



ELM Optimization Based on Hidden Layer Output Matrix

by

Haoyi Sun

Supervisor: Dr Andrew Baker

Wales Institute of Science and Art
School of Applied Computing
MSc Applied Computing

November 2020

1. ABSTRACT

Abstract: Aiming at the problem of the error from the hidden layer to the output layer in the Extreme Learning Machine (ELM), it is found through analysis that the error comes from the process of solving the Moore-Penrose generalized inverse matrix $H^\dagger$ of the hidden layer output matrix H , that indicates the matrix $H^\dagger H$ is deviated from the identity matrix. The appropriate output matrix H can be selected according to the degree of deviation to obtain a smaller training error. According to the definition of the generalized inverse matrix and Auxiliary matrix. Firstly, the target matrix $H^\dagger H$ and the error index L21-Norm are determined. Second, the experimental analysis shows that the L21-Norm of $H^\dagger H$ is linearly related to the ELM error. Finally, Gaussian filtering is introduced to reduce the noise of the target matrix, which effectively reduces the L21-Norm of the target matrix, and at the same time reduces the ELM error, and achieves the purpose of optimizing the error of the ELM algorithm.

Keywords: Extreme Learning Machine; Moore-Penrose generalized inverse matrix; L21-Norm; linear correlation; Gaussian filtering

2. CONTENTS PAGE

1. ABSTRACT.....	2
2. CONTENTS PAGE	3
3. CHAPTER 1: INTRODUCTION.....	5
4. CHAPTER 2: LITERATURE REVIEW.....	6
5. CHAPTER 3: RESEARCH AND/OR DEVELOPMENT METHODOLOGY	10
6. CHAPTER 4: DESIGN AND IMPLEMENTATION.....	12
7. CHAPTER 5: ANALYSIS AND RESULTS DISCUSSION.....	16
8. CHAPTER 6: CONCLUSIONS AND RECOMMENDATIONS.....	26
9. CHAPTER 7: REFLECTION.....	27
10. REFERENCES	27
11. APPENDICES	30

List of Tables

Table 1 Training sets generate table.....	18
Table 2 List of activation functions.....	18
Table 3 relation of neuron node number and error.....	19
Table 4 Correlation analysis of errors with the L1-Norm of the objective function.....	20
Table 5 24 sets of comparative experimental data for the six paradigm indicators with the objective function $H^{\dagger}H$	23
Table 6 24 sets of comparative experimental data for the six paradigm indicators with the objective function $(H^{\dagger}H)^3$	24
Table 7 24 sets of comparative experimental data for the six paradigm indicators with the objective function $(H^{\dagger}H)^7$	25
Table 8 Correlation analysis of errors with the L1-Norm of 3 target matrices.....	26

List of Diagrams

Fig.1 ELM network structure diagram.....	13
Fig.2 ELM optimization flow diagram.....	16
Fig.3 The numerical three-dimensional diagram of $H^{\dagger}H$ corresponding to different nodes (50, 100, 200, 499).....	19
Fig.4 Diagram of multi-objective function and multi-index.....	20
Fig.5 Diagram of 9 alternative target matrices L1-norm and algorithm error.....	21
Fig.6 Diagrams of before and after filtering error, L21-norm and matrix L21-norm with error (negative correlation).....	26
Fig.7 Diagrams of before and after filtering error, L21-norm and matrix L21-norm with error (positive correlation).....	27

3. CHAPTER 1: INTRODUCTION

- My undergraduate and graduate students in China are both majors in computational mathematics, which is an interdisciplinary subject of mathematics and computer. I am also very interested in machine learning algorithms, so I have made new discoveries about Extreme Learning Machine (ELM) in my learning research. Based on this idea, I began to design experiments for ELM research.
- The Chinese version of this MSc Project has been submitted to a Chinese journal called "Computer Application" and is under review.

With the continuous development and application of big data, in recent years, machine learning has been a hot direction for data mining and modeling. Over-limit learning machine has been proposed as a single hidden layer feedforward neural network algorithm. It has been a one-time algorithm. It has the advantages of simple model structure, fast running speed, good generalization performance, etc. and is highly valued. The over-limit learning machine has also been continuously improved and optimized and is widely used in industrial, mechanical, financial and economic fields. Aiming at the over-limit learning machine algorithm. The error comes from the hidden layer output matrix H . It is planned to reduce the algorithm error by optimizing the error of the output matrix H , while considering the robustness and generalization ability.

Since the ELM algorithm was proposed, it has simple structure, fast learning speed and good generalization performance. A large amount of research on algorithm improvement followed, mainly around the error, generalization and stability of the over-limit learning machine to improve, including the preprocessing of the training data, the determination of the input layer weight and bias, and the hidden layer neural The number of elements and their explicit and implicit expression, and the choice of activation function should be improved. Based on the findings during the operation of the ELM algorithm, it is based on theory. If the expected results can be achieved through experiments, it will further improve the accuracy and generalization of ELM. More research applications can be carried out based on the improved ELM in this article.

The goal is to find that the ELM error comes from the process of solving the Moore-Penrose generalized inverse matrix of the hidden layer output matrix according to the analysis, and to determine the target matrix and error indicators. Secondly, the experimental analysis shows that the L21 norm of the target matrix is significantly linearly related to the ELM error. Finally, the Gaussian filter is introduced to reduce the noise of the target matrix. By effectively reducing the L21 norm of the target matrix, while reducing the ELM error, the ELM algorithm error is optimized.

4. CHAPTER 2: LITERATURE REVIEW

The Extreme Learning Machine (ELM) is a new single-hidden layer feedforward neural network (single-hidden layer feedforward neural networks, SLFNs) proposed by Huang Guangbin, a professor at Nanyang Technological University in Singapore.[1], Ultra-Limited Learning The network model of the machine is divided into three layers, namely the input layer, the hidden layer and the output layer. The input layer realizes the function of accepting the input variables of the external environment; the activation function in the hidden layer is mainly used to realize the functions of calculation and recognition; output layer is used to output the result. The over-limit learning machine theoretically proves that when the hidden layer activation function of SLFNs is infinitely differentiable, its learning ability has nothing to do with the selection of input weights and biases, that is, the input layer weights can be randomly selected and bias [2].

The over-limit learning machine is a one-time completion algorithm, which can achieve good generalization performance at a very fast learning speed, thereby solving the limitation of the slow learning speed of traditional neural networks and broadening the application range of the over-limit learning machine [3]. The research progress of ELM can be seen from these aspects:

(1) There are improvements to ELM in terms of classification and regression, in order to improve the stability of the output weight of the solution (essentially, this step is equivalent to solving (regularized) least squares problem or ridge regression). An improved ELM solution the method of output weight stability is to find high-quality feature maps in the initial stage. Wang proved in (A study on effectiveness of extreme learning machine) that for certain activation functions (such as RBF functions) [4], there will always be inputs The weight makes the mapping matrix H belong to full column rank or full row rank. Then, an effective input weight selection algorithm is proposed to replace the random feature mapping in ELM, thereby improving the stability of the output weight solution. Later, Chen said This algorithm is extended to ELM (A modified extreme learning machine with sigmoidal activation functions) [5]. In Yuan (Optimization approximation solution for regression problem based on extreme learning machine), the conditions based on H are different the way to solve the output weight: column full rank, row full rank, column and row are not full rank. In this way, compared with the traditional ELM, the output weight can be calculated in a more stable way [6].

(2) In order to improve the compactness of ELM, one of the ideas is to train ELM in a dynamic way, that is, grow, trim or replace hidden neurons during the training process. In the incremental ELM (I-ELM) (Convex incremental extreme learning machine), Huang & Chen proposed that newly added hidden neurons can be selected from the candidate pool, and only appropriate neurons can be added [7]. In 2012, Yang proposed in (Bidirectional extreme learning machine for regression problem and its learning effectiveness) Two-way ELM (B-ELM)

fast incremental ELM to reduce the network size of traditional ELM [8]. In 2010, Miche proposed trimming ELM (P-ELM) (OPELM: Optimally pruned extreme learning machine), using traditional ELM establishes an initial network, and then those hidden neurons that contribute less to the training performance will be deleted [9]. In 2012, Zhang proposed the adaptive ELM (A-ELM) (Universal approximation of extreme learning machine with adaptive growth of hidden nodes), the size of the hidden layer may increase, decrease or remain unchanged in any step of the training process [10]. Subsequently, the two-stage ELM algorithm proposed by Deng et al. (Fast automatic two-stage nonlinear model identification based on the extreme learning machine) is to integrate ELM and leave-one-out (LOO) cross-validation with a step-by-step construction process, which can automatically determine the size of the network and improve the compactness of the model built by ELM [11]. Wang in 2014 (Parsimonious Extreme Learning Machine Using Recursive Orthogonal Least Squares) used the recursive orthogonal least squares method to forward selection and reverse elimination of hidden neurons in ELM, thereby obtaining the simplified structure [12]. In the same year, Luo et al. (Sparse Bayesian Extreme Learning Machine for Multi-classification) proposed A sparse Bayesian method for learning compact ELM models is proposed. Instead of subjectively adding or removing hidden neurons in most existing sparse ELMs, the method uses a hypothetical prior distribution to automatically adjust most of the output weights. Is zero, thereby improving the sparsity and high generalization performance [13]. In 2012 (Efficient and effective algorithms for training single hidden-layer neural networks), Yu and Deng proposed to reduce the parameters of the θ hidden layer by gradient descent. The size of the small ELM network [14]. In 2014, Bai et al. proposed a sparse ELM (S-ELM, Sparse extreme learning machine for classification) instead of the inequality constraint of the traditional ELM model, which greatly reduces the storage Space and test time. Similar to support vector machines, S-ELM with inequality constraints can also cause quadratic programming problems. Since the hidden layer of ELM does not need to consider bias terms, S-ELM is more efficient in training than SVM [15] . In Gastaldo et al. in 2013 (Combining ELM with random projections), ELM was used in combination with random projection (RP) to reduce the number of hidden neurons without affecting generalization performance [16].

(3) In order to enhance the ability of ELM in time-sensitive data, Liang et al. proposed an online sequential ELM (OS-ELM, A fast and accurate online sequential learning algorithm for feedforward networks) in 2006, which can be fixed or different In the case of block size, learn the data block by block or step by step [17]. Zhao et al. introduced the forgetting mechanism into OS-ELM (FOS-ELM, online sequential extreme learning machine with forgetting mechanism) in 2012 to reflect short-term predictions Timeliness of training data. This algorithm is suitable for practical applications where the training data is not only discrete or block data, but also has the characteristics of training delay. For example, when short-term prediction of stock prices, outdated training data with low efficiency, Weighted processing

should be performed during the training process [18]. In 2009 (Online sequential fuzzy extreme learning machine for function approximation and classification problems), Rong et al. proposed an OS-Fuzzy-ELM for the TSK fuzzy mode, which has any the non-constant piecewise continuous membership function of the world [19]. In 2013, Ye et al. extended OS-ELM (Online sequential extreme learning machine in nonstationary environments) to data that changes at training time under non-stationary conditions [20].

(4) The development of ELM is very diversified. In dealing with imbalanced data, Zong et al. proposed weighted ELM (W-ELM, weighted extreme learning machine for imbalance learning) in 2013 to deal with imbalanced data. Compared with the classic ELM All training data are standardized differently, and W-ELM re-weights the training errors corresponding to different input data by adding different penalty coefficients [21].

(5) For noisy or missing data, in 2011, Man et al. (A new robust training algorithm for a class of single-hidden layer feedforward neural networks) proposed a FIR-ELM that improves the performance of ELM on noisy data. The input weights are assigned based on the finite impulse response filter (FIR), making the hidden layer as the preprocessing layer, which improves the robustness (resistance to change) of the model [22]. In 2012, Man(Robust single -hidden layer feedforward network-based pattern classifier) continues to propose a DFT-ELM algorithm based on Discrete Fourier Transform (DFT), similar to FIR-ELM, DFT-ELM eliminates noise in input data by designing input weights The influence of interference has improved the robustness (generalization) of ELM [23]. In 2013, Horata et al. (Robust extreme learning machine) proposed three improved ELM algorithms to solve the problem of outlier robustness: ① ELM based on iterative weighted least squares (IRWLS-ELM); ② ELM based on multivariate least edge square (MLTS-ELM); ③ ELM based on one-step weighted MLTS (RMLTS-ELM). In IRWLS, the function is estimated by M (For example, the Huber function) iteratively weights the training data, and then solves the weighted ELM in each loop. Therefore, the influence of outliers can be gradually reduced by reducing the weight. In MLTS-ELM and RMLTSELN, The Multivariate Least Squares Estimator (MLTS) based on the Minimum Covariance Determinant (MCD) estimator was introduced to improve the robustness of ELM against outliers [24]. In 2013, Yu et al. (Regularized extreme learning machine) for regression with missing data) studied the ELM regression problem of missing data, and proposed a Tikhonov regularized optimal pruning ELM (TROP-ELM) to deal with the missing data problem. In TROP-ELM, missing values are first determined by them the respective traditional mean replacement. It is proposed to use Gaussian function as the active function, randomly select the center from the input data, and estimate the square of the pairwise distance from the observation data. When the distance matrix is ready, the output matrix of the hidden layer can be calculated. In order to improve the compactness of the network, using Least Angle Regression (LARS) for hidden neurons Row sorting, and then using Tikhonov regularized press model to select the

optimal number of neurons according to the leave-one-out method error [25].

ELM collection, combining multiple learning machines can reduce the risk of overfitting and lead to better generalization performance. In 2009, Lan et al. (Ensemble of online sequential extreme learning machine) proposed an online sequence ELM (EOS-ELM) integration, which uses the average prediction of multiple independently trained OS-ELMs as the final prediction. EOS-ELM further improves the prediction accuracy of OS-ELM [26]. In 2010, Liu and Wang (Ensemble based Extreme learning machine) embeds ensemble learning techniques and cross-validation methods into the training stage to alleviate overtraining problems and improve the predictive stability of ELM [27]. In 2012, Cao et al. proposed voting-based ELM (V-ELM, Voting based extreme learning machine) also belongs to the collection of ELM [28].

ELM application ranking problem, which is usually converted into a decomposable learning problem, such as regression or classification, and training a predictive model from the data. In 2014, in Chen et al. (Extreme learning machine for ranking: Generalization analysis and applications) [29] and Zong and Huang (Learning to rank with extreme learning machine. Neural processing letters, ELM is effectively extended to learning ranking tasks. Zong proposed two ranking algorithms based on ELM, namely pointwise Rank-ELM and Cheng For Rank-ELM. In the pointwise ranking model, each training input corresponds to a query and a document, and the training output t is used to determine the degree of relevance between them. It can be learned through the classification model, and it is suitable for ELM in pair. In the ranking model, the goal is to model the relative relevance of query document pairs [30].

ELM is used for semi-supervised learning. In many practical problems such as text classification, spam filtering and natural language processing, it is not easy to obtain training labels, and the amount of unlabeled data is large and easy to collect. Semi-supervised learning is a way of using labeled and unlabeled data to achieve higher prediction accuracy than purely supervised learning or unsupervised learning technology. By 2013, ELM was extended to semi-supervised learning, Li et al. (A semi-supervised extreme learning machine method based on co-training) Proposed a joint training method to train ELM. This method trains one ELM in each iteration, and adds labeled data with corresponding prediction labels to the labeled training set, and gradually expands the pseudo-labeled training set, thereby improving Prediction accuracy [31]. In 2014, Huang et al. (Semi-supervised and unsupervised extreme learning machines) proposed a semi-supervised ELM based on manifold regularization (SS-ELM). SS-ELM inherited the advantages of traditional ELM, Maintains the learning ability and computational efficiency of ELM, and effectively handles multiple types of problems [32]. ELM is also applied in unsupervised learning, Huang proposed US in 2014 (Semi-supervised and unsupervised extreme learning machines) -ELM, that is, the input data is mapped to an m -dimensional space, where the data is well clustered. Therefore, US-ELM can be used for nonlinear dimensionality reduction or embedding [32]. In 2013, Benoit et al. (Feature selection for nonlinear models

with extreme learning machines) proposed a supervised feature selection method based on over-limit learning machine (FS-ELM). In this algorithm, ELM with leave-one-out error (LOO) is used to quickly evaluate generalization error, in order to achieve the search for the best feature subset [33].

In addition, the model of the over-limit learning machine ELM is simple and flexible. ELM has many variants developed for special applications. Some important variants of ELM are summarized as follows: Bayesian ELM, fuzzy ELM, wavelet ELM, complex ELM, genetic/evolution ELM etc. [3].

Lu Yang et al. discovered the necessity of selecting the output matrix and selected stable and good training errors with the help of the singular value characteristics of the output matrix [34]. In summary, the improvements to ELM are all related to the output matrix. The input, input weights and biases related to data preprocessing are the columns after passing through the hidden layer, the number of neuron nodes is the row, and the solution of the output weight is also related to Correlation, illustrates the necessity of selecting and improving the output matrix.

Based on the error from the hidden layer to the output layer in the ELM algorithm, this paper analyzes the ELM error carefully and finds that the error comes from the process of solving the generalized inverse matrix of the hidden layer output matrix. In order to further reduce the algorithm error, explore the appropriate target matrix and stable index related to the algorithm error, through experiments, it is determined that the L21 norm of the target matrix is linearly related to the ELM error. According to this phenomenon, Gaussian filtering is introduced to reduce the noise of the target matrix. To change the L21 norm of the target matrix to achieve the purpose of optimizing the error of the ELM algorithm.

5. CHAPTER 3: RESEARCH AND/OR DEVELOPMENT METHODOLOGY

The research method used in this article is experiment (hypothesis testing).

- Finding the problem: Aiming at the error problem of the ELM algorithm from the hidden layer to the output layer, through analysis, it is found that the error comes from the hidden layer output matrix H to solve the process of the Moore-Penrose generalized inverse matrix;
- Design experiments: determine the source of error, explore the appropriate target matrix, and determine the stability index of the target matrix;
- Conclusion: The experimental analysis shows that the L21 norm has a significant linear correlation with the ELM error;
- The improvement is proposed: Gaussian filter is introduced to reduce the noise of the target matrix, which effectively reduces the L21 norm of the target matrix, and at the same time reduces the error of the ELM, and achieves the purpose of optimizing the error of the ELM algorithm.

The design method, activation function, error calculation method of the training set data in the above experiment are all from the paper that first proposed the ELM algorithm. The experiments in this paper adopt the control variable method. The independent variables include the noise interval of the training set, the regression function of the training set, the activation function, etc., and the dependent variable is the error of the algorithm and the index of the candidate objective function matrix. After changing the independent variable factors, observe and record the data of the dependent variable, and analyze the relationship between the dependent variables.

The first part of the experiment: analyze the source of error. The experimental hypothesis of this article is to find that the ELM algorithm error comes from the process of solving the generalized inverse matrix with the hidden layer output matrix. Matlab2017(b) will be used for data simulation experiments. First generate and store the simulated data set; then calculate the ELM algorithm and store all the data generated during the calculation; according to the hypothetical experiment, it is found that the input in the training set corresponds to the column after passing through the hidden layer, and the row corresponds to the number of neuron nodes. Through the control variable method, adjust the size of the hidden layer neurons in the ELM under the same training set. When the error of the output matrix H to find the generalized inverse matrix becomes smaller, the ELM error will also become smaller.

The second part of the experiment: Determine the target matrix index linearly related to the error. Based on this finding, it is determined that the error comes from the process of finding the generalized inverse matrix of the output matrix. In order to clarify the correlation, a follow-up experiment is designed to determine a target matrix index that is correlated with the ELM error. According to the definition of the generalized inverse matrix and the auxiliary matrix in the ELM operation process, firstly, three candidate target matrices are determined when the fixed index is the L21 norm. Subsequently, six indicators (18 indicators in three candidate target matrices) are introduced to determine the only objective function and indicator—the L21 norm of $H^{\dagger}H$. Experimental analysis shows that the L21 norm has a significant linear correlation with the ELM error.

The third part of the experiment: Based on the phenomenon of the above experiment, it is found that the goal of optimizing the error of the ELM algorithm can be achieved by improving the index of the target matrix. The Gaussian filter is introduced to reduce the noise of the target matrix, which effectively reduces the L21 norm of the target matrix and reduces the ELM error.

The suitability of the above design can be reproduced through the procedures in the appendix. The experiment is distributed in an orderly manner, is effective, and the experimental results are as expected. And according to the experiments in this article, we can study the output matrix optimization based on Lasso regression, evolutionary algorithm and other methods. It is also possible to reduce the ELM training error by optimizing the target matrix and other related to the output matrix, or directly optimizing the output matrix.

6. CHAPTER 4: DESIGN AND IMPLEMENTATION

6.1 Design Extreme learning machine algorithm and related concepts

The extreme learning machine is a single hidden layer feedforward neural network (SLFN), which consists of an input layer, a hidden layer and an output layer. Due to the randomness of the input weight and bias, the hidden layer to the output layer is a linear output, compared with SLFN There is no output bias, so the structure of the extreme learning machine is shown in Figure (1):

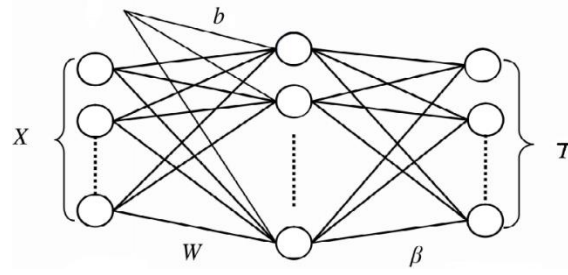


Fig.1 ELM network structure diagram

For N arbitrary distinct samples (x_i, t_i) , where $x_i \in R$ and $t_i \in R$, $i = 1, 2, \dots, N$. SLFNs with Ne hidden nodes and activation function $g(x)$ are mathematically modeled as

$$\sum_{i=1}^N \beta_i g(w_i x_j + b_i) = t_j, j = 1, 2, \dots, N \quad (1)$$

For arbitrary $i = 1, 2, \dots, N$, w_i is the input weight connecting the i_{th} hidden node and the input nodes, β_i is the output weight connecting the i_{th} hidden node and the output nodes, and b_i is the threshold of the i_{th} hidden node.

$$H(w_1, \dots, w_N, b_1, \dots, b_N, x_1, \dots, x_N)$$

When $\beta = (\beta_1, \dots, \beta_N)^T$, $T = (t_1, \dots, t_N)^T$ and

$$= \begin{bmatrix} g(w_1 x_1 + b_1) & \dots & g(w_N x_1 + b_N) \\ \dots & \dots & \dots \\ g(w_1 x_N + b_1) & \dots & g(w_N x_N + b_N) \end{bmatrix}_{N \times N}$$

H is called the hidden layer output matrix of the neural network; the i_{th} column of H is the i_{th} hidden node output with respect to inputs $x_1, \dots, x_N$. Then the above formula can be simplified to

$$H\beta = T \quad (2)$$

Because in ELM, when the activation function $g(x)$ is infinitely differentiable, the input weight w_i and bias b_i can be randomly determined [1]. At this time, the optimization model of ELM is as follows:

$$\begin{aligned} \min_{w_i, b_i} & \left\| H(w_1, \dots, w_N, b_1, \dots, b_N) \hat{\beta} - T \right\| \\ \text{s.t. } & \hat{\beta} = H^\dagger T \end{aligned} \quad (3)$$

Among $\varepsilon = \left\| H(w_1, \dots, w_N, b_1, \dots, b_N) \hat{\beta} - T \right\|$ is defined as the algorithm error of ELM, and $H^\dagger$ is the Moore-Penrose generalized inverse matrix of H . Training the ELM algorithm is to get the least square solution $\hat{\beta}$ of the model.

6.1.1 Moore-Penrose generalized inverse matrix

For any matrix, if there exists a matrix that satisfies the following Moore-Penrose equations.

$$AGA = A \quad (4)$$

$$GAG = G \quad (5)$$

$$(AG)^* = AG \quad (6)$$

$$(GA)^* = GA \quad (7)$$

Then call G the Moore-Penrose generalized inverse matrix of A , denoted by $A^\dagger$, where A^* denotes the transposed conjugate matrix of A .

6.1.2 The norm of a matrix

When a certain matrix space is established as a normed vector space, the norm equipped for the matrix is called the norm of the matrix. The norm of a matrix is a certain numerical characteristic of the reaction matrix, so according to different definitions, L1-Norm, F-Norm (L2-Norm), 1-Norm, kernel-Norm, L21-Norm, L12-Norm, etc. [35].

The L1-Norm is the sum of the absolute values of all elements of the matrix, which can describe the sparsity of the matrix and is defined as

$$\|W\|_{L1} = \sum_{i,j} |w_{i,j}| \quad (8)$$

F-Norm (L2-Norm) is the European norm of a matrix, that is the arithmetic square root of the sum of squares of all elements of the matrix, defined as:

$$\|W\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n W_{i,j}^2} \quad , \quad W \in \mathbb{R}^{m \times n} \quad (9)$$

1-Norm is to sum the absolute value of each column of the matrix, and then select the value with the largest value, which is defined as:

$$\|W\|_1 = \max_j \sum_{i=1}^m |W_{i,j}| \quad (10)$$

The kernel norm is equivalent to the sum of matrix eigenvalues, defined as:

$$\|W\|_* = \text{tr}(\sqrt{W^T W}) \quad (11)$$

The L21 norm is defined as: for matrix W , first find the 2-norm of each row vector (the sum of squares of each element and then take the square root), and then find the 1-norm of the generated column vector (that is the absolute value of each element) Sum of values, therefore the formula is:

$$\|W\|_{2,1} = \sum_{i=1}^m \sqrt{\sum_{j=1}^n |W_{i,j}|^2} \quad (12)$$

The L12 norm is the same as the L21 norm. For the matrix, first find the 1-norm of each column vector, and then find the 2-norm of the generated row vector, so the formula is

$$\|W\|_{1,2} = \sqrt{\sum_{i=1}^m \left(\sum_{j=1}^n |W_{i,j}| \right)^2} \quad (13)$$

6.1.3 Gaussian filtering

Gaussian filtering is a linear smoothing filter, which is suitable for eliminating Gaussian noise and is widely used in the denoising process of image processing. Gaussian filtering is the process of weighted averaging the entire data matrix. Each element is weighted and averaged by itself and other element values in the neighborhood [36] The following is a two-dimensional Gaussian distribution:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (14)$$

6.2 ELM optimization based on hidden layer output matrix

The ELM algorithm process is that when the input of the training set is X and the output is T , and the input weight and bias can be randomly determined under the premise that the

activation function $g(x)$ is infinitely differentiable, and the corresponding hidden layer output matrix is generated, and the corresponding output weight is obtained Output value. In the training set, every time the ELM algorithm is completed, a set of parameters X 、 T 、 W 、 b 、 H 、 β and $\mathcal{E}$ will be generated, and one-to-one correspondence. Therefore, after N times of ELM training, the best set of ELM parameters in N times of experiments can be obtained from the error index.

When analyzing the error of the ELM algorithm, it is found that formula (3) can be further simplified as:

$$\varepsilon = \|H\hat{\beta} - T\| = \|HH^\dagger T - T\| = \|HH^\dagger - E\|T \quad (15)$$

Where E is the identity matrix. It is known from formula (15) that the error of the ELM algorithm comes from the output matrix and the generalized inverse matrix, and the size of the training set in ELM is much larger than the number of hidden layer neurons, so it is a singular matrix, and only the generalized inverse exists. Therefore, it is inferred that the error of the ELM algorithm comes from the process of generating the generalized inverse matrix of the hidden layer output matrix. The above formula (15) also shows that in the same training set, the error is the distance from the matrix to the identity matrix. According to the size of this distance, the output matrix corresponding to the small training error can be selected. An index is needed to measure the distance from the matrix to the identity matrix, that is, a target matrix and index need to be determined to establish the relationship with ELM error. Based on this discovery, the ELM algorithm optimization process shown in Figure (2) is designed.

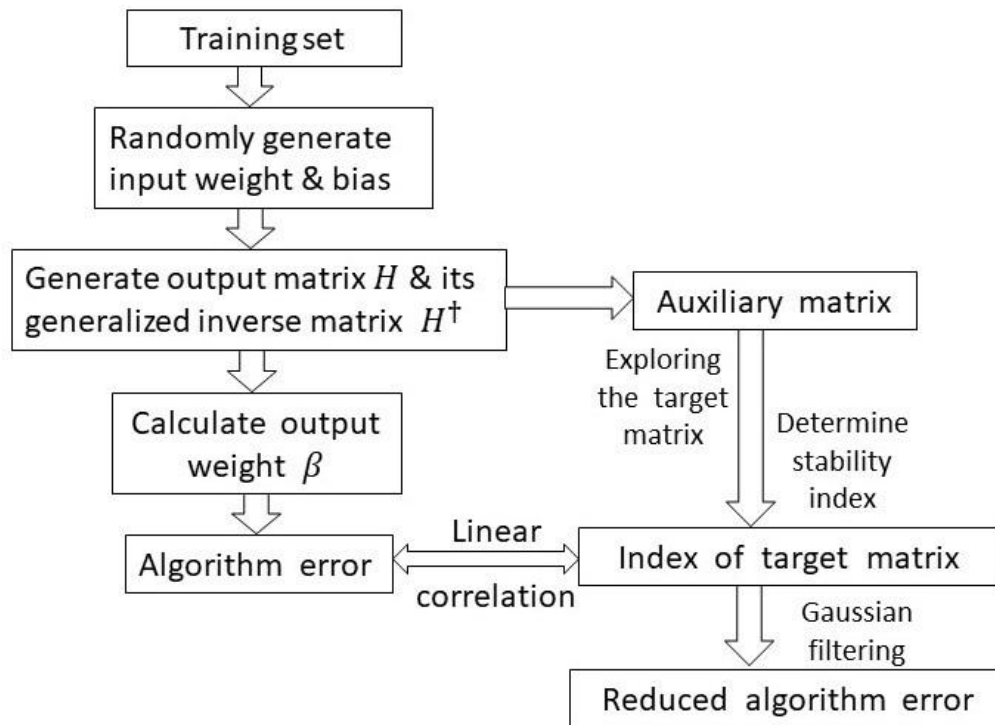


Fig.2 ELM optimization flow diagram

In summary, the optimization process design experiment based on the ELM algorithm is as follows:

Step 1: Analyze the relationship between the error and the output matrix H ; as shown in formula (9), the error and the auxiliary matrix $H^\dagger H$ are related to the distance between the identity matrix.

Step 2: Search for a suitable target matrix; when observing the distance from the auxiliary matrix to the identity matrix, it is obtained from the definition of the Moore-Penrose generalized inverse matrix [14] $H^\dagger H = H^\dagger H H^\dagger H = H^\dagger H H^\dagger H H^\dagger H = \dots$. The target matrix may be, the square of or the root of. Select $H^\dagger H$, $\sqrt{H^\dagger H}$, $\sqrt[3]{H^\dagger H}$, $\sqrt[4]{H^\dagger H}$, $\sqrt[5]{H^\dagger H}$ and $(H^\dagger H)^2$, $(H^\dagger H)^3$, $(H^\dagger H)^5$, $(H^\dagger H)^7$ a total of 9 candidate target matrices.

Step 3: Determine stable indicators; observe the target matrix to the identity matrix

When the distance is, the diagonal element of the target matrix is required to be close to 1, and the other elements are close to 0. Therefore, the index of norm is introduced to explore the relationship with the existence of error. Including L1-Norm, F-Norm (L2-Norm), 1-Norm, nuclear-Norm, L21-Norm, L12-Norm, etc.

Step 4: Apply Gaussian filtering for noise reduction processing; perform noise reduction processing for the target matrix. Because the index of the target matrix is related to the algorithm error, the algorithm error is reduced by improving the target matrix.

7. CHAPTER 5: ANALYSIS AND RESULTS DISCUSSION

7.1 Experimental environment

The experiment platform is Intel i7-8550U 1.8GHz, 16GB and 1TB hard disk notebook, and the experiment was implemented with Matlab2017(b) on Windows10 system [37].

7.2 Experimental design and result analysis

The ELM algorithm optimization based on the hidden layer output matrix aims to show the process of generating the generalized inverse matrix of the hidden layer output matrix found during the algorithm operation. It is planned to determine whether the target matrix and the stability index have a linear relationship with the error through experiments. Design the following 4-step experiment:

Experiment 1. Analyze the relationship between error and output matrix;

Experiment 2. To find a suitable target matrix;

Experiment 3. To determine stable indicators;

Experiment 4. Apply Gaussian filtering to optimize the algorithm.

In this article, the training set is to randomly generate 500 sets of data input, input noise, and

output under uniform distribution, as shown in Table 1.

Table 1 Training sets generate table

Group	The interval of the input X	Noise σ interval	Function of output T	
1	[-10,10]	[-0.2,0.2]	①	$T = \frac{\sin x}{x} + e$
2			②	
3	[-10,10]	[-2,2]	①	
4			②	
5	[-2,2]	[-0.2,0.2]	①	$T = \sin(\frac{1}{x}) + e$
6			②	
7	[-2,2]	[-2,2]	①	
8			②	

Because ELM can determine the input weight w and bias b randomly under the premise that the activation function $g(x)$ is infinitely differentiable, and generate the corresponding hidden layer output matrix. Therefore, the choice of activation function $g(x)$ is also an important step of the algorithm. There are three commonly used activation functions as shown in Table 2 [1].

Table 2 List of activation functions

Activation function	Formula
Hard limit	$G(w, b, x) = \begin{cases} 1, w \cdot x + b \leq 0 \\ 0, otherwise \end{cases}$
Sigmoid	$G(w, b, x) = \frac{1}{1 + \exp(-(wx + b))}$
Sin	$G(w, b, x) = \sin(wx + b)$

7.3 Sources and results of analysis errors

ELM error is a process from hidden layer to output layer. When the output weight w is calculated from the output matrix H and output T , the generalized inverse matrix $H^\dagger$ needs to be calculated, and the error occurs. The auxiliary matrix $H^\dagger H$ generated during the introduction of ELM operation. If the output matrix is a non-singular matrix, $H^{-1}H = E$, it is speculated that the auxiliary matrix should be closer to the identity matrix, and the resulting ELM algorithm error is smaller. The rows and columns of the output matrix correspond to the number of neuron nodes and the size of the training set respectively.

The design experiment is as follows. Determine the column of $H^\dagger H$ (training set size) as 1x500 (take the first group in Table 1 as an example), adjust the size of the row of $H^\dagger H$ (number of neuron nodes) to record the corresponding ELM algorithm error, and select Sin The activation function takes 50 experiments as a group, and takes the average value after 100 cycles. The data obtained is shown in Table 3.

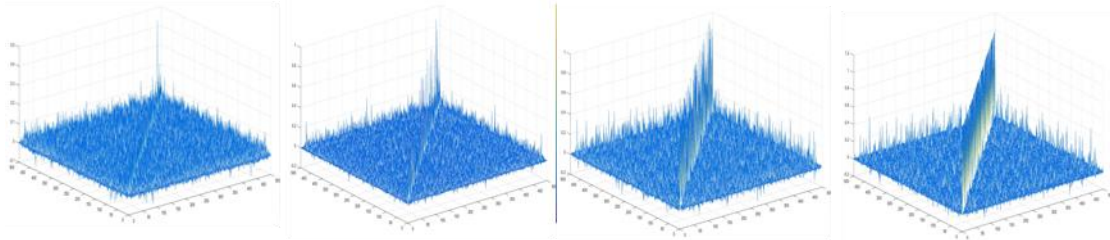


Fig.3 The numerical three-dimensional diagram of $H^\dagger H$ corresponding to different nodes (50, 100, 200, 499)

Table 3 relation of neuron node number and error

Number of neuron nodes	Minimum error	Average error
50	0.112691	0.1878
100	0.103175	0.1068
200	0.0875043	0.0894
499	0.0585408	0.0631

In Table 3, it can be seen that both the minimum error and the average error decrease with the increase of the number of neuron nodes, so the change of the output matrix H affects the change of the error. Further experiment, select the parameter group corresponding to the smallest error, adjust the number of neuron nodes, observe the distance from the auxiliary matrix $H^\dagger H$ to the identity matrix, and obtain the $H^\dagger H$ matrix corresponding to the different nodes (50, 100, 200, 499) shown in Figure 3 Numerical three-dimensional graph.

In Figure 3, when the number of nodes is 50, 100, 200, and 499, the diagonal element value of the corresponding matrix fluctuates between 0.1, 0.2, 0.4, and 1, and the corresponding error is also reduced. Explain that the size of the error is related to the distance between the auxiliary matrix $H^\dagger H$ and the identity matrix. The output matrix with small training error can be selected according to the distance between the matrix $H^\dagger H$ and the identity matrix. According to the phenomenon found in Experiment 1, the next step is to introduce indicators to measure the distance from the matrix $H^\dagger H$ to the unit matrix, determine the target matrix with a smaller distance from the unit moment through experiments, and further analyze the relationship with the error.

7.4 Search for a suitable target matrix and result analysis

In order to find a suitable target matrix, by the definition of auxiliary matrix and Moore-Penrose generalized inverse matrix, it is found that the target matrix may be $H^\dagger H$, $\sqrt{H^\dagger H}$, $\sqrt[3]{H^\dagger H}$, $\sqrt[5]{H^\dagger H}$, $\sqrt[7]{H^\dagger H}$ and $(H^\dagger H)^2$, $(H^\dagger H)^3$, $(H^\dagger H)^5$, $(H^\dagger H)^7$ etc. Possible indicators are L1-Norm, F-Norm, 1-Norm, 2-Norm (spectral norm), nuclear-Norm, etc. After that, an association analysis will be performed, multiple target matrices and multiple indicators. The correlation diagram is shown in Figure 4.

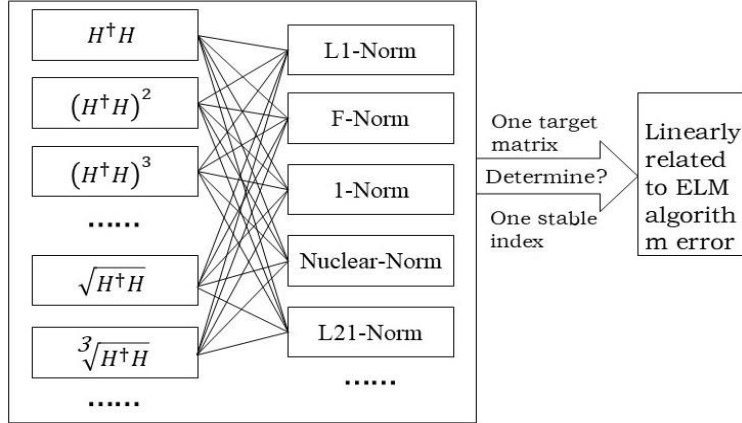


Fig.4 Diagram of multi-objective function and multi-index

The research object is the distance between the target matrix and the identity matrix. The L1 norm which can better represent the numerical characteristics of the matrix is temporarily selected as the index, and the target matrix is initially selected through experiments. Carry out the following experiment. In the same training set with input X of 1×500 (take the first group in Table 1 as an example), the activation function is the Sin function, 50 experiments are used as a group, and the average value is taken for 100 cycles. The experimental results See Table 4 and Figure 5.

Table 4 Correlation analysis of errors with the L1-Norm of the objective function

Experimental group	Training set group	Activation function	The correlation coefficient of error and the L1-Norm of candidate target matrix $H^\dagger H$, $\sqrt{H^\dagger H}$, $\sqrt[3]{H^\dagger H}$, $\sqrt[5]{H^\dagger H}$, $\sqrt[7]{H^\dagger H}$, $(H^\dagger H)^2$, $(H^\dagger H)^3$, $(H^\dagger H)^5$, $(H^\dagger H)^7$
1	1	Hard limit	-0.7456, -0.6732, -0.6323, -0.5839, -0.5496, 0.05553, 0.4452, 0.1141, 0.0636
2		Sig	-0.0174, 0.1447, 0.1814, 0.2096, 0.2214, 0.01934, 0.756531 , 0.8602 , 0.85058
3		Sin	0.7726 , 0.7572 , 0.7530 , 0.7499 , 0.7486 , 0.0209, -0.8189, -0.7645, -0.7316

4	2	Hard limit	-0.0724、-0.0672、-0.0709、-0.0721、-0.0698、 -0.0088、 0.3251 、0.2918、0.2578
5		Sig	-0.4482、-0.2882、-0.2372、-0.1976、-0.1810、 -0.2412、 0.7933 、 0.8045 、 0.7852
6		Sin	-0.4270、-0.4117、-0.4069、-0.4033、-0.4019、 0.1695、0.4257、0.4163、 0.4374
7	3	Hard limit	-0.3483、-0.2043、-0.1628、-0.1330、-0.1214、 -0.4659 、-0.0704、-0.0575、-0.1662
8		Sig	-0.1813、-0.0772、-0.0537、-0.0379、-0.0318、 -0.2644、-0.1293、-0.1271、 0.4431
9		Sin	-0.6458 、 -0.6418 、 -0.6397 、 -0.6375 、 -0.6365 、 -0.1204、-0.3876、-0.3873、0.5181
10	4	Hard limit	0.0227、0.2802、0.3561、0.4109、 0.4327 、 -0.2210、0.1779、0.1991、-0.2388
11		Sig	0.227、0.2802、0.3561、0.4109、 0.4327 、 -0.2210、0.1779、0.1991、-0.2388
12		Sin	-0.8735 、 -0.8713 、 -0.8707 、 -0.8702 、 -0.8700 、 -0.1594、 -0.8098 、 -0.8100 、 -0.6672

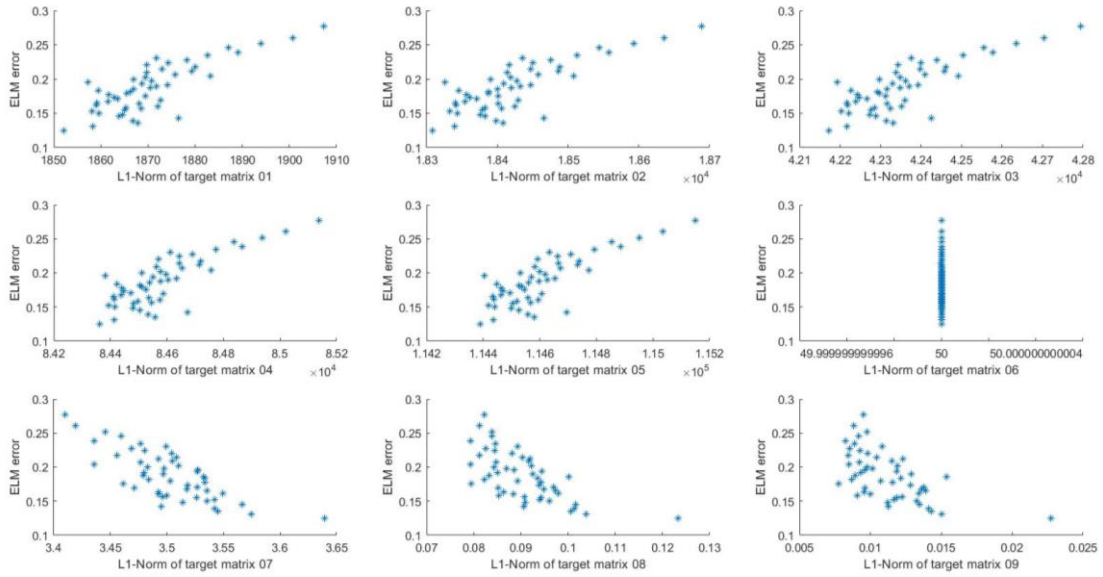


Fig.5 Diagram of 9 alternative target matrices L1-norm and algorithm error

In the 12 sets of comparative experiments in Table 4, the control variable method is used to initially select the target matrix. As shown in the above table, the independent variables include the noise interval of the training set, the regression function of the training set, the activation function, etc., and the dependent variable is the algorithmic Error and candidate target matrix, L1-Norm of $H^{\dagger}H$, $\sqrt{H^{\dagger}H}$, $\sqrt[3]{H^{\dagger}H}$, $\sqrt[4]{H^{\dagger}H}$, $\sqrt[5]{H^{\dagger}H}$ and $(H^{\dagger}H)^2$, $(H^{\dagger}H)^3$, $(H^{\dagger}H)^5$, $(H^{\dagger}H)^7$. The error in Figure 5 has a linear relationship with the L1 norm of the

candidate target matrix, which determines the direction for subsequent experiments.

According to the two principles that the absolute value of the correlation coefficient is greater than 0.6 and the absolute value of the correlation coefficient in each group of experiments is the largest, the top three primary selection target matrices $H^\dagger H$ 、 $(H^\dagger H)^3$ and $(H^\dagger H)^7$ and sums are counted and voted. It is planned to introduce more indicators related to the target matrix. Further experiments include six indicators: L1-Norm, F-Norm (L2 norm), 1-Norm, kernel-Norm, L21-Norm, and L12-Norm.

7.5 Determining stable indicators and result analysis

According to the results of experiment 2, it is proposed to adopt the stability index with the best correlation coefficient with the error to determine the optimal target matrix among the three candidate target matrices in reverse, in order to illustrate the phenomenon of linear correlation between the error and the target matrix norm index Increase the stability of the comparison experiment to 24 groups. Table 5 shows the correlation analysis structure of the multiple norms and algorithm errors in the 3 candidate matrices.

Table 5 24 sets of comparative experimental data for the six paradigm indicators with the objective function $H^\dagger H$

Experimental group	Training set group	Activation function	The correlation coefficient of error and the L1-Norm、F-Norm、1-Norm、Kernel-Norm, L21-Norm, and L12-Norm of candidate target matrix $H^\dagger H$
1	1	Hard-limit	-0.8277、-0.2115、-0.5524、-0.2106、-0.8042、-0.8198
2		Sig	0.0037、0.0718、0.8180、0.1297、-0.8321、-0.0340
3		Sin	0.8320、-0.0279、0.6180、-0.0066、0.4669、0.8379
4	2	Hard-limit	-0.3127、-0.4824、-0.1199、-0.4824、-0.6267、-0.2677
5		Sig	-0.6933、0.1583、0.6643、0.1481、-0.7152、-0.7019
6		Sin	-0.5869、-0.0040、0.5776、0.0220、-0.6885、-0.5782
7	3	Hard-limit	-0.4685、-0.6303、-0.3325、-0.6306、-0.7873、-0.4375
8		Sig	-0.3331、0.1424、0.1447、0.1035、-0.8014、-0.2990
9		Sin	-0.5994、-0.1985、0.4404、-0.0234、-0.7300、-0.5860
10	4	Hard-limit	-0.5315、-0.6496、-0.0416、-0.6493、-0.6635、0.5553
11		Sig	-0.0926、-0.1396、0.5147、0.4531、-0.0019、-0.1532
12		Sin	-0.9100、-0.20752、0.7332、0.1709、-0.9197、-0.9093
13	5	Hard-limit	-0.8617、-0.6857、0.7785、-0.6848、-0.8807、-0.8295
14		Sig	0.0405、-0.0499、0.0962、-0.0123、-0.5376、0.0275
15		Sin	0.5742、0.0769、-0.3437、-0.2100、0.3454、0.5421
16	6	Hard-limit	-0.7407、-0.6050、-0.3824、-0.6047、-0.7780、-0.7459
17		Sig	0.5946、0.2742、0.6936、-0.5017、0.7220、0.6794
18		Sin	0.4672、0.2040、0.8478、-0.1789、-0.1161、0.6766
19	7	Hard-limit	-0.7444、-0.9000、-0.1401、-0.9004、-0.8814、-0.7648
20		Sig	-0.0405、0.0589、0.1200、-0.2826、-0.4458、-0.0546
21		Sin	-0.7911、-0.0418、0.6249、0.0277、-0.8705、-0.7915
22	8	Hard-limit	-0.7912、-0.8644、-0.6953、-0.8643、-0.871、-0.8032
23		Sig	0.6181、0.2768、0.2921、-0.3954、-0.2683、0.6559
24		Sin	0.6503、-0.0818、-0.1654、0.1251、-0.3323、0.6683

Table 6 24 sets of comparative experimental data for the six paradigm indicators with the objective function $(H^\dagger H)^3$

Experimental group	Training set group	Activation function	The correlation coefficient of error and the L1-Norm、F-Norm、1-Norm、Kernel-Norm, L21-Norm, and L12-Norm of candidate target matrix $(H^\dagger H)^3$
1	1	Hard-limit	0.6655、0.3576、0.3676、0.4704、0.5466、0.3915
2		Sig	0.8927、0.8990、0.8500、0.9162、0.9086、0.9074
3		Sin	-0.8560、-0.8057、-0.8224、-0.7520、-0.8129、-0.8337
4	2	Hard-limit	0.5186、0.4031、0.3542、0.5169、0.5175、0.4321
5		Sig	0.8430、0.8407、0.8487、0.8329、0.8200、0.8493
6		Sin	-0.5869、0.6092、0.4673、0.6650、0.6274、0.5895
7	3	Hard-limit	0.4539、0.2408、0.2289、0.4106、0.4211、0.2630
8		Sig	0.6216、0.6056、0.6534、0.5992、0.5792、0.6155
9		Sin	0.6483、0.5912、0.3182、0.7008、0.6734、0.5951
10	4	Hard-limit	0.3220、0.1400、0.0965、0.2961、0.3061、0.1657
11		Sig	-0.1514、-0.3232、-0.2132、-0.1840、-0.2966、-0.2929
12		Sin	0.9133、0.8687、0.8422、0.8994、0.9046、0.8794
13	5	Hard-limit	0.5717、0.3254、0.2878、0.5553、0.5478、0.3663
14		Sig	-0.1403、-0.5205、-0.6794、-0.2142、-0.1859、-0.4744
15		Sin	-0.6089、-0.6845、-0.6771、-0.6278、-0.5954、-0.6779
16	6	Hard-limit	0.5612、0.5113、0.5242、0.5397、0.5582、0.5313
17		Sig	-0.8551、-0.9070、-0.9016、-0.8313、-0.8261、-0.9143
18		Sin	-0.6621、-0.8606、-0.7463、-0.7566、-0.6761、-0.8625
19	7	Hard-limit	0.4239、0.0412、0.0479、-0.1932、-0.2316、-0.0115
20		Sig	0.9198、0.9255、0.9175、0.8944、0.9066、0.9261
21		Sin	-0.7911、-0.0418、0.6249、0.0277、-0.8705、-0.7915
22	8	Hard-limit	-0.3445、-0.1955、-0.1591、-0.3246、-0.3059、-0.1959
23		Sig	-0.8102、-0.9242、-0.8691、-0.8577、-0.8555、-0.9203
24		Sin	-0.6537、-0.8313、-0.8639、-0.6493、-0.6530、-0.8290

Table 7 24 sets of comparative experimental data for the six paradigm indicators with the objective function $(H^\dagger H)^7$

Experimental group	Training set group	Activation function	The correlation coefficient of error and the L1-Norm、F-Norm、1-Norm、Kernel-Norm, L21-Norm, and L12-Norm of candidate target matrix $(H^\dagger H)^7$
1	1	Hard-limit	0.3146、0.3243、0.3347、0.3108、0.3117、0.3243
2		Sig	0.9237、0.8951、0.8773、0.9210、0.9217、0.8962
3		Sin	-0.6584、-0.6084、-0.6116、-0.6285、-0.6392、-0.6167
4	2	Hard-limit	0.3533、0.3484、0.3449、0.3523、0.3523、0.3484
5		Sig	0.8981、0.8838、0.8773、0.8996、0.8976、0.8824
6		Sin	0.8219、0.7778、0.7432、0.8348、0.8308、0.7734
7	3	Hard-limit	0.2415、0.2217、0.2132、0.2372、0.2386、0.2230
8		Sig	0.5779、0.5302、0.5224、0.5731、0.5739、0.5361
9		Sin	0.5165、0.5101、0.4901、0.5329、0.5271、0.4926
10	4	Hard-limit	0.2782、0.2369、0.1887、0.2784、0.2783、0.2368
11		Sig	-0.4055、-0.2625、-0.1774、-0.3870、-0.3992、-0.2754
12		Sin	0.8368、0.8171、0.8151、0.8337、0.8347、0.8173
13	5	Hard-limit	0.4520、0.4352、0.4204、0.4503、0.4510、0.4355
14		Sig	-0.5916、-0.6604、-0.6526、-0.5949、-0.5933、-0.6602
15		Sin	-0.6985、-0.7257、-0.7361、-0.6996、-0.6988、-0.7256
16	6	Hard-limit	0.2500、0.2514、0.2521、0.2481、0.2489、0.2518
17		Sig	-0.8936、-0.8994、-0.8993、-0.8933、-0.8932、-0.8994
18		Sin	-0.8000、-0.7230、-0.7117、-0.8034、-0.8023、-0.7231
19	7	Hard-limit	-0.2031、-0.2184、-0.2267、-0.2012、-0.2011、-0.2189
20		Sig	0.7230、0.6766、0.6226、0.7221、0.7225、0.6765
21		Sin	0.8983、0.9111、0.8876、0.8979、0.8980、0.9111
22	8	Hard-limit	0.2432、0.2773、0.2976、0.2409、0.2415、0.2782
23		Sig	-0.9201、-0.9153、-0.8887、-0.9188、-0.9196、-0.9154
24		Sin	-0.8355、-0.8821、-0.8841、-0.8351、-0.8352、-0.8820

In the above experiment, a comparative experiment was carried out on 6 norm indexes of 3 candidate target matrices $H^\dagger H$. In order to obtain a stable index, 24 sets of experiments were performed on each target matrix and the data was recorded. In Table 5, the absolute value of the correlation coefficient between the error and the L21 norm is greater than 0.6 in 18 of the 24 groups, and 6 of the groups are greater than 0.8. In the 3 candidate matrices, $(H^\dagger H)^3$ and $(H^\dagger H)^7$ are also subjected to 24 sets of experiments, and the data is obtained to calculate the 6 norm index data (take the absolute value and then average), see Table 6 below.

Table 6 Correlation analysis of errors with the L1-Norm of 3 target matrices

Index Target matrix	L1- Norm	F-Norm	1-Norm	Kernel- Norm	L21-Norm	L12-Norm
$H^\dagger H$	0.5461	0.2935	0.4474	0.3258	0.6286	0.5591
$(H^\dagger H)^3$	0.6173	0.5606	0.5567	0.5715	0.6094	0.5965
$(H^\dagger H)^7$	0.5973	0.583	0.5698	0.5956	0.5963	0.5833

As shown in Table 6, the L21-Norm of $H^\dagger H$ found in the 18 indicators is the most stable indicator, and the absolute number average of the 24 experiments is the best correlation coefficient of 0.6286.

7.6 Algorithm optimization and result analysis using Gaussian filtering

After determining that the ELM algorithm error is linearly related to the L21 norm of the target matrix $H^\dagger H$, according to the linear correlation and the nature of the matrix norm, it is proposed to use Gaussian filtering to reduce the noise of the target matrix [16], through the reduced L21-Norm number of $H^\dagger H$, so as to achieve the purpose of reducing the error of the ELM algorithm. The specific experimental operation is to first put forward the diagonal elements of the original target matrix, filter the remaining non-diagonal elements, first zero out the diagonal elements of the current matrix, and then remove the diagonal elements of the original target matrix The elements are added back to get the new target matrix after filtering.

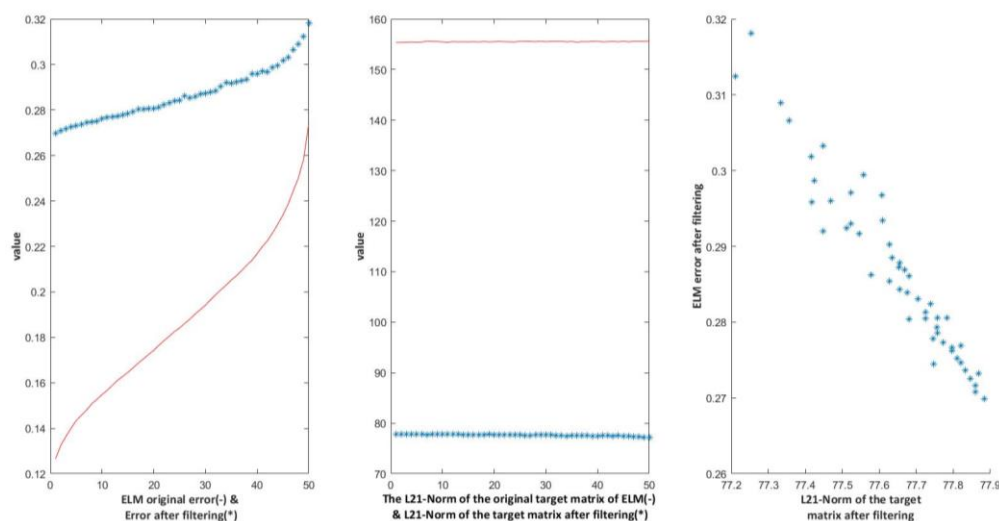


Fig.6 Diagrams of before and after filtering error, L21-norm and matrix L21-norm with error (negative correlation)

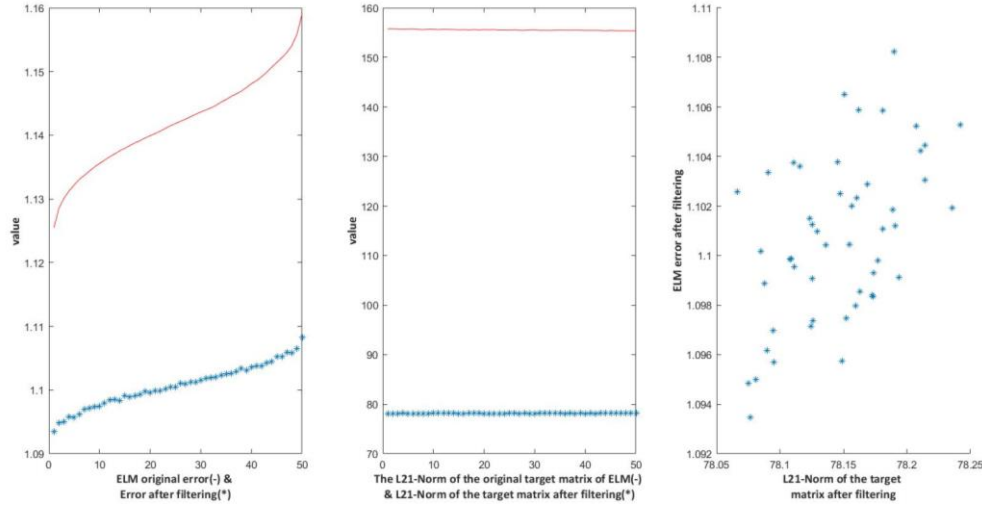


Fig.7 Diagrams of before and after filtering error, L21-norm and matrix L21-norm with error (positive correlation)

The above figures 6 and 7 are the change curves when the L21 norm of the target matrix and the algorithm error are negatively correlated and positively correlated. The red curve represents the initial data, and the blue scatter diagram represents the filtered data. In the case of positive correlation in Fig. 7, Gaussian filtering is used to optimize the target matrix $H^{\dagger}H$ to reduce its L21 norm, thereby achieving the goal of reducing algorithm errors.

8. CHAPTER 6: CONCLUSIONS AND RECOMMENDATIONS

This article is based on ELM's hidden layer output matrix H to optimize the algorithm error. In ELM, the size of the training set is much larger than the number of hidden layer neuron nodes, so the output matrix H generated by its corresponding row and column is a singular matrix, and the corresponding Moore-Penrose generalized inverse matrix HH needs to be generated to solve the output weight. The error of the generalized inverse matrix N causes the error of the ELM algorithm. According to the definition of the generalized inverse and the auxiliary matrix, it is inferred that the target matrix is HH , the square of HH or the square root of HH and the error index are the norm of the target matrix. The design experiment shows that the L21 norm of the target matrix HH is linearly related to the ELM error. Finally, the target matrix HH is optimized by applying Gaussian filtering to change its L21 norm to reduce the algorithm error.

As a one-time completion algorithm, ELM requires multiple trainings and then selects a good hidden layer output matrix. The training error corresponding to each output matrix has room for improvement [38]. In the experiment, the auxiliary matrix HH is selected. When considering the distance from the optimizing matrix HH to the identity matrix, Gaussian filtering is applied here to optimize HH , which can better reduce the effect of error. Other methods based on Lasso regression and evolutionary algorithms can also be studied for matrix optimization. The research in this paper shows that for the ELM algorithm, it is possible to further reduce the training error by optimizing the target matrix and other related to the

output matrix H, or directly optimizing the output matrix H be included [39].

9. CHAPTER 7: REFLECTION

In the experiment of this article, the first part of the experiment found the source of error, which is the analysis and discovery of the ELM algorithm operation process, which is based on theoretical discovery; the second part of the experiment further determines the target matrix index linearly related to the error, which is an experimental phenomenon Indicated. The experiment is reproducible, but it is unstable. In addition to the random determination of the input weight and bias of the ELM algorithm, it is impossible to obtain the same experimental data in the recurring experiment; the third part of the experiment is to optimize the ELM based on linear relationship and introduce Gaussian filtering for denoising , Optimize the target matrix and reduce the index, so as to achieve the purpose of reducing the ELM training error. This process can also apply other methods to achieve the purpose of optimizing the matrix. The optimization experiment is to optimize the target matrix, not the output matrix. Because of the computational complexity, when the output matrix is adjusted, the final algorithm error changes randomly, so this article optimizes the target matrix. In addition to the experiments that have been done, subsequent experiments can be optimized with the output matrix as the target matrix or the matrix related to the output matrix to achieve the purpose of stably reducing the ELM training error and optimizing the ELM algorithm [40].

10. REFERENCES

- [1] Huang G B, Zhu Q Y, Siew C K. Extreme learning machine: Theory and applications[J]. Neurocomputing, 2006, 70(1/3): p.489-501.
- [2] Huang G B, Wang D H, Lan Y. Extreme learning machines: a survey. Int J Mach Learn Cybern[J]. International Journal of Machine Learning and Cybernetics, 2011, 2(2): 107-122.
- [3] Huang G, Huang G B, Song S, et al. Trends in extreme learning machines: A review[J]. Neural Networks, 2015 , 61: 32-48.
- [4] Wang, Y. G., Cao, F. L. & Yuan, Y. B. A study on effectiveness of extreme learning machine. Neurocomputing, 2011, 74(16), 2483-2490.
- [5] Chen, Z. X. X., Zhu & H. Y. Y. A modified extreme learning machine with sigmoidal activation functions. Neural Computing & Applications, 2013, 22(3-4), 541-550.
- [6] Yuan, Y. B., Wang, Y. G. & Cao, F. L. Optimization approximation solution for regression problem based on extreme learning machine. Neurocomputing, 2011, 74(16), 2475-2482.
- [7] Huang, G.-B. & Chen, L. Convex incremental extreme learning machine. Neurocomputing, 2007, 70(16), 3056-3062.
- [8] Yang, Y. M., Wang, Y. N. & Yuan, X. F. Bidirectional extreme learning machine for regression problem and its learning effectiveness. IEEE Transactions on Neural Networks and Learning Systems, 2012, 23(9), 1498-1505.
- [9] Miche Y, Sorjamaa A, Bas P, et al. OP-ELM: Optimally Pruned Extreme Learning Machine[J]. Neural Networks IEEE Transactions on, 2010, 21(1): P.158-162.
- [10] Zhang, R., Lan, Y., Huang, G. B. & Xu, Z. B. Universal approximation of extreme learning machine with adaptive growth of hidden nodes. IEEE Transactions on Neural Networks

- and Learning Systems, 2012, 23(2), 365-371.
- [11] Deng, J., Li, K. & Irwin, G. W. Fast automatic two-stage nonlinear model identification based on the extreme learning machine. *Neurocomputing*, 2011, 74(16), 2422-2429.
 - [12] Wang N, Er M J, Han M. Parsimonious Extreme Learning Machine Using Recursive Orthogonal Least Squares[J]. *IEEE Transactions on Neural Networks & Learning Systems*, 2014, 25(10): 1828-1841.
 - [13] Luo J, Vong C M, Wong P K. Sparse Bayesian Extreme Learning Machine for Multi-classification[J]. *IEEE Transactions on Neural Networks & Learning Systems*, 2014, 25(4): 836-843.
 - [14] Yu, D. & Deng, L. Efficient and effective algorithms for training single hidden- layer neural networks. *Pattern Recognition Letters*, 2012, 33(5), 554-558.
 - [15] Bai Z, Huang G B, Wang D, et al. Sparse Extreme Learning Machine for Classification[J]. *IEEE Transactions on Cybernetics*, 2014, 44(10): 1858-1870.
 - [16] Gastaldo, P., Zunino, R., Cambria, E. & Decherchi, S. Combining ELM with random projections. *IEEE Intelligent Systems*, 2013, 28(5), 18-20.
 - [17] Liang, N.-Y., Huang, G.-B., Saratchandran, P. & Sundararajan, N. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, 2006, 17(6), 1411-1423.
 - [18] Zhao J, Wang Z, Park D S. Online sequential extreme learning machine with forgetting mechanism[J]. *Neurocomputing*, 2012, 87: p. 79-89.
 - [19] Rong, H. J., Huang, G. B., Sundararajan, N. & Saratchandran, P. Online sequential fuzzy extreme learning machine for function approximation and classification problems. *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, 2009, 39(4), 1067-1072.
 - [20] Ye Y, Squartini S, Piazza F. Online sequential extreme learning machine in nonstationary environments[J]. *Neurocomputing*, 2013, 116(sep.20): 94-101.
 - [21] Zong, W. W., Huang, G.-B. & Chen, Y. Q. Weighted extreme learning machine for imbalance learning. *Neurocomputing*, 2013, 101, 229-242.
 - [22] Man, Z. H., Lee, K., Wang, D. H., Cao, Z. W. & Miao, C. Y. A new robust training algorithm for a class of single-hidden layer feedforward neural networks. *Neurocomputing*, 2011, 74(16), 2491-2501.
 - [23] Man, Z. H., Lee, K., Wang, D. H., Cao, Z. W. & Khoo, S. Y. Robust single-hidden layer feedforward network-based pattern classifier. *IEEE Transactions on Neural Networks and Learning Systems*, 2012, 23(12), 1974-1986.
 - [24] Horata, P., Chiewchanwattana, S. & Sunat, K. Robust extreme learning machine. *Neurocomputing*, 2013, 102, 31-44.
 - [25] Yu, Q., Miche, Y., Eirola, E., van Heeswijk, M., Severin, E. & Lendasse, A. Regularized extreme learning machine for regression with missing data. *Neurocomputing*, 2013, 102, 45-51.
 - [26] Lan, Y., Soh & Huang, G. B. Ensemble of online sequential extreme learning machine. *Neurocomputing*, 2009, 72(13-15), 3391-3395.
 - [27] Liu, N., & Wang, H. Ensemble based extreme learning machine. *IEEE Signal Processing Letters*, 2010, 17(8), 754-757.

- [28] Cao, J. W., Lin, Z. P., Huang, G. B. & Liu, N. Voting based extreme learning machine. *Information Sciences*, 2012, 185(1), 66-77.
- [29] Chen, H., Peng, J., Zhou, Y., Li, L. & Pan, Z. Extreme learning machine for ranking: Generalization analysis and applications. *Neural Networks*, 2014, 53, 119-126
- [30] Zong, W. W. & Huang, G.-B. Learning to rank with extreme learning machine. *Neural processing letters*, 2014, 9(2), 155-166.
- [31] Li, K., Zhang, J., Xu, H., Luo, S. & Li, H. A semi-supervised extreme learning machine method based on co-training. *Journal of Computational Information Systems*, 2013, 9(1), 207-214.
- [32] Huang, G., Song, S., Gupta, J. & Wu, C. Semi-supervised and unsupervised extreme learning machines. *IEEE Transactions on Cybernetics*, 2014.
- [33] Benoit, F., van Heeswijk & M., Miche. A. Feature selection for nonlinear models with extreme learning machines. *Neurocomputing*, 2013, 102, 111-124.
- [34] Yang Lv, Yiming Ding, Bang Li, Jinghu Yu. Reduce Training Error of Extreme Learning Machine by Selecting Appropriate Hidden Layer Output Matrix. To appear in *Journal of systems science and systems engineering*, 2020.
- [35] Gudmundsson T , Kenney C S , Laub A J . Small-Sample Statistical Estimates for Matrix Norms[J]. *Siam Matrix Analysis & Applications*, 2006, 16(3):776-792.
- [36] Ito K . Gaussian filter for nonlinear filtering problems[C]// *IEEE Conference on Decision & Control*. IEEE, 2002.
- [37] You-Liang Y , Shuai R , Cui-Hong M A . Quantitative Analysis of Liquid Steel with Technology of LIBS Based on ELM[J]. *Journal of Heb United University(Natural ence Edition)*, 2016.
- [38] Zhang, Yanan, Zhao, et al. Extreme learning machine: algorithm, theory and applications[J]. *Artificial Intelligence Review: An International ence and Engineering Journal*, 2015, 44(1):103-115.
- [39] Liu X , Wang L , Huang G B , et al. Multiple kernel extreme learning machine[J]. *Neurocomputing*, 2015, 149(pt.a):253-264.
- [40] Albadra M A A , Tiuna S . Extreme Learning Machine: A Review[J]. *International Journal of Applied Engineering Research*, 2017, 12(14):4610-4623.

11. APPENDICES

- 1. Program to generate training data (MATLAB programs)

```
X1=(rand(500,1).*2-1).*10; % 8 training sets in total
```

```
X2=(rand(500,1).*2-1).*10;
```

```
e=(rand(500,1).*2-1).*0.2;
```

```
T=e+((sin(X1+X2))./(X1+X2));
```

```
% X=[X1,X2];
```

```
Uniform=zeros(500,4);
```

```
Uniform(:,1)=X1;
```

```
Uniform(:,2)=X2;
```

```
Uniform(:,3)=e;
```

```
Uniform(:,4)=T;
```

```
save data0101; % Produce uniform distribution
```

- 2. Main program (MATLAB programs)

```
clc
```

```
clear
```

```
load ('data0101.mat') %could load data0102.mat, data0201.mat...
```

```
train_data=Uniform;
```

```
T=train_data(:,4);
```

```
X=train_data(:,1:2);
```

```
clear train_data;
```

```
NumberOfHiddenNeurons=50;
```

```
ActivationFunction='sin';
```

```
NumberOfTrainingData=size(X,1);
```

```
NumberOfInputNeurons=size(X,2);
```

```
Iter=50;% Number of runs in a cycle
```

```
IterNum=100;% Number of iterative cycles
```

```
Hdata=zeros(Iter,IterNum);
```

```
Hdata2=zeros(Iter,IterNum);
```

```

Hdata3=zeros(Iter,IterNum);
Hdata4=zeros(Iter,IterNum);
Hdata5=zeros(Iter,IterNum);
Hdata6=zeros(Iter,IterNum);
Hdata7=zeros(Iter,IterNum);
Hdata8=zeros(Iter,IterNum);
Hdata9=zeros(Iter,IterNum);
Hdata10=zeros(Iter,IterNum);
Hdata11=zeros(Iter,IterNum);
HdataX=zeros(Iter,12);
ELMH=cell(Iter,IterNum);
ELMHsort=cell(Iter,IterNum);
HH=cell(Iter,IterNum);
HHsort=cell(Iter,IterNum);
filterHH=cell(Iter,IterNum);
filterHHsort=cell(Iter,IterNum);
for j=1:IterNum
    for i=1:Iter
        % Random generate input weights InputWeight (w_i) and biases
        BiasofHiddenNeurons (b_i) of hidden neurons
        InputWeight=rand(NumberofHiddenNeurons,NumberofInputNeurons)*2-1;
        BiasofHiddenNeurons=rand(NumberofHiddenNeurons,1);
        tempH=InputWeight*X'; %
        Release input of training data
        ind=ones(1,NumberofTrainingData);% generate 1*NumberofTrainingData 1
        matrix
        BiasMatrix=BiasofHiddenNeurons(:,ind); % Extend the bias
        matrix BiasofHiddenNeurons to match the demention of H
        tempH=tempH+BiasMatrix;
        % Calculate hidden neuron output matrix H

```

```

switch lower(ActivationFunction)

    case {'sig','sigmoid'}% Sigmoid

        H = 1 ./ (1 + exp(-tempH));

    case {'sin','sine'} % Sine

        H = sin(tempH);

    case {'hardlim'} % Hard Limit

        H = hardlim(tempH);% More activation functions can be added here

end

OutputWeight=T' * pinv(H);

Y=OutputWeight * H;          % Y: the actual output of the training data

TrainingAccuracyTEMP=sqrt(mse(T' - Y));    % Calculate training accuracy
(RMSE) for regression case

tempHdata=pinv(H)*H;  % sub-matrix

DtempHdata=diag(tempHdata);

DD=zeros(500,500);

for ii=1:500

    DD(ii,ii)=DtempHdata(ii,1);

end

LBtempHdata=tempHdata-DD;

% Adopt the filtering

HHfilter=LBtempHdata;

LineHHfilter=reshape(HHfilter,1,250000);

LineHHfilter02=smoothdata(LineHHfilter,'gaussian',10);% a is the data,
'gaussian' is the Gaussian filter, and 10 is the data window of 10

HHfilter=reshape(LineHHfilter02,500,500);

HHfilterdiag=diag(HHfilter);

DH=zeros(500,500);

for ii=1:500

    DH(ii,ii)=HHfilterdiag(ii,1);

```

```

end

comHH=HHfilter-DH+DD;

TrainingAccuracyFilter=sqrt(mse(T'*(eye(500)-comHH)));% new error

Hdata(i,j)=TrainingAccuracyTEMP;

Hdata2(i,j)=sum(sum(abs(tempHdata)));    % target matrix L1-Norm

Hdata3(i,j)=norm(tempHdata,'fro');    % F-Norm(L2-Norm), The sum of the
squares of the matrix elements and then the recalcitrance.

Hdata4(i,j)=norm(tempHdata,1);    % 1-Norm, The maximum value of the
sum of absolute values of matrix column vectors.

Hdata5(i,j)=sum(svd(tempHdata));    % Nuclear-Norm, Sum of singular
values of matrix

Hdata6(i,j)=sum((sum(tempHdata.*tempHdata,2)).^0.5); % L21-Norm

Hdata7(i,j)=sum(sum(abs(tempHdata),2).*sum(abs(tempHdata),2)).^0.5;    %
L12-Norm

Hdata8(i,j)=TrainingAccuracyFilter; % new error

Hdata9(i,j)=sum(sum(abs(comHH)));    % L1-Norm of the new target matrix

Hdata10(i,j)=sum((sum(comHH.*comHH,2)).^0.5);    % L21-Norm of the
new target matrix

Hdata11(i,j)=sum(sum(abs(comHH),2).*sum(abs(comHH),2)).^0.5;    % L12-
Norm of the new target matrix

ELMH{i,j}=H;

HH{i,j}=tempHdata;

filterHH{i,j}=comHH;

end

[Hdata(:,j),Index]=sort(Hdata(:,j));    % Error & L1-Norm. Every time the calculation
of i is completed, a sort is performed and an index is obtained.

Hdata2(:,j)=Hdata2(Index,j);    % Sort the target matrix according to the index
obtained above.

Hdata3(:,j)=Hdata3(Index,j);

Hdata4(:,j)=Hdata4(Index,j);

Hdata5(:,j)=Hdata5(Index,j);

```

```

Hdata6(:,j)=Hdata6(Index,j);

Hdata7(:,j)=Hdata7(Index,j);

Hdata8(:,j)=Hdata8(Index,j);

Hdata9(:,j)=Hdata9(Index,j);

Hdata10(:,j)=Hdata10(Index,j);

Hdata11(:,j)=Hdata11(Index,j);

HHsort(:,j)=HH(Index,j);% Each column is arranged in ascending order of error. The
first line has the smallest error

ELMHsort(:,j)=ELMH(Index,j);

filterHHsort(:,j)=filterHH(Index,j);

disp([i,j])

end

HdataX(:,1)=(sum(Hdata,2)./IterNum);

HdataX(:,2)=(sum(Hdata2,2)./IterNum);

HdataX(:,3)=(sum(Hdata3,2)./IterNum);

HdataX(:,4)=(sum(Hdata4,2)./IterNum);

HdataX(:,5)=(sum(Hdata5')./IterNum)';

HdataX(:,6)=(sum(Hdata6,2)./IterNum);

HdataX(:,7)=(sum(Hdata7,2)./IterNum);

HdataX(:,8)=(sum(Hdata8,2)./IterNum);

HdataX(:,9)=(sum(Hdata9,2)./IterNum);

HdataX(:,10)=(sum(Hdata10,2)./IterNum);

HdataX(:,11)=(sum(Hdata11,2)./IterNum);


figure(1)

subplot(2,3,1);

scatter(HdataX(:,2),HdataX(:,1),'*')

r01=corr(HdataX(:,2),HdataX(:,1),'type','pearson');

r001=corrcoef(HdataX(:,2),HdataX(:,1));

```

```

ylabel('ELM error');

xlabel('L1-Norm of the target matrix')% Find the linear relationship between the L1norm
of the target matrix and the ELM error.

str1={'Correlation coefficient:',r01};
text(1563.5,1.073,str1);

subplot(2,3,2);

scatter(HdataX(:,3),HdataX(:,1),'*')

r02=corr(HdataX(:,3),HdataX(:,1),'type','pearson');
r002=corrcoef(HdataX(:,3),HdataX(:,1));

ylabel('ELM error');

xlabel('F-Norm of the target matrix')

str2={'Correlation coefficient:',r02};
text(7.07106781178,1.066,str2);

subplot(2,3,3);

scatter(HdataX(:,4),HdataX(:,1),'*')

r03=corr(HdataX(:,4),HdataX(:,1),'type','pearson');
r003=corrcoef(HdataX(:,4),HdataX(:,1));

ylabel('ELM error');

xlabel('1-Norm of the target matrix')

str3={'Correlation coefficient:',r03};
text(6.27,1.072,str3);

subplot(2,3,4);

scatter(HdataX(:,5),HdataX(:,1),'*')

r04=corr(HdataX(:,5),HdataX(:,1),'type','pearson');
r004=corrcoef(HdataX(:,5),HdataX(:,1));

ylabel('ELM error');

```

```

xlabel('Nuclear-Norm of the target matrix')

str4={'Correlation coefficient:',r04};
text(50.0000004,1.072,str4);


subplot(2,3,5);
scatter(HdataX(:,6),HdataX(:,1),'*')
r05=corr(HdataX(:,6),HdataX(:,1),'type','pearson');
r005=corrcoef(HdataX(:,6),HdataX(:,1));
ylabel('ELM error');
xlabel('L21-Norm of the target matrix')
str5={'Correlation coefficient:',r05};
text(148.66,1.072,str5);


subplot(2,3,6);
scatter(HdataX(:,7),HdataX(:,1),'*')
r06=corr(HdataX(:,7),HdataX(:,1),'type','pearson');
r006=corrcoef(HdataX(:,7),HdataX(:,1));
ylabel('ELM error');
xlabel('L12-Norm of the target matrix')
str6={'Correlation coefficient:',r06};
text(71.23,1.072,str6);


figure(2)
subplot(1,3,1);
plot(HdataX(:,1),'r')
hold on
plot(HdataX(:,8),'*')
ylabel('value');
xlabel('ELM original error(-) & Error after filtering ( * ) ')

```

```

subplot(1,3,2);
plot(HdataX(:,6),'r')
hold on
plot(HdataX(:,10),'*')
ylabel('value');
xlabel('The L21-Norm of the original target matrix of ELM(-) & L21-Norm of the target
matrix after filtering ( * ) ')

```

```

subplot(1,3,3);
scatter(HdataX(:,10),HdataX(:,8),'*')
r07=corr(HdataX(:,10),HdataX(:,8),'type','pearson');
r007=corrcoef(HdataX(:,10),HdataX(:,8));
ylabel('ELM error after filtering');
xlabel('L21-Norm of the target matrix after filtering')
str7={'Correlation coefficient:',r07};
text(78.07,1.106,str7);

```