	α_{31}	α_{32}	β_1	β_2	β_3
Values	2	1	3	1	2	2	2.0	3.0	4.0

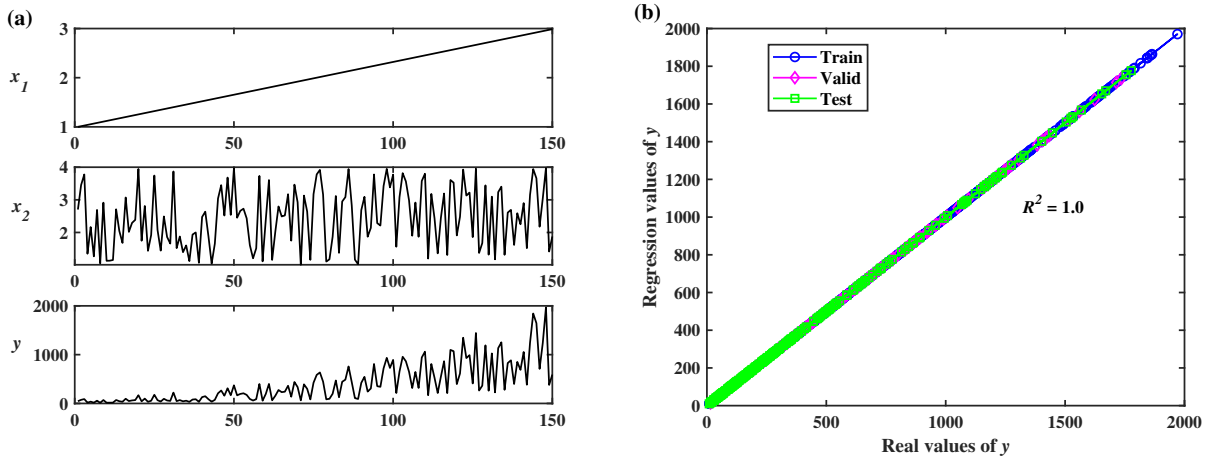
From the results above, it is evident that our designed SPNN effectively solves the integer-order multivariate polynomial regression problem. The coefficients and exponents of the integer-order

multivariate polynomial function align with the weight parameters of the designed SPNN. In the next example, we use the SPNN to solve a fractional-order multivariate polynomial regression problem.

3.2. Example 2: Fractional-order multivariate polynomial regression

$$y = 2 * x_1^{2.2} * x_2^{1.5} + 3 * x_1^{3.4} * x_2^{1.0} + 4 * x_1^{2.8} * x_2^{2.0}. \tag{3.2}$$

In this example, we use the designed SPNN to solve the fractional-order polynomial regression problem. The values of x_1 and x_2 are generated in the same manner as in Example 1, and the output signal is computed according to Eq (3.3). The dataset contains 3000 samples, and the training is conducted for 10,000 epochs. The Adam optimizer is employed combined with a one-cycle scheduler. The maximum learning rate is set to 0.009, while the optimizer's learning rate is initialized at 0.004. The corresponding time series are plotted in Figure 4(a), and the regression result is shown in Figure 4(b).



(a) (b)

Figure 4. The fractional-order polynomial regression: (a) the generated data of the fractional-order polynomial function; (b) the plot of real values versus regression values.

The values of mean squared error (MSE), coefficient of determination (R^2), and corresponding weight parameters are listed in Table 2.

Table 2. The values of MSE, R^2, and weight parameters.

Parameters	MSE	R^2	α_{11}	α_{12}	α_{21}	α_{22}	α_{31}	α_{32}	β_1	β_2	β_3
Values	6.2×10^{-5}	1.0	2.21	1.54	2.81	2.01	3.39	1.00	2.05	3.87	3.07
Approximation			2.2	1.5	2.8	2.0	3.4	1.0	2	4	3

From the regression results, we observe small discrepancies between the actual polynomial coefficients (or exponents) and the weight parameters. However, the parameters closely align with the polynomial coefficients and exponents when considering their approximate values. Thus, we conclude

that the designed SPNN can effectively solve fractional-order multivariate polynomial regression problems.

3.3. Application: Multi-variable regression of the maximum stress

In this part, we apply the designed SPNN to analyze the maximum stress of concrete sewage pipelines under the combined influence of traffic load, earth pressure, and groundwater level, with a consideration of 12 physical parameter variables. These variables encompass corrosion depth (Cd), corrosion width (Cw), corrosion length (Cl), void width (Vw), void length (Vl), burial depth (H), traffic load (P), pipe diameter (D), wall thickness (t), bedding modulus (Eb), backfill soil modulus (Es), and groundwater level (hw). The detailed parameter ranges for each variable are provided in Table 3.

Table 3. The parameter ranges of each physical variable.

Physical variable	Value		Physical variable	Value	
	Minimum	Maximum		Minimum	Maximum
Cd (cm)	5	90	P (MPa)	0.5	1.5
Cw (°)	0	180	D (mm)	300	1200
Cl (m)	0	10	t (mm)	40	120
Vw (°)	0	120	Eb (MPa)	6	580
Vl (m)	0	3	Es (MPa)	5	65
H (m)	0.5	3	hw/H	0.31	8.85

These physical variables are randomly generated within the specified ranges, resulting in a dataset of 250 samples. The maximum stress signals are formulated by finite element simulation based on these datasets. Figure 5 illustrates the dependence of the maximum stress on each physical variable. As the maximum stress of concrete sewage pipelines is determined by the 12 physical variables, it is challenging to analyze the relationship between individual variables and stress through simple data fitting. As shown in Figure 5, the relationship between maximum stress and each variable is nonlinear and complex, indicating that this is a multivariate nonlinear regression problem. Existing software and algorithms are unable to efficiently and accurately address this challenge; however, our designed SPNN method is capable of handling it effectively.

According to the univariate empirical formula [7, 9], we define the multivariate regression equation of the maximum stress as

$$\begin{aligned} y = {} & \alpha_1 Cd^{\beta_1} + \alpha_2 Cw^{\beta_2} + \alpha_3 Cl^{\beta_3} + \alpha_4 Vw^{\beta_4} + \alpha_5 Vl^{\beta_5} + \alpha_6 H^{\beta_6} + \alpha_7 P^{\beta_7} \\ & + \alpha_8 D^{\beta_8} + \alpha_9 t^{\beta_9} + \alpha_{10} Eb^{\beta_{10}} + \alpha_{11} Es^{\beta_{11}} + \alpha_{12} (hw/H)^{\beta_{12}}. \end{aligned} \tag{3.3}$$

The data set is first normalized to eliminate the influence of dimensionality, and then randomly split into training, validation, and test sets, with respective ratios of 64%, 20%, and 16%. It makes almost no difference to divide the data set at the other ratios, for example, 70%, 15%, and 15%, because of the excellent model performance. The regression model is developed using the explainable SPNN. During the training process, a total of 1000 epochs are set, and the Adam optimizer with with a learning rate equal to 0.006 is employed together with a one-cycle scheduler with the maximum learn rate equal to

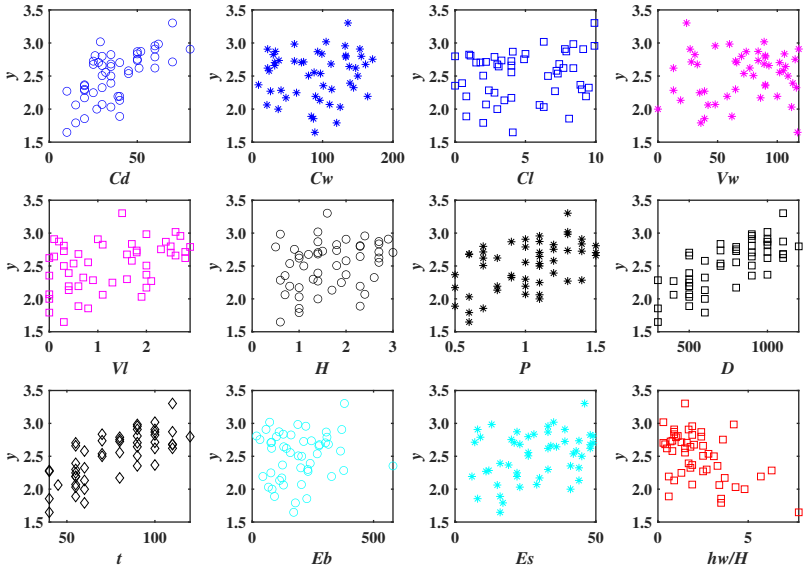
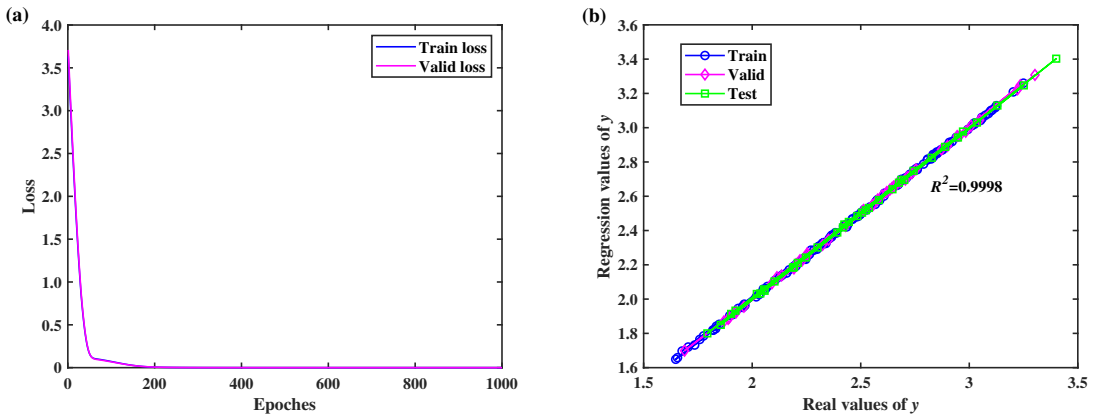


Figure 5. The dependence of maximum stress signals (y) on corrosion depth (Cd), corrosion width (Cw), corrosion length (Cl), void width (Vw), void length (Vl), burial depth (H), traffic load (P), pipe diameter (D), wall thickness (t), bedding modulus (Eb), backfill soil modulus (Es), and groundwater level over burial depth (hw/H).



(a) (b)

Figure 6. The multivariate polynomial regression of the maximum stress: (a) the training loss and valid loss depicted by MSE; (b) the plot of real values versus regression values.

0.01. Figure 6(a) shows the training loss and valid loss depicted by MSE, and Figure 6(b) presents a comparison between the real and predicted maximum stress values.

The regression error MSE = 3.36×10^{-5}, and the coefficient of determination $R^2 = 0.9998$. We save model parameters to two decimal places, and the regression equation is

$$y = 0.35Cd^{0.51} + 0.24Cw^{0.32} + 0.05Cl^{0.62} + 0.25Vw^{0.06} + 0.28Vl^{0.38} + 0.15H^{0.19} + 0.92P^{1.20} + 0.72D^{1.16} + 0.46t^{1.13} + 0.37Eb^{0.24} - 0.04Es^{0.63} - 0.04(hw/H)^{0.43}. \quad (3.4)$$

It should be noted that the values of coefficients and exponents in Eq (3.4) may exhibit minor fluctuations due to the broad feasible region of the solutions. However, the contributions of each physical variable to the maximum stress derived from Eq (3.4) can be well interpreted. It is evident that the corrosion depth, corrosion width, corrosion length, void width, void length, burial depth, traffic load, pipe diameter, wall thickness, and bedding modulus have positive effects on the maximum stress, while the backfill soil modulus and groundwater level exhibit negative effects. These findings align with existing empirical knowledge.

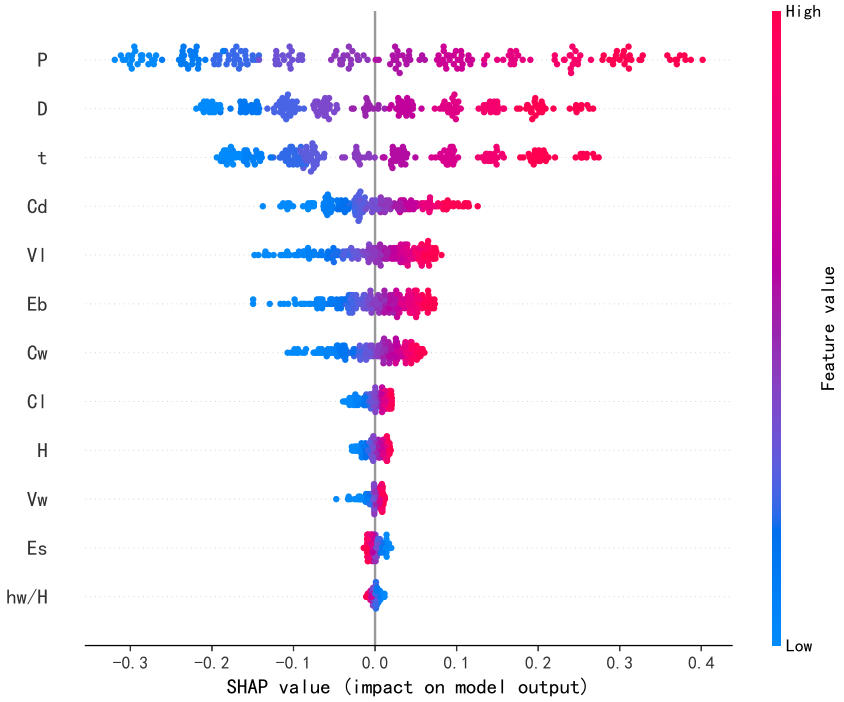


Figure 7. The feature analysis of 12 physical variables on the maximum stress by SHAP values.

Additionally, the SHAP approach [16–18] is utilized to analyze the impact of input features on the model's output. The SHAP value for each sample (x) corresponding to feature (f_i) is computed using Eq (3.5).

$$SHAP_{f_i}(x) = \sum_{f_i \in f} [|f| \times \binom{F}{|f|}]^{-1} \times [P_f(x) - P_{f\setminus f_i}(x)], \quad (3.5)$$

where f_i represents the i^{th} feature (i.e., the i^{th} physical variable), F is the total number of features, and f is a feature subset that includes f_i. $|f|$ denotes the number of elements in subset f, and $\binom{F}{|f|}$ is

the number of combinations of F items taken $|f|$ at a time. $P_f(x)$ is the predicted value when all the physical variables in f are selected, and $P_{f\setminus f_i}(x)$ is the predicted value with all the variables in f except f_i. The SHAP values for the samples related to each physical variable are shown in Figure 7.

In Figure 7, if the higher feature value corresponds to larger SHAP values, it indicates a positive effect of that feature on the model output. Conversely, if the higher feature value results in smaller SHAP values, it suggests a negative effect. Thus, we conclude that features $P, D, t, Cd, Vl, Eb, Cw, Cl, H$, and Vw have a positive effect on the maximum stress, while Es and hw/H have a negative effect. These conclusions align with the regression analysis results from Eq (3.4). Additionally, the importance ranking of the input features is $P > D > t > Cd > Vl > Eb > Cw > Cl > H > Vw > Es > hw/H$.

According to the regression results, we can further analyze the evolution of the maximum stress of concrete sewage pipelines with respect to the univariate input. By selecting a sample data point and employing the control variable method, the maximum stress signal can be calculated using Eq (3.4), with only one variable being varied. Figure 8 shows the evolution of maximum stress with respect to each physical variable. These univariate evolutionary trends provide theoretical guidance for strategies aimed at repairing and improving the properties of concrete sewage pipelines.

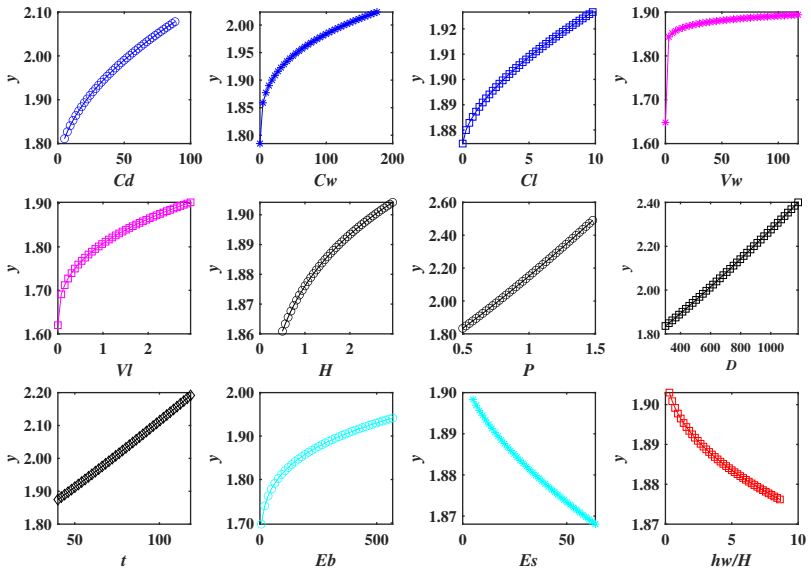


Figure 8. The evolution of maximum stress with respect to each physical variable by the control variable method.

4. Conclusions

In this manuscript, we propose an explainable sigma-pi neural network to address the multi-variable nonlinear polynomial regression problem. The coefficient and exponent parameters of polynomials are effectively represented by the corresponding weight parameters in the SPNN. To accelerate the regression process, the back-propagation algorithm is employed for parameter optimization. The examples reveal that our designed SPNN can efficiently and accurately solve both

integer-order and fractional-order multivariate polynomial regression problems. In practical application, the SPNN provides high-precision fitting of the maximum stress in concrete sewage pipelines under the combined influence of 12 physical parameter variables. Furthermore, feature importance ranking and additional analyses on the relationship between maximum stress and these variables can be conducted based on this explainable machine learning model.

Building upon the framework for solving multi-variable polynomial regression, future algorithms for addressing more complex nonlinear regression problems could be developed through higher-order neural networks, such as sigma-pi or sigma-pi-sigma networks. This study lays the theoretical foundation for developing generalized tools to solve multi-variable nonlinear regression problems.

Use of AI tools declaration

The authors declare that this manuscript is the authors' original work, and they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by the China Postdoctoral Science Foundation Funded Project (Grant No. 2022M712902), the Natural Science Foundation of Henan (Grant No. 232300421345), and the Zhongyuan Youth Top Talent Plan (Zhongyuan Youth Postdoctoral Innovative Talents).

Conflict of interest

The authors declare no conflict of interest with respect to the research, authorship, and/or publication of this article.

References

1. Clauset A, Larremore D, Sinatra R, (2017) Data-driven predictions in the science of science. *Science* 355: 477–480. https://doi.org/10.1126/science.aal4217

2. Subrahmanian VS, Kumar S, (2017) Predicting human behavior: The next frontiers. *Science* 355: 489. https://doi.org/10.1126/science.aam7032

3. Wang Z, Sun ZH, Yin H, Liu XH, Wang JL, Zhao HT, et al. (2022) Data-driven materials innovation and applications. *Adv Mater* 34: 2104113. https://doi.org/10.1002/adma.202104113

4. Chen X, Yan CC, Zhang XT, Zhang X, Dai F, Yin J, et al. (2016) Drug-target interaction prediction: Databases, web servers and computational models. *Briefings Bioinf* 17: 696–712. https://doi.org/10.1093/bib/bbv066

5. Zhao Y, Yin J, Zhang L, Zhang Y, Chen X, (2024) Drug-drug interaction prediction: Databases, web servers and computational models. *Briefings Bioinf* 25: bbad445. https://doi.org/10.1093/bib/bbad445

6. Guo XX, Sun YT, Ren JL, (2020) Low dimensional mid-term chaotic time series prediction by delay parameterized method. *Inf Sci* 516: 1–19. https://doi.org/10.1016/j.ins.2019.12.021

7. Li B, Guo XX, Fang HY, Ren JL, Yang KJ, Wang F, et al. (2020) Prediction equation for maximum stress of concrete drainage pipelines subjected to various damages and complex service conditions. *Constr Build Mater* 264: 120238. https://doi.org/10.1016/j.conbuildmat.2020.120238

8. Guo XX, Xiong NN, Wang HY, Ren JL, (2022) Design and analysis of a prediction system about influenza-like illness from the latent temporal and spatial information. *IEEE Trans Syst Man Cybern Syst* 52: 66–77. https://doi.org/10.1109/TSMC.2020.3048946

9. Li B, Fang HY, Yang KJ, Zhang XJ, Du XM, Wang NN, et al. (2022) Impact of erosion voids and internal corrosion on concrete pipes under traffic loads. *Tunnelling Underground Space Technol* 130: 104761. https://doi.org/10.1016/j.tust.2022.104761

10. Rudin W, (1976) *Principles of Mathematical Analysis*, McGraw-Hill Companies.

11. Guo XX, Han WM, Ren JL, (2023) Design of a prediction system based on the dynamical feed-forward neural network. *Sci China Inf Sci* 66: 112102. https://doi.org/10.1007/s11432-020-3402-9

12. Yu LP, Guo XX, Wang G, Sun BA, Han DX, Chen C, et al. (2022) Extracting governing system for the plastic deformation of metallic glasses using machine learning. *Sci China Phys Mech Astron* 65: 264611. https://doi.org/10.1007/s11433-021-1840-9

13. Liu Z, Wang Y, Vaidya S, Ruehle F, Halverson J, Soljacic M, et al. (2024) KAN: Kolmogorov-Arnold networks. preprint, arXiv:2404.19756. https://doi.org/10.48550/arXiv.2404.19756

14. Gurney KN, (1992) Training nets of hardware realizable sigma-pi units. *Neural Networks* 5: 289–303. https://doi.org/10.1016/S0893-6080(05)80027-9

15. Penny WD, Stonham TJ, (1995) Generalization in multi-layer networks of sigma-pi units *IEEE Trans Neural Networks* 6: 506–508. https://doi.org/10.1109/72.363490

16. Lundberg SM, Lee SI, (2017) A unified approach to interpreting model predictions. *Adv Neural Inf Process Syst* 30: 4765–4774.

17. Wu LL, Wei GY, Wang G, Wang HY, Ren JL, (2022) Creating win-wins from strength-ductility trade-off in multi-principal element alloys by machine learning. *Mater Today Commun* 32: 104010. https://doi.org/10.1016/j.mtcomm.2022.104010

18. Xiao L, Wang G, Long WM, Liaw PK, Ren JL, (2024) Fatigue life prediction of the FCC-based multi-principal element alloys via domain knowledge-based machine learning. *Eng Fract Mech* 296: 109860. https://doi.org/10.1016/j.engfracmech.2024.109860

AIMS Press

© 2024 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0)