DOI:Registering DOI

Polynomial Fitting Algorithm Based on Neural Network

Yuerong Tong[1], Lina Yu[1], Sheng Li[2], Jingyi Liu[1], Hong Qin[1] and Weijun Li[1,3,*]

[1] Institute of Semiconductors, Chinese Academy of Sciences, Beijing 100083, China
[2] School of Information and Safety Engineering, Zhongnan University of Economics and Law, Wuhan 430073, China
[3] Shenzhen DAPU Microelectronics Company Ltd., Shenzhen 518100, China
* Corresponding author: Weijun Li (wjli@semi.ac.cn)

Manuscript Revised 26 April 2021; Accepted 26 April 2021; Published 27 April 2021

Academic Editor: *Weiwei Cai*

***Abstract*: As a method of function approximation, polynomial fitting has always been the main research hotspot in mathematical modeling. In many disciplines such as computer, physics, biology, neural networks have been widely used, and most of the applications have been transformed into fitting problems using neural networks. One of the main reasons that neural networks can be widely used is that it has a certain sense of universal approximation. In order to fit the polynomial, this paper constructs a three-layer feedforward neural network, uses Taylor series as the activation function, and determines the number of hidden layer neurons according to the order of the polynomial and the dimensions of the input variables. For explicit polynomial fitting, this paper uses non-linear functions as the objective function, and compares the fitting effects under different orders of polynomials. For the fitting of implicit polynomial curves, the current popular polynomial fitting algorithms are compared and analyzed. Experiments have proved that the algorithm used in this paper is suitable for both explicit polynomial fitting and implicit polynomial fitting. The algorithm is relatively simple, practical, easy to calculate, and can efficiently achieve the fitting goal. At the same time, the computational complexity is relatively low, which has certain application value.**

Index Terms*: *Polynomial fitting, neural network, Taylor series, Implicit polynomial representations.

1. Introduction

With the development of Internet technology, various fields of scientific research are flooded with large amounts of data. For example, in the health sector will have a lot of medical diagnostic data, the financial sector will have a lot of capital flows data, the computer graphics field will generate a lot of graphical image data [1, 2]. Therefore, people are increasingly concerned about how these growing massive amounts of data effectively organize and manage, it has also become the target research experts and scholars at home and abroad to focus on.

The fitting of polynomial functions has always been the main research hotspot in mathematical modeling. Polynomial fitting generally includes explicit polynomial fitting and implicit polynomial fitting. As a method of function approximation, polynomial fitting has been applied in many fields [3, 4]. By choosing different fitting methods to fit the discrete points in the data set, the errors produced by the fitting are also different. Therefore, choosing a suitable fitting method has become the foundation of polynomial fitting research. In order to meet the needs of various fields, more and more experts and scholars have launched the research on polynomial fitting.

Explicit polynomial fitting often uses Taylor series [5, 6]. Taylor series are often used in the field of mathematics, especially in the research of approximate calculations. In mathematics, in order to realize the expression of a specific function, a series of terms is obtained by deriving the function at a certain point, and the Taylor series is obtained by adding these terms. In recent years, implicit polynomial curves have been widely used in computer graphics [7, 8], computer vision [9, 10], time series [11, 12] and so on. The implicit polynomial curve can realize the description of irregular objects with few parameters, and it has a clear analytical formula and is easy to use [13]. For implicit polynomial fitting, iterative fitting algorithms are often used in the early days. But the iterative fitting algorithm has a common disadvantage that the polynomials generated are not stable. When fitting complex object contours, the iterative fitting algorithm will cause a large time cost due to the large amount of calculation. Therefore, the early iterative fitting algorithm is only suitable for fitting the contours of simple objects. In recent years, related research scholars have also carried out further research on this [14, 15]. At present, some mainstream fitting algorithms such as the

general linear least squares fitting algorithm [16] and the 3L curve fitting algorithm [17] based on the improved linear least squares method. Compared with the earlier iterative fitting algorithms, these algorithms have better performance in fitting the contours of complex objects, and the time overhead in the calculation process is smaller.

For traditional polynomial fitting methods, the theoretical analysis is very rigorous, but the theoretical derivation is more complicated. Some algorithms derived from this have higher requirements for raw data and poor adaptability. Neural networks have been widely used in many disciplines [18, 19]. The application of neural networks to polynomial fitting has advantages different from traditional methods.

The main contributions of this article are as follows:

1) This paper constructs the neural network topology structure based on the polynomial fitting model, uses Taylor series as the activation function of the network, and uses the original experimental data samples to train the model parameters to obtain the optimal fitting parameters.

2) This paper uses the gradient backpropagation algorithm, which can effectively solve the coefficients of the polynomial function without complicated formula derivation. To a certain extent, the complexity of polynomial fitting is reduced.

3) This paper has carried out experimental verification on explicit polynomial fitting and implicit polynomial fitting, and compared the current popular polynomial fitting algorithms to prove the effectiveness of the algorithm.

2. Related Work

In mathematics, the algebraic expression formed by adding several monomials is called a polynomial, which can be recorded as $f(x) = a_0x^0 + a_1x^1 + a_2x^2 + \cdots + a_nx^n$, where $a_0, a_1, \cdots, a_n$ is the coefficient of the polynomial. Each monomial is the term of the polynomial, and the highest degree is called the order of the polynomial. Polynomial fitting uses a polynomial expansion to fit discrete data points. It does not require the polynomial curve to pass through all data points. The purpose of polynomial fitting is to find a set of $a_0, a_1, \cdots, a_n$ so that the polynomial curve matches the actual sample data as much as possible.

2.1 Explicit Polynomial Fitting

Explicit polynomial fitting often uses Taylor series. Take a univariate function as an example. Taylor's theorem: Let n be a positive integer. Let n be a positive integer. If a function $f(x)$ defined on an interval containing a is differentiable $n+1$ times at the point a, then for any x on this interval, there is:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + r_n(x) \quad (1)$$

Where $r_n(x-a)$ is the remainder.

Taylor's theorem describes a differentiable function. At a certain point, the derivative of each order of the function is obtained, and the derivative value is used as the coefficient of the polynomial, and the polynomial is used to approximate the function value of the function in the neighborhood of this point. From this we know that Taylor's formula can use polynomials to approximate the situation around a certain point. Therefore, Taylor series are often used for explicit polynomial fitting.

2.2 Implicit Polynomial Fitting

Implicit polynomial curve is a special kind of algebraic curve, which has precise expressive ability, has clear analytical formula and is therefore very easy to use, and can describe the outline of data point collection well. Therefore, the implicit polynomial fitting algorithm has become the key research direction of related researchers, and the research on it has a realistic background basis. A binary implicit polynomial function is shown in formula (2).

$$f(x,y) = \sum_{0\le i,j\le d;0\le i+j\le d}' a_{ij}x^iy^j = 0 \quad (2)$$

The research results show that the zero set composed of binary implicit polynomials can represent arbitrarily complex curves or surfaces in two-dimensional space. An implicit polynomial function is usually written as:

$$\begin{aligned} f_k(x,y) = \sum_{i+j\le n;i,j\ge 0} a_{ij}x^iy^j \\ = a_{00} + a_{10}x + a_{01}y + a_{20}x^2 \\ + a_{11}xy + a_{02}y^2 + \cdots + a_{k0}x^k \\ + a_{k-1,1}x^{k-1}y + \cdots + a_{0k}y^k = 0 \end{aligned} \quad (3)$$

Formula (3) is expressed in vector form,

$$f_k(x,y) = A^TX \quad (4)$$

In formula (4), $A^T = (a_{00} \quad a_{10} \quad \cdots \quad a_{1,k-1} \quad a_{0k})$ is the coefficient vector, a_{ij} represents the coefficient of x^iy^j, $X^T = (1 \quad x \quad \cdots \quad x^{k-1}y \quad y^k)$ is a vector of items.

Use implicit polynomial fitting. When the contour of the fitted object is more complicated, if you choose a low-order implicit polynomial, the curve obtained by fitting cannot well reflect the edge characteristics of the contour of the object, and the fitting effect is not good; The contour of the object is relatively simple. If you choose a higher-order implicit polynomial, it will not only greatly increase the time required in the fitting process, but also make the obtained implicit polynomial more complicated and difficult to handle, and it will be more difficult in the future application process.

The correct choice of the order of the implicit polynomial is extremely important in the process of fitting the contour of the object. There have been many research scholars on how to determine the fitting order of implicit polynomials. The research results point out that when the order of the implicit polynomial is even, the zero set can be bounded or unbounded; when the implicit polynomial is an even number, the zero set can be bounded or unbounded; When the order of is odd, the zero set is always unbounded. Therefore, even order is mostly used in implicit polynomial fitting.

3. Methodology

3.1 Number of Hidden Layer Nodes

Choosing the right number of hidden layer nodes is very important for neural networks It not only has a great impact on the performance of the established neural network model, but also is the direct cause of overfitting during training. The number of hidden layer nodes has a great influence on the iteration speed of the neural network. The fewer hidden layer nodes, the faster the iteration speed of the neural network. However, if the number of hidden layer nodes is too small, the neural network cannot achieve the effect of learning and approximation, and the network performance is very poor; if the number of hidden layer nodes is too large, the neural network may appear overfitting, etc. Undesirable phenomena will also make the hardware implementation and software calculation of the neural network more complicated. Under normal circumstances, it is necessary for the experimenter to obtain the most suitable number of hidden layer nodes of the neural network based on experience and a large amount of experimental exploration. In order to avoid the phenomenon of overfitting in the training process of the neural network as much as possible, on the basis of correctly satisfying the input-output relationship, the number of hidden layer nodes is selected as few as possible.

Because this paper uses neural network to fit polynomial functions, according to the structural characteristics of polynomials, we can accurately select the number of hidden layer nodes in the process of specific experiments, instead of selecting the number of hidden layer nodes through continuous exploration. In other words, if the number of inputs s is known, the number of hidden layer nodes can be accurately represented by the polynomial order r. That is, when the number of inputs is 1, the number of hidden layer nodes required is $r+1$, and when the number of inputs is greater than 1, the number of hidden layer nodes in the neural network is $r(r+3)/2$.

3.2 Neural Network Model

Kolmogorov's theorem is the mathematical foundation of neural networks [20]. According to Kolmogorov's theorem, it can be known that given any continuous function, the function can be accurately implemented with a three-layer feedforward network. Under the condition that the number of hidden layer nodes can be selected arbitrarily, the three-layer neural network can approximate any continuous function with arbitrary precision under the action of the activation function.

Series refers to the form and sum of a finite or infinite sequence. According to the definition of series, a neural network can be regarded as the form and sum of a series of neuron activation functions. The process of neural network training is to gradually correct the series coefficients of these activation functions. Taylor's theorem states that for a function that is differentiable and smooth enough, the derivative values of a certain point can be used as the coefficients of the polynomial function. Then use this coefficient to approximate the value of the function in the neighborhood of this point. In the experiment, the zero point is selected as the expansion point of the Taylor series, and the weight of the neuron corresponds to the Taylor coefficient.

In this paper, a three-layer feedforward neural network is used to create a Taylor series neural network model. Take the Taylor series of the unary function as an example. In order to describe the finite Taylor series, considering the sum of the first $n+1$ terms, we can get formula (5).

$$f(x) = f(a) + f^{'}(a)(x-a) + \frac{f^{''}(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{5}$$

According to formula (5), we can get the neural network model as shown in Figure 1.

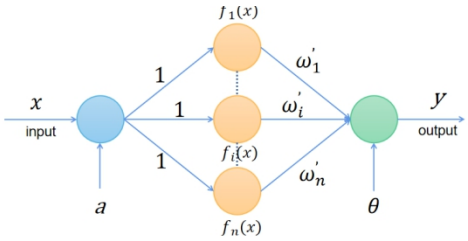

Fig.1. Neural network model diagram.

The bias of each neuron in the hidden layer of the neural network is 0, and the weight between the neurons in the hidden layer and the neurons in the input layer is set to a constant: $\omega_i = 1(i = 1,2,\cdots,n)$.

For each neuron in the hidden layer of the neural network, its output is shown in formula (6).

$$f_i(x) = \omega_i(x-a)^i \tag{6}$$

The output of the neural network is shown in formula (7).

$$\begin{aligned} y &= \sum_{i=1}^{n} \omega_i^{'} f_i(x) + \theta \\ &= \omega_n^{'}(x-a)^n + \cdots + \omega_2^{'}(x-a)^2 + \omega_1(x-a) + \theta \end{aligned} \tag{7}$$

We can get the first $n+1$ coefficient of Taylor expansion, and get the weight $\omega_i^{'}$ and bias θ by training the neural network.

$$\begin{aligned} \omega_1 &= f^{'}(a) \\ \omega_2^{'} &= \frac{f^{''}(a)}{2} \\ &\cdots \\ \omega_n^{'} &= \frac{f^{(n)}(a)}{n!} \\ \theta &= f(a) \end{aligned} \tag{8}$$

3.3 Data Preprocessing

When using a polynomial to fit the contour of an object, if the distance between the point on the given original data set and the origin of the coordinate is too large, the polynomial coefficient will increase exponentially. Not only greatly increases the calculation time of the algorithm, but also inconvenient for future applications. Therefore, we must first normalize the data. The data points are concentrated near the origin of the coordinates, which is conducive to the subsequent fitting.

In the process of neural network training, if the amount of training data is small, the neural network may learn some

irrelevant features due to its strong fitting ability, but the characteristics of the data itself cannot be learned. Therefore, it is necessary to perform enhancement operations on the training data. The data enhancement operation can expand the original data set by introducing some transformations without substantially increasing the data. In this way, limited data can generate value equivalent to more data.

In order to improve the fitting ability of the neural network training model, the data is scaled equally using the idea similar to the 3L algorithm. We call this operation 3L scaling. First multiply the normalized original data by a scaling factor $c\ (c > 0)$, and use the scaling factor c as the value of the implicit polynomial function. The specific operation is shown in formula (9).

$$\begin{matrix} f((1-c)x,(1-c)y) = c \\ f(x,y) = 0 \\ f((1+c)x,(1+c)y) = -c \end{matrix} \quad (9)$$

The 3L scaling is different from the D-Euclidean distance transformation in the 3L algorithm. But in the experiment, this method can achieve similar experimental results, which also simplifies the tedious and complicated calculation process in the 3L algorithm.

Let the scaling factor $c = 0.05$, the zoomed effect diagram is shown in the figure 3.

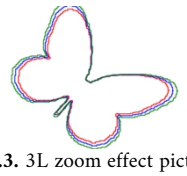

Fig.3. 3L zoom effect picture.

4. Experiments

4.1 Datasets

For explicit polynomial fitting, a non-linear function is often selected as the objective function. For the description and fitting of object contours, implicit polynomial curves have natural advantages compared with explicit polynomial curves. Implicit polynomial fitting mostly uses the target in Figure 4 as the fitting object, and there is no public data set for the two-dimensional target curve.

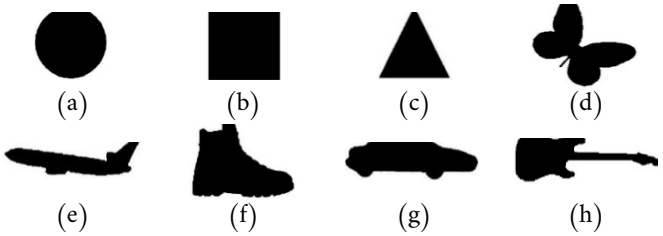

Fig.4. Object outline.

After detecting the edge of the original image by using edge detection technology, sampling is performed to obtain the sample points as shown in Figure 5.

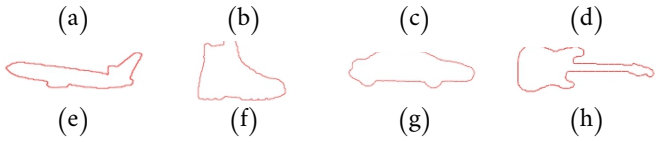

Fig.5. Data points obtained by edge detection.

4.2 Comparison of the Results of the Experiment

4.2.1 Explicit Polynomial Fitting

For explicit polynomial fitting, one-variable polynomial is the simplest and most basic form of the approximated function. Take function $f(x) = \sin(x)$ as an example to study the effect of the algorithm used in this paper on the fitting effect under the same number of iterations and different orders of polynomial selection. The learning rate in neural network training is set to 0.03, and the number of iterations of the neural network is 5000 times. The fitting results of the polynomial order from the second to the fifth order and the corresponding loss function are shown in Figure 6.

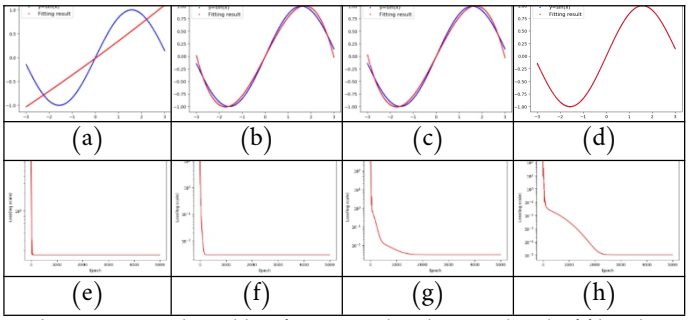

Fig.6. Fitting results and loss function under the second to the fifth order.

Using the same method for function $f(x) = e^x$, $f(x) = \sin(x)$, $f(x) = \cos(x)$, and fit them separately in the presence or absence of noise, and all the result curves obtained are shown in Figure 7.

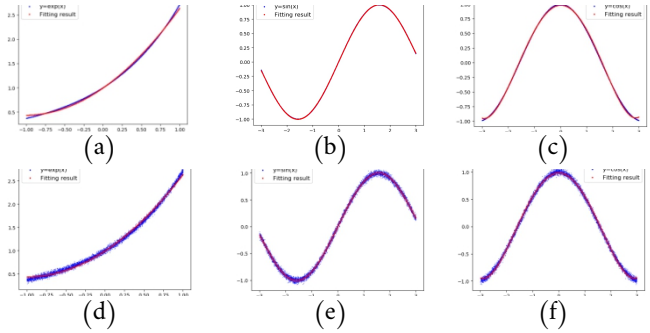

Fig.7. Fitting results in the presence or absence of noise.

Then we perform a binary explicit polynomial fitting on the saddle surface, the conical surface and the Gaussian surface. The formula for the saddle surface is shown in formula (10).

$$\frac{x^{\angle}}{a} - \frac{y^{\angle}}{b} = 2px \quad (10)$$

The modeling effect of 5000 data points of the saddle surface under random sampling conditions is shown in Figure 8. In the experiment, the saddle surface was fitted under the second and third order respectively. The learning rate is 0.03,

and the number of iterations is 1000. The specific parameter settings in the experiment are as follows:$a = 3$, $b = 3$, $p = \frac{1}{2}$.

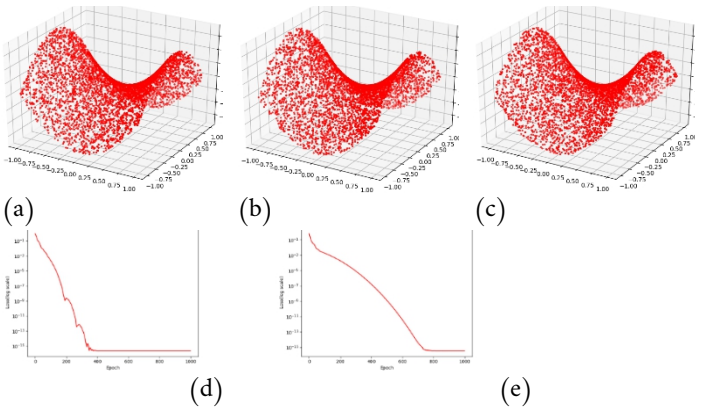

Fig.8. Saddle surface, fitting results and loss function under the second and third order.

The formula for conical surface is shown in formula (11).

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 0 \tag{11}$$

Model 5000 data points of the conical surface under random sampling conditions. In the experiment, the conical surface is fitted under the third and fourth order respectively. The learning rate is 0.03, and the number of iterations is 1000.

The specific parameter settings in the experiment are as follows: $a = \sqrt{3}$, $b = \sqrt{3}$, $c = 1$. Take the data points of the conical surface in the positive direction of the vertical axis of the coordinate axis to conduct the experiment. Figure 9 shows the fitting results and loss function of the conical surface under different orders.

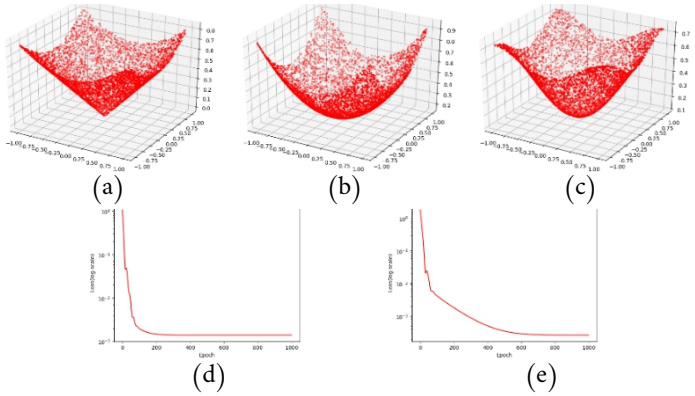

Fig.9. Conical surface, fitting results and loss function under the third and fourth order.

The formula of Gaussian surface is shown in formula (12).

$$N(\bar{x}|\bar{u}, \Sigma) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} exp\left[-\frac{1}{2}(\bar{x} - \bar{u})^T \Sigma^{-1} (\bar{x} - \bar{u})\right] \tag{12}$$

Model 5000 data points of the Gaussian surface under random sampling conditions. The specific parameters of the Gaussian surface in the experiment are set as follows: $\mu = [0.5, 0.1]$, $\lambda = [0.8, 0.08]$, $\Sigma = \begin{bmatrix} 0.0 & 1.5 \\ 0.0 & 0.0 \end{bmatrix}$, $w_1 = 0.008$, $w_2 = 10$. In the experiment, the Gaussian surface is fitted in the second to sixth order. The learning rate is 0.05, and the number of iterations is 6000, 6000, 6000, 10000, and 10000 respectively. The fitting results and loss function under different orders are shown in Figure 10.

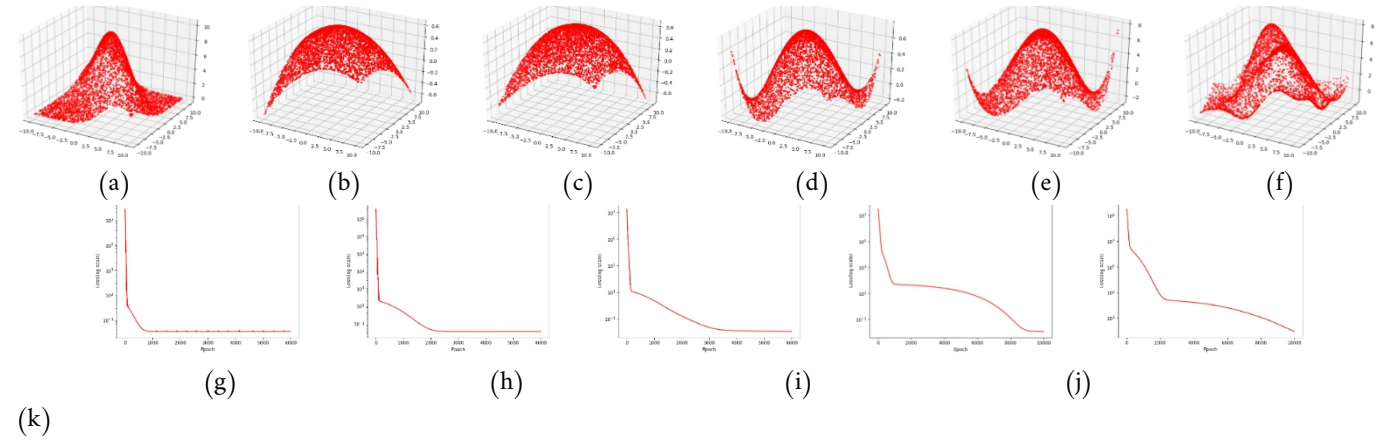

Fig.10. Gaussian surface , fitting results and loss function under the second to sixth order.

4.2.2 Implicit polynomial fitting

For implicit polynomial fitting, in order to study the influence of the polynomial order on the fitting effect and find the most suitable order for fitting the target curve, we take the butterfly curve as an example to conduct experiments.

The research results point out that when the order of the implicit polynomial is even, the zero set can be bounded or unbounded; when the order is odd, the zero set is always unbounded. In an ideal situation, as the order of the polynomial increases, the polynomial fitting ability will be stronger. Therefore, in the specific experiment process, even-order polynomials are used to fit the target curve. The butterfly curve is fitted under the second, fourth, and sixth order. The fitting results and loss function are shown in Figure 11. The red curve is the original data points, and the blue curve is the fitting result.

By comparing the fitting results of the butterfly curve under different orders, we can find that as the order of the polynomial increases, the fitting effect of the polynomial is

better. Although the sixth-order polynomial has better fitting ability than the fourth-order polynomial, it can be seen from the fitting results in Figure 11 that the higher-order terms of the polynomial will amplify the unsmooth interference points and affect the final fitting result. Next, we explore the fitting effect of the polynomial fitting algorithm based on neural network, 3L curve fitting algorithm and general linear least squares fitting algorithm under the fourth order. The results are shown in the figure 12.

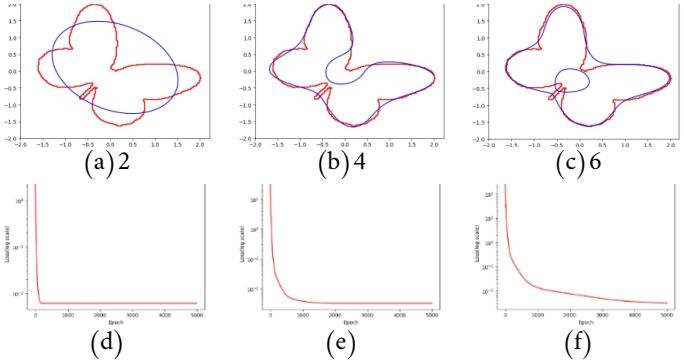

Fig.11. Fitting results and loss function under the second, fourth, and sixth order.

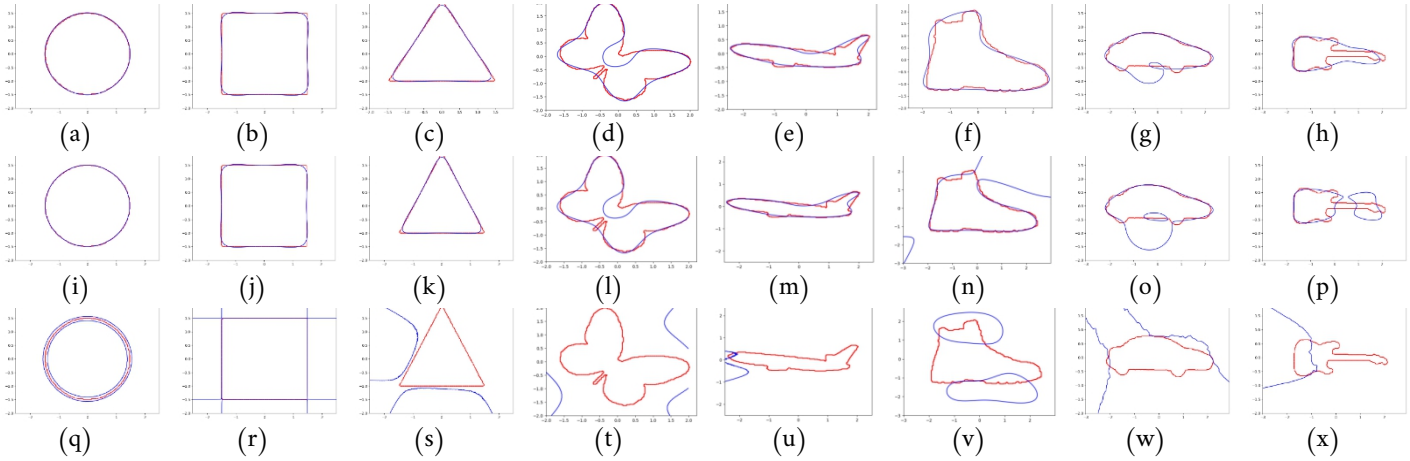

Fig.12. Fitting results of eight kinds of curves using polynomial fitting algorithm based on neural network, 3L curve fitting algorithm, general linear least squares fitting algorithm under fourth-order.

Through experiments, it can be found that the resulting curves obtained by the general linear least squares fitting algorithm under the fourth-order polynomial cannot be very close to the original data. Therefore, the general linear least squares fitting algorithm is not suitable for complex curve fitting.

For most curves, the difference between the fitting results of the polynomial fitting algorithm based on neural network and the 3L curve fitting algorithm is little. But for the fitting of shoes and guitar curves, the polynomial fitting algorithm based on neural network can fit the closed The curve, and the fitting curve obtained by using the 3L curve fitting algorithm is open and incomplete.

The following explores the effect of the polynomial fitting algorithm based on neural network and the 3L fitting algorithm in the presence of noise. Taking the butterfly curve as an example, add 20 noise points, 40 noise points, and random disturbances to the original butterfly data points. The fitting results are shown in Figure 13, where the first line uses the results obtained from the neural network-based polynomial fitting algorithm, and the second line uses the results obtained from the 3L curve fitting algorithm.

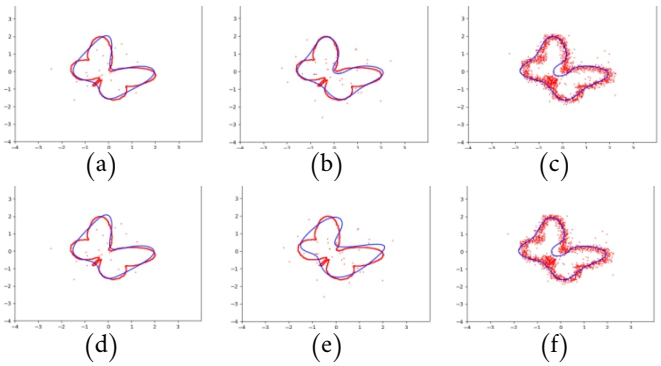

Fig.13. The result of fitting the butterfly curve after adding noise and random disturbance under the polynomial fitting algorithm based on neural network and the 3L curve fitting algorithm.

5. Conclusion

In order to fit the polynomial, the neural network-based polynomial fitting algorithm used in this paper constructs a three-layer feedforward artificial neural network, and uses Taylor series as the activation function of the network. For explicit polynomial fitting, this paper uses a variety of non-linear functions as the objective function, and compares the fitting effects under different orders of polynomials. For the fitting of implicit polynomial curve, it can be found through experiments that the general linear least squares fitting

algorithm has a particularly poor fitting effect. Under the condition of the original data, the fitting results of the algorithm used in this paper and the 3L curve fitting algorithm are very similar from a visual point of view, but by adding random noise, it can be found that the fitting result of the 3L curve fitting algorithm is not stable enough. Moreover, in the specific experimental process, inappropriate selection of the zoom factor of the original data set and the new data set will cause the polynomial curve to oscillate, making the fitting effect unsatisfactory, especially in the fitting of high-order polynomial curves. Therefore, it can be proved that the neural network-based polynomial fitting algorithm used in this paper is suitable for implicit polynomial fitting, and has good robustness, and can efficiently achieve the fitting goal, while the computational complexity is low.

Acknowledgement

This work is supported by the Key-Area Research and Development Program of Guangdong Province (No.2019B010107001).

References

[1] Ning, X., Li, W., Tang, B., & He, H. (2018). BULDP: biomimetic uncorrelated locality discriminant projection for feature extraction in face recognition. *IEEE Transactions on Image Processing*, 27(5), 2575-2586.

[2] Ning, X., Li, W., & Liu, W. (2017). A fast single image haze removal method based on human retina property. *IEICE TRANSACTIONS on Information and Systems*, 100(1), 211-214.

[3] Rogers, S. S., Waigh, T. A., Zhao, X., & Lu, J. R. (2007). Precise particle tracking against a complicated background: polynomial fitting with Gaussian weight. *Physical Biology*, 4(3), 220.

[4] Lu, Y. H., & Lu, W. K. (2009). Edge-preserving polynomial fitting method to suppress random seismic noise. *Geophysics*, 74(4), V69-V73.

[5] Chen, X., Ma, Q., & Alkharobi, T. (2009, August). New neural networks based on Taylor series and their research. In 2009 2*nd IEEE International Conference on Computer Science and Information Technology* (pp. 291-294).

[6] Hu, Z., Wu, H., & Zhu, S. (2011, September). An Engineering Solution to Taylor Series Expansion Coefficients Based on BP Neural Network. In 2011 *International Conference on Internet Computing and Information Services* (pp. 491-494).

[7] Sederberg, T. W., Anderson, D. C., & Goldman, R. N. (1984). Implicit representation of parametric curves and surfaces. *Computer Vision, Graphics, and Image Processing*, 28(1), 72-84.

[8] Bajaj, C. L., & Ihm, I. (1992). Smoothing polyhedra using implicit algebraic splines. *ACM SIGGRAPH Computer Graphics*, 26(2), 79-88.

[9] Biswas, S., Ghoshal, D., & Hazra, R. (2016). A new algorithm of image segmentation using curve fitting based higher order polynomial smoothing. *Optik*, 127(20), 8916-8925.

[10] Magu, G., Lucaciu, R., & Isar, A. (2021). Improving the Targets' Trajectories Estimated by an Automotive RADAR Sensor Using Polynomial Fitting. *Applied Sciences*, 11(1), 361.

[11] Wu, G., & Yang, J. (2013). A representation of time series based on implicit polynomial curve. *Pattern Recognition Letters*, 34(4), 361-371.

[12] Briggs, K. (1990). An improved method for estimating Liapunov exponents of chaotic time series. *Physics Letters A*, 151(1-2), 27-32.

[13] Keren, D., Cooper, D., & Subrahmonia, J. (1994). Describing complicated objects by implicit polynomials. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1), 38-53.

[14] Helzer, A., Barzohar, M., & Malah, D. (2004). Stable fitting of 2D curves and 3D surfaces by implicit polynomials. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10), 1283-1294.

[15] Wang, G., Li, W., Zhang, L., Sun, L., Chen, P., Yu, L., & Ning, X. (2021). Encoder-X: Solving Unknown Coefficients Automatically in Polynomial Fitting by Using an Autoencoder. *IEEE Transactions on Neural Networks and Learning Systems*.

[16] Bajaj, C., Ihm, I., & Warren, J. (1993). Higher-order interpolation and least-squares approximation using implicit algebraic surfaces. *ACM Transactions on Graphics (TOG)*, 12(4), 327-347.

[17] Blane, M. M., Lei, Z., Civi, H., & Cooper, D. B. (2000). The 3L algorithm for fitting implicit polynomial curves and surfaces to data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3), 298-313.

[18] Ning, X., Nan, F., Xu, S., Yu, L., & Zhang, L. (2020). Multi-view frontal face image generation: A survey. *Concurrency and Computation*: *Practice and Experience*, e6147.

[19] Ning, X., Li, W., & Xu, J. (2018). The principle of homology continuity and geometrical covering learning for pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 32(12), 1850042.

[20] Kůrková, V. (1992). Kolmogorov's theorem and multilayer neural networks. *Neural networks*, 5(3), 501-506.

Biographies

Yuerong Tong received the B.E. degree from School of Information and Safety Engineering, Zhongnan University of Economics and Law, Wuhan, China. She is currently pursuing the master's degree with the Institute of Semiconductors, Chinese Academy of Sciences, Beijing. Her research focuses on time series modeling.

Lina Yu received his Ph.D. in 2008 from the School of Computer Science, Huazhong University of Science and Technology. He is currently a teacher of the School of Information and Safety Engineering, Zhongnan University of Economics and Law. His researches focus on information retrieval and information security.

Sheng Li is a professor of College of Information and Electrical Engineering, China Agricultural University. His research interests include the informationization of new rural areas, intelligence agriculture, and the service for rural comprehensive information.

Jingyi Liu PhD candidate, College of Information Science and Engineering, China Agricultural University. Her research interest: Agricultural informationization and application.

Hong Qin received her Ph.D. from Institute of Semiconductors, Chinese Academy of Sciences in 2011. She is currently a senior engineer at Institute of Semiconductors Chinese Academy of Sciences. Her research interests include deep modeling, machine learning, computer and vision.

Weijun Li received his Ph.D. in 2004 from Institute of Semiconductors, Chinese Academy of Sciences. He is currently a Professor of Artificial Intelligence at Institute of Semiconductors Chinese Academy of Sciences (ISCAS) and the University of Chinese Academy of Sciences. He is in charge of the Artificial intelligence research Center of ISCAS, also the Director of the Lab of Highspeed Circuits Neural Networks of ISCAS. His research interests include deep modeling, machine art, pattern recognition, artificial neural networks and intelligent system. He is a senior member of IEEE.