Cognitive Computation (2023) 15:496–519
https://doi.org/10.1007/s12559-022-10050-2

C-Loss-Based Doubly Regularized Extreme Learning Machine

Qing Wu[1] · Yan–Lin Fu[1] · Dong–Shun Cui[2] · En Wang[3]

Received: 17 August 2021 / Accepted: 14 August 2022 / Published online: 27 August 2022

Abstract
Extreme learning machine has become a significant learning methodology due to its efficiency. However, extreme learning machine may lead to overfitting since it is highly sensitive to outliers. In this paper, a novel extreme learning machine called the C-loss-based doubly regularized extreme learning machine is presented to handle dimensionality reduction and overfitting problems. The proposed algorithm benefits from both L_1 norm and L_2 norm and replaces the square loss function with a C-loss function. And the C-loss-based doubly regularized extreme learning machine can complete the feature selection and the training processes simultaneously. Additionally, it can also decrease noise or irrelevant information of data to reduce dimensionality. To show the efficiency in dimension reduction, we test it on the Swiss Roll dataset and obtain high efficiency and stable performance. The experimental results on different types of artificial datasets and benchmark datasets show that the proposed method achieves much better regression results and faster training speed than other compared methods. Performance analysis also shows it significantly decreases the training time, solves the problem of overfitting, and improves generalization ability.

Keywords Extreme learning machine · C-loss function · Feature selection · Regularization

Introduction

Processing a large quantity of data carries a high computational cost and slows the training process. To resolve these issues, a fast and stable algorithm needs to be proposed. In 1986, Rumelhart et al. [1] proposed the back propagation neural network (BPNN), which is a multilayer feedforward network for error correction. Support vector regression (SVR), used to minimize the generalization error bound so as to achieve generalized performance, was then presented by Vapnik et al. [2]. Single-layer feedforward neural networks (SLFNs) have a powerful nonlinear mapping capability and generally use the gradient descent algorithm to deal with the problems of classification and regression [3]. However, they have several disadvantages such as low training efficiency and being trapped easily in a local minimum. In 2006, extreme learning machine (ELM) for SLFNs was proposed by Huang et al. [4], which is still widely used in many research fields, such as foreign accent identification [5], fault detection [6], and emotion recognition [7]. Compared with traditional methods, such as the gradient descent algorithm, ELM can significantly increase training speed and improve generalization performance [8].

In ELM, the input network weights and hidden bias can be generated randomly. Meanwhile, the output network weights can be obtained by only calculating the Moore–Penrose inverse [9]. If the amplitude distribution of the singular value is relatively continuous and the minimum singular value is very close to 0, a large value of output weight vector will be obtained. Therefore, basic ELM, based on empirical risk minimization, leads to overfitting and affects prediction ability [10]. However, ELM uses the traditional least squares method to compute the output weight. As a convex function, the square loss function can cause outliers to sustain large losses because of unboundedness [11]. When outliers exist in the dataset, the approximation function of ELM may significantly deviate from the optimal function, resulting in poor generalization.

✉ Yan–Lin Fu
fuyanlin@stu.xupt.edu.cn

[1] School of Automation, Xi'an University of Posts and Telecommunications, Xi'an 710121, China

[2] School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore

[3] School of Marxism, Xi'an Shiyou University, Xi'an 710065, China

To overcome the above shortcomings, researchers have proposed several schemes. Deng et al. [12] put forward a regularized ELM embedded L_2 norm (L_2–ELM). The algorithm uses the weighted least squares method to obtain anti–noise ability by introducing the regularization factor γ. L_1–ELM with a sparse solution was proposed by Balasundaram et al. [13]. Clearly, L_1 norm is less sensitive to outliers than L_2 norm. The decision function of L_1–ELM uses a smaller number of hidden nodes than ELM. Martínez et al. [14] introduced L_1 norm and hybrid penalties to solve regression problems of ELM. The purpose of introducing different penalties is to moderate the detrimental effect of outliers. Taking the importance of features into consideration, the methods assign different weights to different features automatically. As a result, the smallest weight is assigned to outliers. An ELM model based on L_1 norm and L_2 norm regularizations (DRELM) is proposed to handle regression and multiple-class classification problems [15]. It is robust in both regression and classification applications. In 2014, the C-loss function for pattern classification was presented by Abhishek et al. [16]. The proposed loss function can improve the performance of neural network classifiers. In fact, the paper just introduces the C-loss function for only classification problems. Zhao et al. [17] offered an algorithm named C-loss-based extreme learning machine (CELM). Although CELM has good generalization performance, it has difficulty solving the problem of overfitting.

More recently, other alternative methods were proposed to eliminate the distraction caused by outliers. Jing et al. [18] proposed domain-invariant feature learning framework for partial domain adaptation. Fu et al. [19] developed a novel model termed partial feature selection and alignment by employing a feature selection vector based on the correlation among the features of multiple sources and target domains. Both of them show that re-weighting and feature selection can eliminate the distraction caused by outliers. However, they mainly tackled distribution shift and label shift problems.

To develop a more stable, stronger anti-interference and faster algorithm, we propose a doubly regularized ELM based on C-loss function called CDRELM. The proposed algorithm replaces the square loss function with C-loss function and embeds L_1 norm and L_2 norm on ELM. L_1 norm has the ability to reduce the feature dimension of samples. Therefore, CDRELM can not only deal with regression problems with strong generalization performance but also, as a method of feature selection, can decrease the dimension at high speed. CDRELM tends to be more robust and achieves much better generalization with a faster learning speed than L_2–ELM, L_1–ELM, CELM, DRELM, BPNN, and SVR. To find solutions for this mathematical model, CDRELM is transformed into least absolute shrinkage and selection operator (Lasso) [20]. The three main contributions in this paper are as follows:

1. The C-loss function is used for regression problems rather than classification problems. To overcome the unsteadiness of the square loss function to outliers, the square loss function used in ELM is replaced by the C-loss function which is bounded, non-convex, and smooth. Thus, a novel algorithm CDRELM is proposed based on the C-loss function. In comparison with the traditional ELM, CDRELM overcomes the problem of overfitting and the insufficient robustness to outliers, which greatly improve the generalization capability.
2. As a new method of feature selection which simultaneously allows feature selection and the training process, CDRELM can generate sparse eigenvalues by embedding the L_1 norm. In addition, the L_2 norm is added to maintain the amplitude of output weight sparsity and avoid increased sparsity. It can solve regression problems much faster with its ability of dimension reduction. It can also reduce the computational cost and process high-dimensional datasets efficiently.
3. The new mathematical model is transformed into a Lasso problem for calculating the results. According to the proximal gradient descent (PGD) algorithm [21], an improved operator replaces the original operator to solve the Lasso problem. Compared with PGD, the new improved method can obtain the solution fast and efficiently decrease the number of iterations. It can also compute the solution, which is applied to various datasets with fast and accurate performance.

The rest of this paper is organized as follows. "Related Work" introduces the related work, including ELM, C-loss function, and proximal gradient descent algorithm. In "Proposed CDRELM Method," the novel algorithm CDRELM, including a mathematical model, solution, and computational complexity analysis, is presented. The proposed algorithm can not only possess the nonconvex and bounded loss function with robustness to outliers but embed L_1 norm and L_2 norm to carry out feature selection at high speed. CDRELM can be solved by an improved alternating optimization method. To test the effectiveness of the proposed CDRELM, "Experiments and Discussion" presents the experimental results including improved solution, dimensionality reduction, and regression. "Performance for Regression" shows four artificial datasets and five benchmark datasets. The Friedman and Nemenyi tests are also shown for comparative analysis. "Conclusion" presents conclusions and future work.

Related Work

ELM

As a single-hidden-layer feedforward neural network, ELM plays a key role in academia and industry. The development of SLFNs has enabled ELM to reach enhanced generalization performance for classification and regression at high speed.

In SLFN, for Q arbitrary distinct samples (x_i, t_i), where $x_i = [x_{i1}, x_{i2}, \ldots, x_{im}]^T \in R^m$ and $t_i = [t_{i1}, t_{i2}, \ldots, t_{in}]^T \in R^n$, the relationship between input x_i and output $f(x_i)$ is given as follows:

$$f(x_i) = \sum_{j=1}^{P} \beta_j G(\varpi_j, b_j, x_i) = \sum_{j=1}^{P} \beta_j G(\varpi_j \cdot x_i + b_j), \quad i = 1, 2, \ldots, Q, \tag{1}$$

where $\varpi_j = [\varpi_{j1}, \varpi_{j2}, \ldots, \varpi_{jn}]^T$ and $b_j = [b_{j1}, b_{j2}, \ldots, b_{jn}]^T$ are the randomly generated learning parameters of hidden nodes; $\beta_j = [\beta_{j1}, \beta_{j2}, \ldots, \beta_{jn}]^T$ is the weight connecting the j-th hidden node and the output nodes; $G(\cdot)$ represents the activation function; P is number of hidden nodes; Q is the number of datasets. The output function of ELM is expressed as follows:

$$H\beta = y \tag{2}$$

Here, $\beta = [\beta_1, \beta_2, \ldots, \beta_P]^T$ is the matrix of output weights and $y = [y_1, y_2, \ldots, y_Q]^T$ is the matrix of targets. The hidden-layer output matrix is as follows:

$$H = \begin{bmatrix} G(\varpi_1, b_1, x_1) & \ldots & G(\varpi_P, b_P, x_1) \\ \vdots & \ddots & \vdots \\ G(\varpi_1, b_1, x_Q) & \cdots & G(\varpi_P, b_P, x_Q) \end{bmatrix} \tag{3}$$

The value of the output weights β can be determined by calculating the linear system Eq. (2) as follows:

$$\beta = \mathrm{H}^{+} y \tag{4}$$

where H^+ is the Moore–Penrose generalized inverse matrix H [22]. ELM computes H^+ in Eq. (4) based on the singular value decomposition (SVD) of H.

C-loss Function

There are several loss functions such as hinge loss function [23], $\psi-$ learning loss function [24], normalized sigmoid loss function [25], and ramp loss function [26]. Compared with square loss function, these loss functions perform better in enhancing the robustness because of nonconvexity and boundedness. To find a better loss function, Abhishek et al. [16] proposed the C-loss function defined by the following:

$$l_C(\theta) = 1 - \exp\left\{-\frac{\theta^2}{2\sigma^2}\right\} \tag{5}$$

where $\theta = y - f(x)$ is the space of errors, and σ is window width. The comparison of various loss functions is depicted in Fig. 1.

Compared with the other loss functions, the C-loss function is bounded, nonconvex, and smooth being more stable to outliers. C-loss can process all sizes of errors for classification problems. In this paper, we introduce the C-loss function to a doubly regularized ELM for regression problems.

Proximal Gradient Descent Algorithm

In 2004, Boyd et al. [21] proposed PGD to solve the problems of L_1 regularization. It is an effective and rapid solution to Lasso problems in many applications.

Let ∇ be a differential operator. The optimization objective is as follows:

$$\min_x g(x) + \eta \|x\|_1 \tag{6}$$

If $g(x)$ is derivative and ∇g meets the condition of *L–Lipschitz*,

$$\exists L \in R^+, \left\|\nabla g(x') - \nabla g(x)\right\|_2^2 \le L\|x' - x\|_2^2 \quad (\forall x, x') \tag{7}$$

In the neighborhood of x_k, the $g(x)$ can be approximately calculated by second-order Taylor expansion as follows:

$$\begin{aligned} \hat{g}(x) &\simeq g(x_k) + \left\langle \nabla g(x_k), x - x_k \right\rangle + \frac{L}{2}\|x - x_k\| \\ &= \frac{L}{2}\left\|x - \left(x_k - \frac{1}{L}\nabla g(x_k)\right)\right\|_2^2 + const \end{aligned} \tag{8}$$

where *const* is a constant, and $\langle\cdot\rangle$ is inner product. The minimum value of Eq. (8) can be obtained from the following:

$$x_{k+1} = x_k - \frac{1}{L}\nabla g(x_k) \tag{9}$$

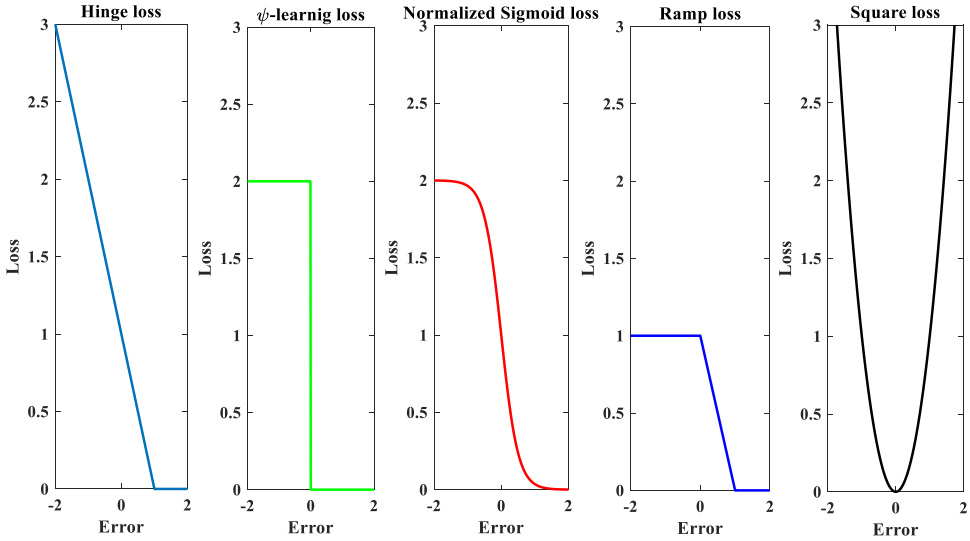
Gradient descent can be adopted to minimize $g(x)$. Each step of gradient descent iteration is equivalent to minimizing the quadratic function $\hat{g}(x)$. According to Eq. (6), each iteration step is similarly shown as follows:

$$x_{k+1} = \arg\min_x \frac{L}{2}\left\|x - \left(x_k - \frac{1}{L}\nabla g(x_k)\right)\right\|_2^2 + \eta\|x\|_1 \tag{10}$$

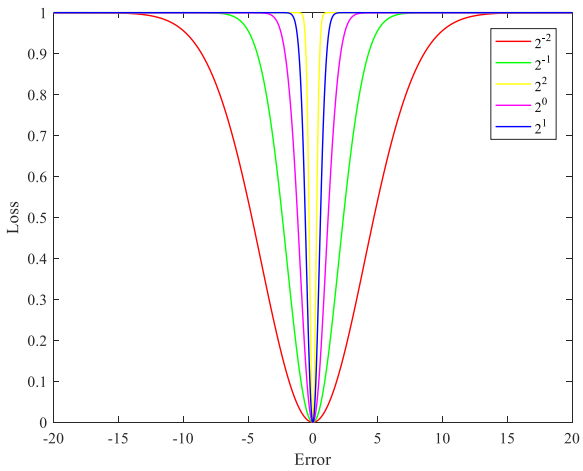
Each step of gradient descent iteration for $g(x)$ should consider minimizing the L_1 norm at the same time.

For Eq. (10), let $h = x_k - \frac{1}{L}\nabla g(x_k)$ and let x^i be the i-th component of x. Then, the closed-form solution is written as follows:

Fig. 1 Comparison of loss functions



(a) Hinge loss, ψ- learning loss, normalized sigmoid loss, ramp loss, and square loss



(b) C–loss with different window width σ

$$x_{k+1}^{i} = \begin{cases} h^{i} - \eta/L, & \eta/L < h^{i} \\ 0, & \left|h^{i}\right| \leq \eta/L \\ h^{i} + \eta/L, & h^{i} < -\eta/L \end{cases} \tag{11}$$

where x_{k+1}^{i} and h^{i} are the i-th component of x_{k+1}^{i} and h, respectively.

Proposed CDRELM Method

Our framework adopts bounded, nonconvex, and smooth C-loss function, leading to processing the outliers successively. In addition, CDRELM based on L_1 and L_2 regularization can complete the feature selection and the training process simultaneously, which greatly decreases the training time. Therefore, CDRELM can be considered a new method of embedded feature selection that is fast and offers stable performance.

Mathematical Model

We know that the regression problems investigate the relationship between the prediction and the target. To solve these problems, factors such as prediction accuracy, time, robustness, and size of model should be considered.

A single-output regression problem is formulated as follows:

$$y = H\beta + \theta \tag{12}$$

where $\beta = \left[\beta_1, \beta_2, \ldots, \beta_P\right]^T$ is the regression weights and $\theta = \left[\theta_1, \theta_2, \ldots, \theta_Q\right]^T$ is the loss between prediction value and target value. The input of the problem H is a $Q \times P$ matrix that can be described as follows:

$$H = \begin{bmatrix} h_{11} & \cdots & h_{1P} \\ \vdots & \ddots & \vdots \\ h_{Q1} & \cdots & h_{QP} \end{bmatrix} \tag{13}$$

The traditional solution is estimated by square loss function and can be defined as follows:

$$\hat{\beta} = \arg\min_{\beta} \frac{\|y - H\beta\|_2^2}{2} \tag{14}$$

where $\hat{\beta} = \left[\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_P\right]^T$ is the vector of estimated regression weights. Square loss function is convex and unbounded. However, C-loss, which is smooth, bounded, and nonconvex, can improve robustness and reduce overfitting. To overcome the instability of square loss to outliers, the square loss function is replaced with C-loss function. Then Eq. (14) can be transformed as follows:

$$\hat{\beta} = \arg\min_{\beta} \left\{ 1 - \exp\left\{ -\frac{(y - H\beta)^2}{2\sigma^2} \right\} \right\} \tag{15}$$

It is known that large-scale datasets lead to high computational cost. As a regularization technique, L_1 norm has been proposed to sparse eigenvalue and enhance generalization ability by shrinking some coefficients and setting others to 0. The Lasso estimate is shown as follows:

$$\hat{\beta} = \arg\min_{\beta} \left\{ 1 - \exp\left\{ -\frac{(y - H\beta)^2}{2\sigma^2} \right\} + \eta\|\beta\|_1 \right\} \tag{16}$$

where η is a positive regularization parameter and $\|\cdot\|_1$ is the L_1 norm. The value of η is positively associated with the number of nonzero components of $\hat{\beta}$.

Nevertheless, Zou et al. [27] noted that in the situation of $Q < P$, Lasso can only select at most Q variables. For the general situation of $Q > P$, Lasso cannot behave well if there are high correlations between prediction targets. To solve this problem, the L_2 norm is added to the mathematic model. Therefore, the amplitude of output weight $\hat{\beta}$ maintains sparsity while avoiding "over–sparsity." The modified system can now be expressed as follows:

$$\hat{\beta} = \arg\min_{\beta} \left\{ 1 - \exp\left\{ -\frac{(y - H\beta)^2}{2\sigma^2} \right\} + \eta\|\beta\|_1 + \xi\|\beta\|_2^2 \right\} \tag{17}$$

where ξ is a L_2 norm regularization parameter and $\|\bullet\|$ is L_2 norm. According to [12–15], a 2-norm regularization parameter is introduced to ELM, creating the model L_2–ELM with good generalization performance and strong control ability. Compared with ELM, L_2–ELM realizes the limitation of model space and avoids overfitting. ELM is embedded with a 1-norm regularization parameter, giving the model L_1–ELM fast learning speed. L_1–ELM can achieve sparsity and has good optimization solution characteristics. DRELM can control the complexity of the network and prevent overfitting. In our proposed mathematic model, C-loss function increases the robustness to the outliers, L_1 norm offers an automatic variable selection through a sparse vector, and L_2 norm strengthens the control ability. All the components of the process are performed simultaneously which significantly decreases the time and obtains strong generalization.

It is clear that Eq. (17) is nonconvex and cannot use the traditional optimization algorithm to solve CDRELM. Therefore, it is necessary to develop a more efficient method for solving CDRELM.

Solution

Based on the mathematical model of CDRELM, it can be transformed into an equivalent Lasso problem [20]. According to the proximal gradient descent (PGD) algorithm [21], an improved operator replaces the original operator to solve the Lasso problem. In this paper, the improved PGD is used to compute $\hat{\beta}$ of CDRELM.

Let ∇ be a differential operator. The optimization objective of CDRELM is as follows:

$$J(\beta) = \min_{\beta} \left\{ 1 - \exp\left\{ -\frac{(y - H\beta)^2}{2\sigma^2} \right\} + \eta\|\beta\|_1 + \xi\|\beta\|_2^2 \right\} \tag{18}$$

Let $\lambda_k = \beta_k + \frac{k(\beta_k - \beta_{k-1})}{k+5}$, where β_k is the k-th step of β and the initial β_0 and β_1 are both equal to 0 with the size of $n \times 1$. Replacing β_k with λ_k, it decreases the difference between the next gradient updating direction and the current gradient direction.

$$g(\lambda) = 1 - \exp\left\{ -\frac{(y - H\lambda)^2}{2\sigma^2} \right\} + \xi\|\lambda\|_2^2 \tag{19}$$

$$\nabla g = -\frac{H^T(y - H\lambda)}{\sigma^2} \cdot \left(-\exp\left\{ -\frac{(y - H\lambda)^2}{2\sigma^2} \right\} \right) + 2\xi\lambda \tag{20}$$

$g(\lambda)$ is differentiable and ∇g meets the condition of L—Lipschitz,

$$\exists L \in R^+, \left\| \nabla g(\lambda') - \nabla g(\lambda) \right\|_2^2 \le L\|\lambda' - \lambda\|_2^2 \ \ (\forall \lambda, \lambda') \tag{21}$$

In the neighborhood of λ_k, the $g(\lambda)$ can be approximately calculated by second-order Taylor expansion as follows:

$$\begin{aligned} \hat{g}(\lambda) &\simeq g(\lambda_k) + \left\langle \nabla g(\lambda_k), \lambda - \lambda_k \right\rangle + \frac{L}{2}\|\lambda - \lambda_k\|^2 \\ &= \frac{L}{2}\left\| \lambda - \left(\lambda_k - \frac{1}{L}\nabla g(\lambda_k) \right) \right\|_2^2 + const \end{aligned} \tag{22}$$

where *const* is a constant and $\langle\cdot\rangle$ is inner product. The minimum value of Eq. (22) can be obtained from the following:

$$\lambda_{k+1} = \lambda_k - \frac{1}{L}\nabla g(\lambda_k) \tag{23}$$

Gradient descent can be adopted to minimize $g(\lambda)$. Each step of gradient descent iteration is equivalent to minimizing the quadratic function $\hat{g}(\lambda)$. Let this method be extended to Eq. (18). Then, each iteration step is similarly shown as follows:

$$\lambda_{k+1} = \arg\min_{\lambda} \frac{L}{2}\left\|\lambda - \left(\lambda_k - \frac{1}{L}\nabla g(\lambda_k)\right)\right\|_2^2 + \eta\|\lambda\|_1 \tag{24}$$

Namely, each step of gradient descent iteration for $g(\lambda)$ should consider minimizing the L_1 norm at the same time.

For Eq. (24), let $h = \lambda_k - \frac{1}{L}\nabla g(\lambda_k)$ and let λ^i be the i-th component of λ. Then, compute $\lambda_{k+1} = \arg\min_x \frac{L}{2}\|\lambda - h\|_2^2 + \eta\|\lambda\|_1$. The closed-form solution is written as follows:

$$\lambda_{k+1}^i = \begin{cases} h^i - \eta/L, & \eta/L < h^i \\ 0, & |h^i| \le \eta/L \\ h^i + \eta/L, & h^i < -\eta/L \end{cases} \tag{25}$$

Here, λ_{k+1}^i and h^i are the i-th component of λ_{k+1} and h, respectively.

The CDRELM algorithm shown below includes the process of modeling and solving the mathematical model.

Table 1 Data comparison between PGD and improved PGD

Algorithm	Time(s)	Iterations	Optimal value
PGD	0.3243	100	−1.412
Improved PGD	0.0925	59	−1.574

Computational Complexity Analysis

In this section, we analyze the computational complexity of CDRELM.

For the matrix $H \in R^{Q\times P}$, where P is the number of hidden nodes and Q is the number of datasets, the computational complexity of SVD is $O(4QP^2 + 8P^3)$ [28]. As mentioned in "ELM," ELM computes its output weights based on the SVD of $H \in R^{Q\times P}$, so that the computational complexity of ELM is approximately the same as SVD.

According to Eq. (19), the computational complexity of each iteration step is also $O(4QP^2 + 8P^3)$. If we assume that the method converges after K-th iterations, the overall computational time complexity is $K * O(4QP^2 + 8P^3)$.

Experiments and Discussion

We conducted experiments to verify the performance of the presented algorithm. "Performance of Improved PGD" shows the comparison between the traditional PGD and the improved PGD. The performance in dimensionality

Algorithm: CDRELM

Input: a training set: $\{(x_i, y_i)|x_i \in R^m, y_i \in R^n,\ i = 1,\ldots,Q\}$;

related parameter: the number of hidden nodes P; activation function $G(x)$;

L_1 norm term η; L_2 norm term ξ; parameter $\psi \in (0,1)$; window width σ

Output: the output weight matrix β

1. Randomly generate learning parameters of hidden nodes $\varpi_j, b_j, 1 \le j \le P$.
2. Compute the hidden layer output matrix H based on Eq. (3).
3. Let $g(\beta) = 1 - \exp\left\{-\frac{(y - H\beta)^2}{2\sigma^2}\right\} + \xi\|\beta\|_2^2$.
4. Initialize $L_{k-1} \in R^n$, $\beta_0 \in 0^{n\times 1}$ and $\beta_1 \in 0^{n\times 1}$.
5. Let step length $L := L_{k-1}$

while $|J(\beta_k) - J(\beta_{k-1})| \le 10^{-4}$ $(k > 1)$

set $\lambda_k = \beta_k + \frac{k(\beta_k - \beta_{k-1})}{k+5}$, $h = \lambda_k - \frac{1}{L}\nabla g(\lambda_k)$

do Eq. (25)

while $g(h) \le g(\lambda_k) + \nabla g(\lambda_k)^T(h - \lambda_k) + \frac{L}{2}\|\lambda - \lambda_k\|^2$

$\lambda_{k+1} = \arg\min_{\lambda} \frac{L}{2}\|\lambda - h\|_2^2 + \eta\|\lambda\|_1$

update $L := \psi L$

return $L_k := L$

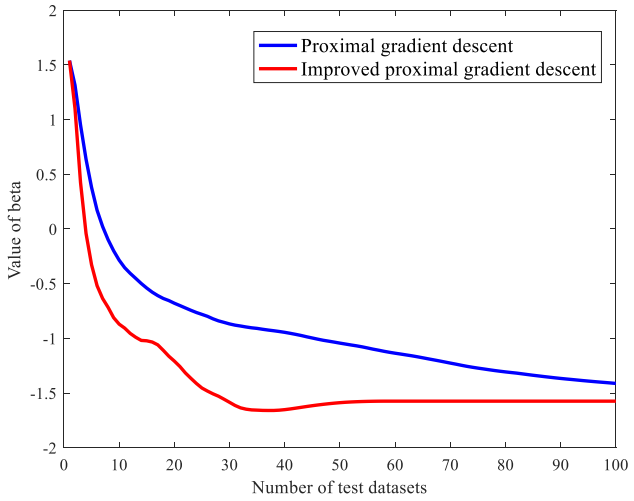


Fig. 2 Comparison between PGD and improved PGD

reduction is given in "Performance on Dimensionality Reduction." "Performance for Regression" details the performance for regression. We evaluated related algorithms, including L_2–ELM, L_1–ELM, CELM, DRELM, BPNN, and SVR, by using different types of datasets and two activation functions. All experiments were performed in MATLAB R2016a on a desktop computer with an Intel Core i7 1160G7 CPU at 2.11 GHz, 16 GB of memory, and Windows 10.

Performance of Improved PGD

The input $x_i = \left[x_{i1}, x_{i2}, \ldots, x_{i500}\right]^T \in R^{500}\ i = 1, \ldots, 2500$ of CDRELM is generated randomly, where $x_{ij} \in (0, 1)$. According to Eq. (18), we can obtain β by using PGD and improved PGD.

Compared with the original PGD, the improved PGD can reach the optimal value fast and requires only 59 iterations. Moreover, the optimal value of β, which is calculated by improved PGD, obtains the fitter results. Table 1 shows the data comparison between PGD and improved PGD, including time, iterations, and optimal value. The visual results between two methods are plotted in Fig. 2.

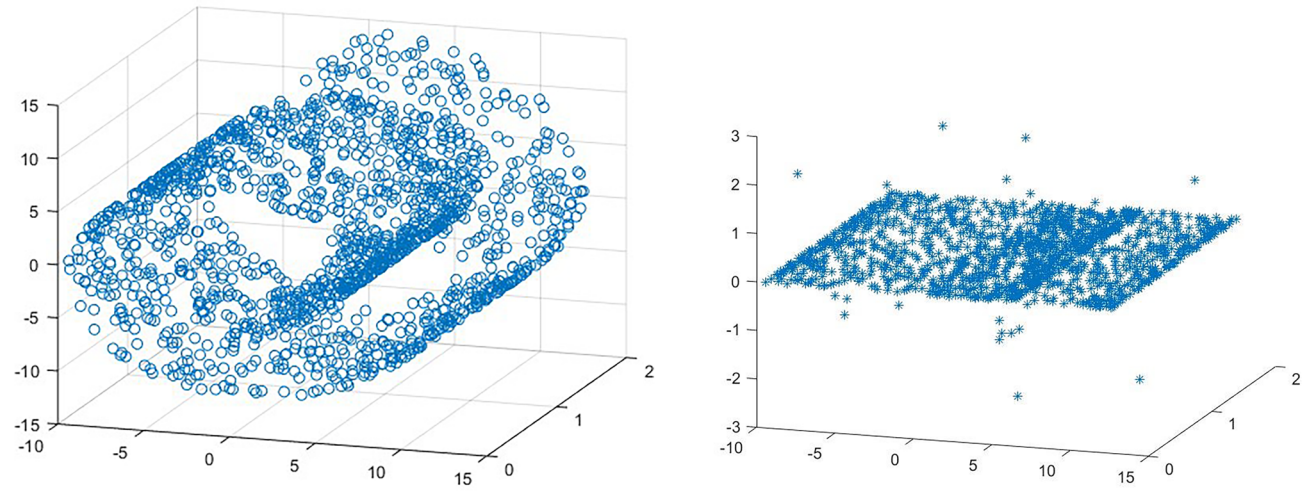
Performance on Dimensionality Reduction

As a new method of embedded feature selection, CDRELM can automatically select the feature to reduce the dimension of the sample dataset and predict samples simultaneously. It can make eigenvalue sparse and enhance generalization ability by shrinking some coefficients and setting others to zero. It also reduces computational complexity and improves computational efficiency by decreasing the number of β.

The Swiss Roll dataset was created to verify different dimensionality reduction algorithms [29]. r and l return two arrays of random numbers generated from the continuous uniform distributions with lower and upper endpoints specified by 0 and 1, respectively. The data on the coordinate axis is generated from the following:

$$t = \frac{3\pi}{2(l + 2r)}$$
$$\begin{cases} x = t * \cos(t) \\ y = 2l \\ z = t * \sin(t) \end{cases} \tag{26}$$

The comparison of before and after the dimensionality reduction using CDRELM is shown in Figs. 3 and 4. Figure 3 signifies the situation of scatter which adopts



(a) Before dimension reduction **(b)** After dimension reduction

Fig. 3 Comparison of value on z-axis before and after dimension reduction using CDRELM

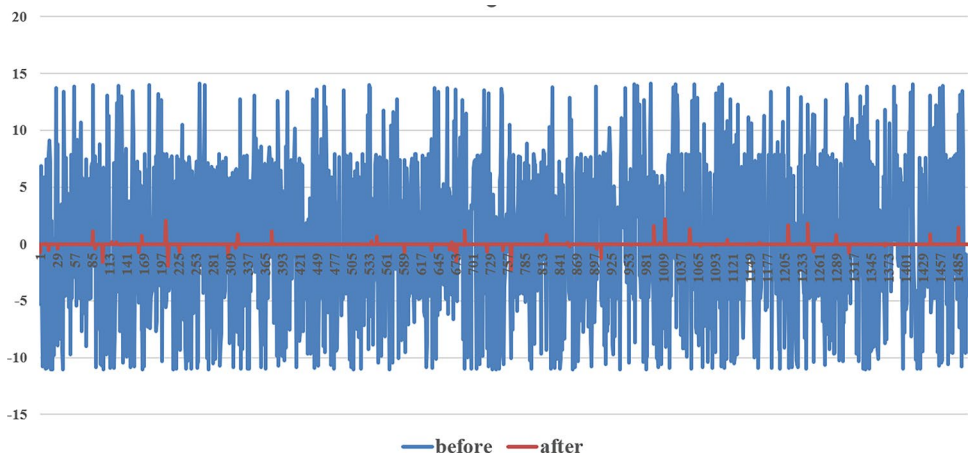


Fig. 4 Performance of dimensionality reduction based on CDRELM

CDRELM to verify the feature selection effect of the proposed method. In Fig. 4, the scatter value on the z-axis which belongs to Fig. 3 is located on the y-axis. The x-axis denotes the amount of scatter in Fig. 3.

From the experimental results, it is obvious that CDRELM can achieve significant dimensionality reduction, including reduced computational complexity and increased efficiency. From Figs. 3 and 4, it can be seen that CDRELM can not only narrow the range but also set some data to 0. Obviously, the impact of dimension reduction on the z-axis is significant.

Performance for Regression

Four artificial datasets and five benchmark datasets from UCI machine learning repository [30] and Kaggle [31] were used to test the proposed algorithm CDRELM. To evaluate the performance of CDRELM, it was compared with six algorithms: L_2–ELM, L_1–ELM, CELM, DRELM, BPNN, and SVR. In the experiments, two activation functions including sigmoid and sine were used on different datasets. Several parameters needed to be adjusted: L_1 norm term, L_2 norm term, window width σ of C-loss function, and the number of hidden layer nodes P. Taking "sinc function datasets" as examples, we analyzed the sensitivity of CDRELM to the number of hidden layer nodes P. In Fig. 5, with the number of hidden layer nodes increasing, the R^2 has no obvious change. The numbers of hidden layer nodes selected were 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, and 400. During the experiments, we fixed the number of hidden layer nodes at 20 and combined the grid search with a cross-validation technique to select the best parameters. Using the best parameters, we performed the experiments 30 times and reported the results with variability information.

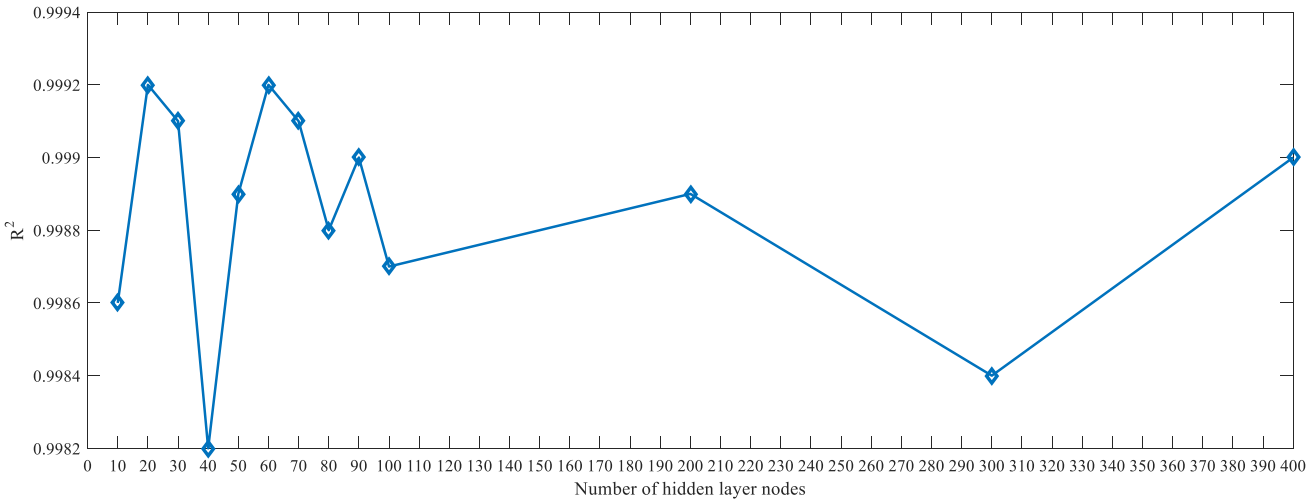


Fig. 5 Relationship between the number of hidden layer nodes and R^2 on sinc function datasets

Table 2 Functions used for generating regression datasets

Dataset	Function definition
Sinc function	$40\sin c(0.4x) = \begin{cases} \frac{40\sin(0.4x)}{x} & if\ x \neq 0 \\ 40 & else \end{cases}$
Linear regression	$y = kx$
Self-defining function	$y = \exp\{0.35x * \sin(x)\}$
Two-moon	$r \sim U\left(r - \frac{w}{2}, r + \frac{w}{2}\right)$
	$\theta_1 \sim U(0,\pi) \quad \theta_2 \sim U(-\pi, 0)$

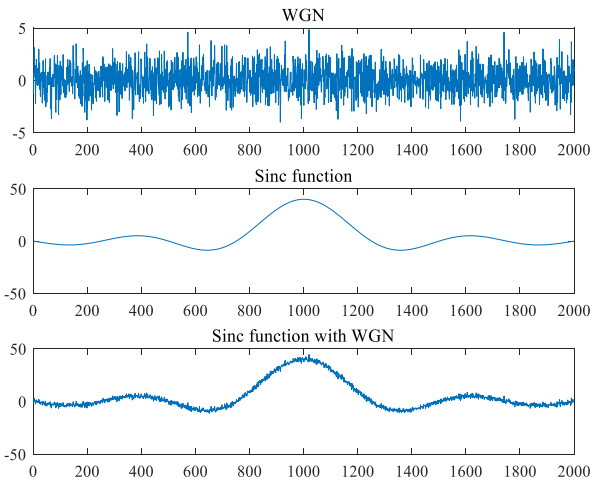
Table 3 Details of artificial datasets

Dataset	Number of training samples	Number of testing samples	Range of independent variables
Sinc function	1800	201	$x \in (-10, 10)$
Linear regression	1800	201	$x \in (-10, 10)$
Self-defining function	1800	201	$x \in (-10, 10)$
Two-moon	1800	201	$x_1 \in (0, 3)$
			$x_2 \in (1.5, 4.5)$
	1800	201	

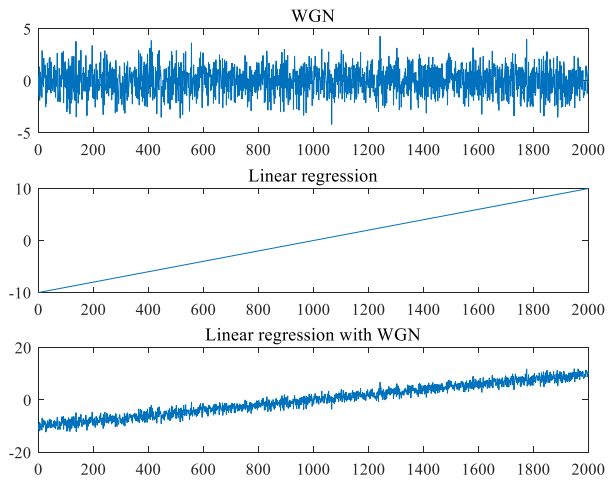
Two performance indices mean squared errors (MSE) and determination coefficient (R^2) are defined by the following:

$$MSE = E\left(\hat{y}_i - y_i\right)^2 \quad i = 1, \ldots, Q \tag{27}$$

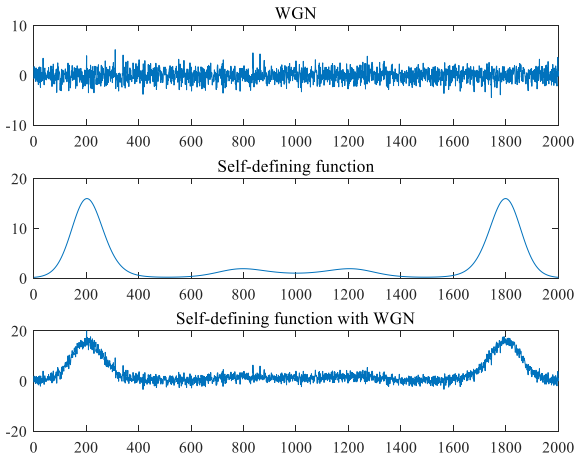
$$R^2 = \frac{\left(l\sum_{i=1}^{l}\hat{y}_i y_i - \sum_{i=1}^{l}\hat{y}_i \sum_{i=1}^{l} y_i\right)^2}{\left(l\sum_{i=1}^{l}\hat{y}_i^2 - \left(\sum_{i=1}^{l}\hat{y}_i\right)^2\right)\left(l\sum_{i=1}^{l} y_i^2 - \left(\sum_{i=1}^{l} y_i\right)^2\right)} \tag{28}$$



(a) Sinc function

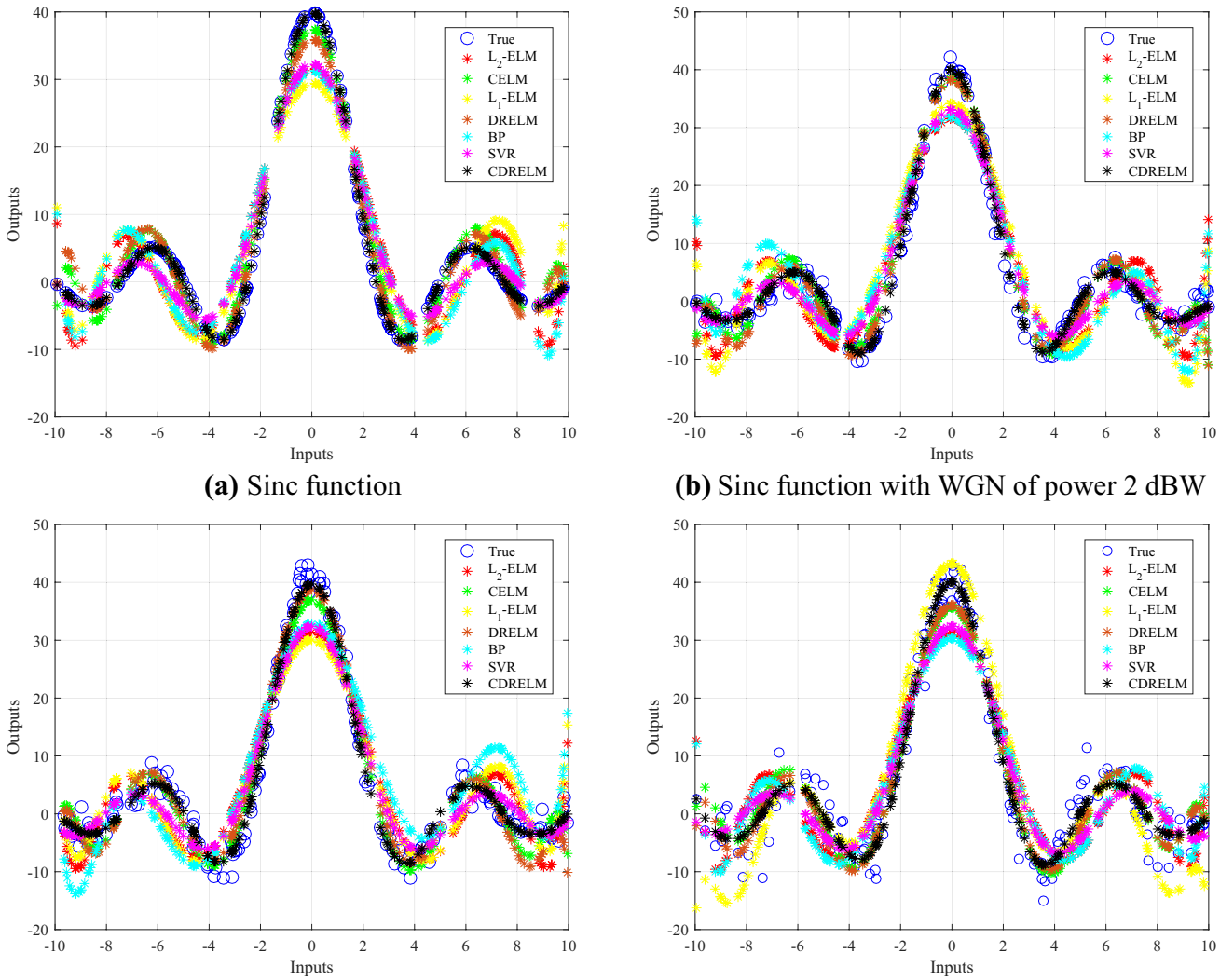


(b) Linear regression



(c) Self-defining function

Fig. 6 Regression shapes of three functions with WGN



(a) Sinc function

(b) Sinc function with WGN of power 2 dBW

(c) Sinc function with WGN of power 5 dBW

(d) Sinc function with WGN of power 10 dBW

Fig. 7 Regression results on sinc function and sinc function with WGN

where $\hat{y}_i$ represents the prediction of the desired y_i, and l is the number of testing samples. A smaller MSE or a larger performance index R^2 reflects better generalization performance.

Here, we use two activation functions for comparing L_2–ELM, L_1–ELM, CELM, DRELM, and CDRELM on the same datasets.

Sigmoid function:

$$F(a,b,x) = \frac{1}{1+\exp\left(-a_i^T x + b_i\right)} \tag{29}$$

Sine function:

$$F(a,x) = \sin\left(a_i^T x\right) \tag{30}$$

To achieve good generalization performance, the appropriate optimization parameter needs to be chosen. We combined grid search with a cross-validation technique to choose these parameters [32]. The regularization parameters λ for L_1–ELM, ξ for L_2–ELM, and λ and ξ for DRELM are all determined from the parameter set $\{2^{-50}, \ldots, 2^0, \cdots, 2^{20}\}$. In CELM, the window width σ and the regularization parameter λ are chosen from the candidate set $\{2^{-2}, \ldots, 2^0, \cdots, 2^2\}$ and $\{2^{-50}, \ldots, 2^0, \cdots, 2^{20}\}$, respectively. In CDRELM, the regularization parameters η and ξ are selected from $\{2^{-50}, \ldots, 2^0, \cdots, 2^{20}\}$ and $\{2^{-50}, \ldots, 2^0, \cdots, 2^{20}\}$, and the window width σ is taken from $\{2^{-2}, \ldots, 2^0, \cdots, 2^2\}$. In BPNN, we set the goal accuracy as $1e-3$ and the maximum iterations as 1500. The number of input layer nodes is equal to the number of input variables. The number of output layer

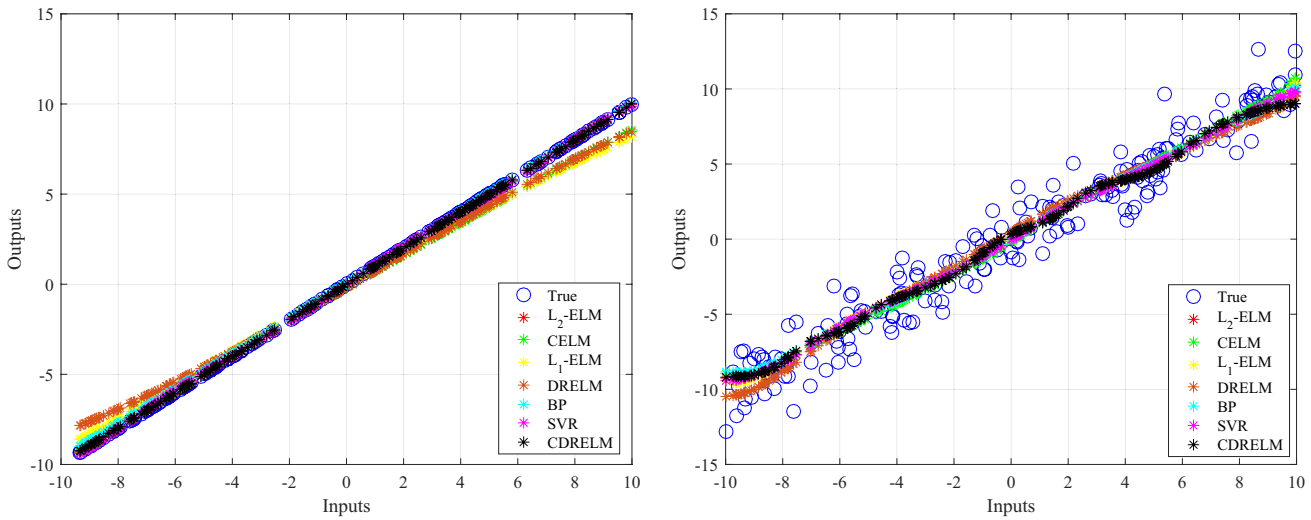
nodes is set as 1. According to the MSE calculated using the trial-and-error method, the learning rate and the number of hidden layer nodes are selected according to different datasets. They are selected as 0.003 and 7 on artificial datasets. Meanwhile, the two parameters of learning rate and the number of hidden layer nodes are set as 0.009 and 3 on the octane number dataset, 0.008 and 7 on the Boston housing dataset, 0.008 and 11 on the life expectancy dataset, 0.005 and 7 on the energy consumption dataset, and 0.003 and 4 on the air quality dataset, respectively. The penalty parameter C for error entries of SVR affects the accuracy and generalization ability [2]. The penalty parameter C is taken from $\{2^{-50}, \ldots, 2^{0}, \cdots, 2^{20}\}$ and the radial basis kernel function $K(\mathbf{x}, \mathbf{x}_i) = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\theta^2}\right\}$ is used in SVR where the parameter θ is selected from $\{2^{-2}, \ldots, 2^{0}, \cdots, 2^{10}\}$.

Performance on Artificial Datasets

Three artificial datasets with white Gaussian noise (WGN) and two-moon datasets were generated to verify the prediction accuracy and generalization ability. The power spectral density of WGN obeys uniform distribution, and the amplitude distribution obeys Gaussian distribution [33]. Based on the traditional function, WGN with the form of 2001×1 was added to the three functions whose definitions are given in Table 1. Moreover, in WGN, the power of each function in decibels relative to a watt is 2 dBW. In particular, to verify the effect of outliers on the comparison of performance and the relationship between outliers and parameters, the performance of seven algorithms on sinc function datasets with WGN of power 0 dBW, 2 dBW, 5 dBW, and 10 dBW is shown in Fig. 7 and Table 4.

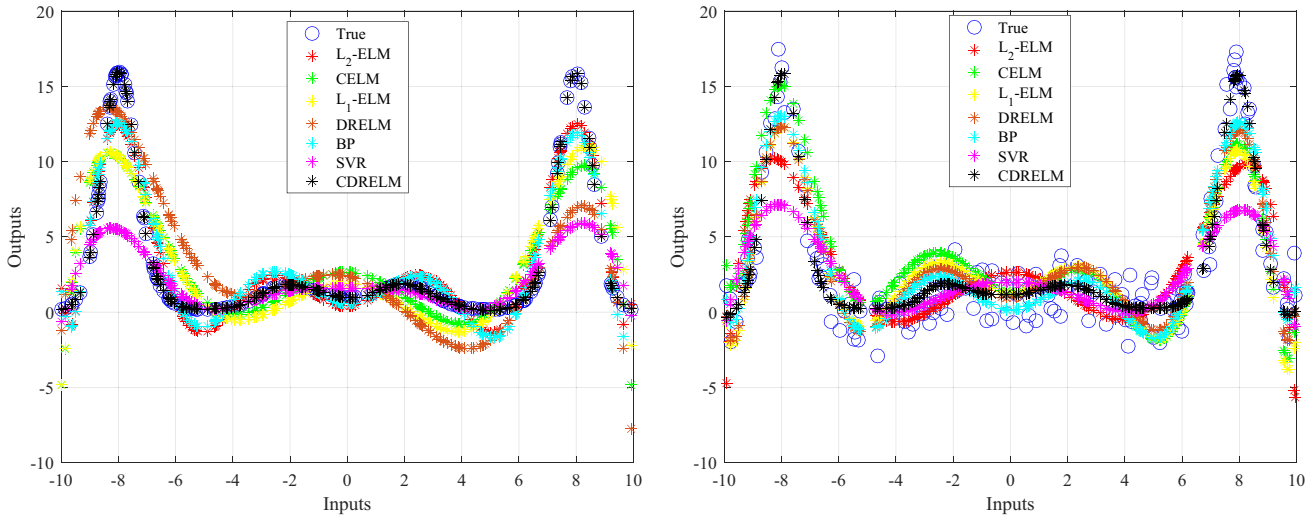
The details of the experiments on the four datasets are listed in Tables 2 and 3. For artificial datasets, almost nine-tenths of the whole dataset are used as the training set, and the remaining one-tenth is used as the testing set. Figure 6 shows regression shapes of three functions with the WGN of power 2 dBW. The regression results of sinc function, linear regression function, self-defining function, and two-moon are plotted in Figs. 7, 8, 9, and 10, respectively. Similarly, the experimental results are given in Tables 4, 5, 6, and 7, respectively.

It can be seen from Figs. 7, 8, 9, and 10 and Tables 4, 5, 6, and 7 that CDRELM achieves comparable performance to other algorithms with a much higher learning speed due to its ability in dimensionality reduction. It is clear that CDRELM takes much less time than other algorithms. In particular, the proposed algorithm can achieve better generalization performance and more significant efficiency than BPNN and SVR. In most cases, the proposed CDRELM obtains the largest R^2 and the smallest MSE, which shows the most robustness and highest accuracy in terms of generalization performance. In general, it can be seen that CDRELM has the best generalization ability and highest learning speed. BPNN, SVR, L_2–ELM, DRELM, and CELM perform well in comparison with L_1–ELM. On the basis of linear regression datasets, L_2–ELM can perform slightly better than CDRELM in terms of robustness and accuracy. However, when the WGN is added to the datasets, CDRELM has the most stable performance. In Fig. 7 and Table 4, as



(a) Linear regression **(b)** Linear regression with WGN

Fig. 8 Regression results on linear regression and linear regression with WGN

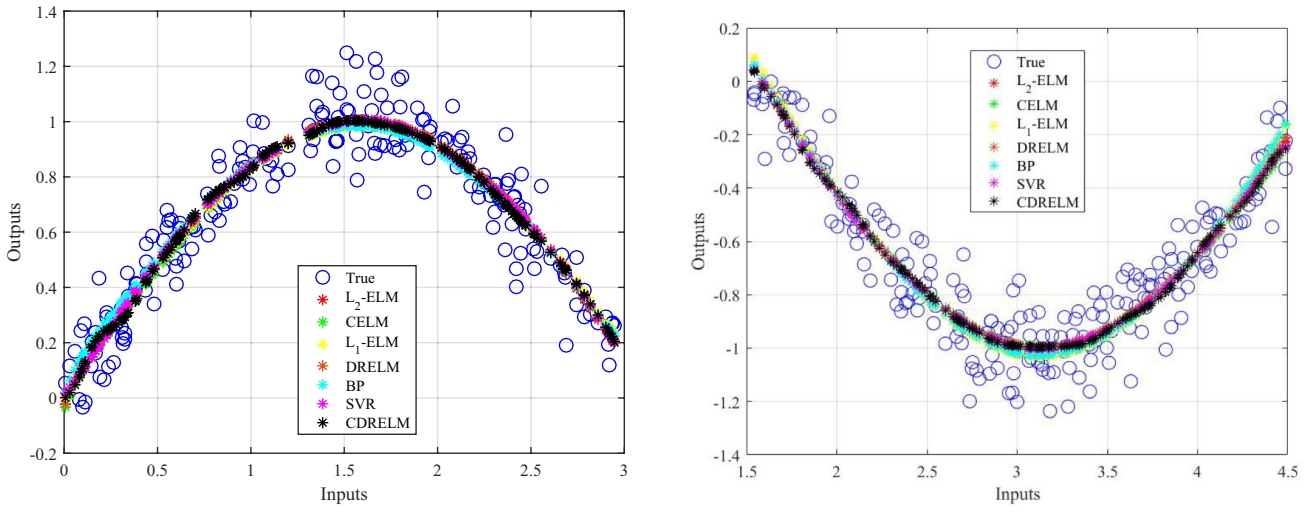


(a) Self–defining function **(b)** Self–defining function with WGN

Fig. 9 Regression results on self-defining function and self-defining function with WGN

the number of outliers increases, the determination coefficient R^2 of seven algorithms decreases slightly. Meanwhile, MSE and time are almost unaffected by outliers. It is clear that CDRELM always maintains a high level of generalization performance and fast efficiency.

According to the papers reported by Jing et al. [18] and Fu et al. [19], as a feature selection method with L_1 norm and L_2 norm, CDRELM has strong anti-interference capability in theory. Meanwhile, the experimental results (Figs. 7, 8, 9, and 10 and Tables 4, 5, 6, and 7) show that CDRELM has a low sensitivity to outliers and a stable performance. It can be seen from Table 4 that the selected parameters can also change as the number of outliers increases. The larger the number of outliers, the lower the regularization parameter η for L_1 norm. The number of outliers is positively associated with the regularization parameter ξ for L_2 norm. The experiments also demonstrate that L_1 norm offers an automatic variable selection through a sparse vector, and L_2 norm strengthens the control ability. With the number of outliers increasing, L_2 norm increases to prevent overfitting and L_1 norm decreases to avoid excessive sparsity in



(a) Moon1 **(b)** Moon2

Fig. 10 Regression results on two-moon dataset

Table 4 Experimental results on sinc function and sinc function with WGN

Dataset	Activation function	Algorithm	MSE	R^2	Time(s)	Parameter $(\eta, \xi, \sigma, C, \theta)$
Sinc function	Sigmoid	L_2–ELM	6.2265 ± 0.0023	0.9248 ± 0.0008	0.0499 ± 0.0001	–, 2^{-31}, –, –, –
		CELM	6.0988 ± 0.0011	0.9728 ± 0.0010	0.0238 ± 0.0003	–, 2^{-30}, 2^{-2}, –, –
		L_1–ELM	6.4575 ± 0.0043	0.9125 ± 0.0005	0.0295 ± 0.0003	2^{-10}, –, –, –, –
		CDRELM	$\mathbf{4.0590 \pm 0.0003}$	$\mathbf{0.9992 \pm 0.0005}$	$\mathbf{0.0224 \pm 0.0001}$	2^{-10}, 2^{-33}, 2^{-2}, –, –
		BPNN	5.3132 ± 0.0015	0.9108 ± 0.0006	1.3255 ± 0.0003	–, –, –, –, –
		DRELM	6.0823 ± 0.0019	0.9730 ± 0.0008	0.0256 ± 0.0006	2^{-10}, 2^{-33}, –, –, –
	Sine	L_2–ELM	6.7276 ± 0.0011	0.9304 ± 0.0009	0.0518 ± 0.0005	–, 2^{-30}, –, –, –
		CELM	6.4560 ± 0.0021	0.9756 ± 0.0023	0.0226 ± 0.0026	–, 2^{-30}, 2^{-2}, –, –
		L_1–ELM	6.4621 ± 0.0046	0.9143 ± 0.0027	0.0268 ± 0.0008	2^{-9}, –, –, –, –
		CDRELM	$\mathbf{4.2525 \pm 0.0009}$	$\mathbf{1.0000 \pm 0.0000}$	$\mathbf{0.0220 \pm 0.0001}$	2^{-8}, 2^{-34}, 2^{-2}, –, –
		BPNN	5.3219 ± 0.0031	0.9032 ± 0.0027	1.3259 ± 0.0004	–, –, –, –, –
		DRELM	6.0825 ± 0.0024	0.9710 ± 0.0014	0.0231 ± 0.0002	2^{-8}, 2^{-30}, –, –, –
		SVR	6.0327 ± 0.0020	0.9101 ± 0.0022	0.0671 ± 0.0004	–, –, –, 2^{6}, 2^{0}
Sinc function with WGN of power 2 dBW	Sigmoid	L_2–ELM	6.2892 ± 0.0031	0.8677 ± 0.0032	0.0422 ± 0.0009	–, 2^{-23}, –, –, –
		CELM	6.0896 ± 0.0029	0.9689 ± 0.0057	0.0400 ± 0.0004	–, 2^{-23}, 2^{-2}, –, –
		L_1–ELM	6.5562 ± 0.0024	0.8676 ± 0.0031	0.0395 ± 0.0005	2^{-14}, –, –, –, –
		CDRELM	$\mathbf{5.8769 \pm 0.0018}$	$\mathbf{0.9912 \pm 0.0026}$	$\mathbf{0.0318 \pm 0.0002}$	2^{-16}, 2^{-24}, 2^{-2}, –, –
		BPNN	7.3872 ± 0.0049	0.8963 ± 0.0046	1.2617 ± 0.0004	–, –, –, –, –
		DRELM	6.3219 ± 0.0026	0.9581 ± 0.0037	0.0345 ± 0.0004	2^{-14}, 2^{-23}, –, –, –
	Sine	L_2–ELM	8.0972 ± 0.0136	0.8245 ± 0.0019	0.0421 ± 0.0004	–, 2^{-23}, –, –, –
		CELM	7.7860 ± 0.0089	0.9581 ± 0.0031	0.0370 ± 0.0002	–, 2^{-25}, 2^{-2}, –, –
		L_1–ELM	8.9091 ± 0.0056	0.8122 ± 0.0028	0.0320 ± 0.0004	2^{-9}, –, –, –, –
		CDRELM	$\mathbf{6.1403 \pm 0.0025}$	$\mathbf{0.9900 \pm 0.0032}$	$\mathbf{0.0308 \pm 0.0001}$	2^{-12}, 2^{-24}, 2^{-2}, –, –
		BPNN	7.2934 ± 0.0035	0.8003 ± 0.0037	1.2669 ± 0.0005	–, –, –, –, –
		DRELM	6.8903 ± 0.0078	0.9579 ± 0.0041	0.0348 ± 0.0003	2^{-13}, 2^{-24}, –, –, –
		SVR	7.8340 ± 0.0031	0.9059 ± 0.0044	0.0735 ± 0.0012	–, –, –, 2^{6}, 2^{3}
Sinc function with WGN of power 5 dBW	Sigmoid	L_2–ELM	6.2738 ± 0.0014	0.8561 ± 0.0029	0.0453 ± 0.0002	–, 2^{-10}, –, –, –
		CELM	6.0744 ± 0.0031	0.9558 ± 0.0049	0.0401 ± 0.0009	–, 2^{-12}, 2^{-2}, –, –
		L_1–ELM	6.9655 ± 0.0053	0.7976 ± 0.0038	0.0396 ± 0.0011	2^{-24}, –, –, –, –
		CDRELM	$\mathbf{5.7569 \pm 0.0018}$	$\mathbf{0.9818 \pm 0.0019}$	$\mathbf{0.0318 \pm 0.0004}$	2^{-24}, 2^{-10}, 2^{-2}, –, –
		BPNN	7.5728 ± 0.0033	0.8932 ± 0.0039	1.2889 ± 0.0007	–, –, –, –, –
		DRELM	5.9882 ± 0.0005	0.9601 ± 0.0043	0.0378 ± 0.0005	2^{-24}, 2^{-11}, –, –, –
	Sine	L_2–ELM	8.0693 ± 0.0057	0.8598 ± 0.0036	0.0421 ± 0.0004	–, 2^{-10}, –, –, –
		CELM	7.7851 ± 0.0013	0.9666 ± 0.0023	0.0299 ± 0.0004	–, 2^{-11}, 2^{-2}, –, –
		L_1–ELM	8.9198 ± 0.0049	0.8604 ± 0.0019	0.0365 ± 0.0006	2^{-23}, –, –, –, –
		CDRELM	$\mathbf{5.0401 \pm 0.0027}$	$\mathbf{0.9856 \pm 0.0010}$	$\mathbf{0.0288 \pm 0.0002}$	2^{-24}, 2^{-10}, 2^{-2}, –, –
		BPNN	7.5892 ± 0.0056	0.8010 ± 0.0034	1.2897 ± 0.0007	–, –, –, –, –
		DRELM	5.9866 ± 0.0035	0.9565 ± 0.0002	0.0375 ± 0.0005	2^{-24}, 2^{-13}, –, –, –
		SVR	7.9360 ± 0.0154	0.8798 ± 0.0038	0.0750 ± 0.0004	–, –, –, 2^{6}, 2^{2}

Table 4 (continued)

Dataset	Activation function	Algorithm	MSE	R^2	Time(s)	Parameter $(\eta, \xi, \sigma, C, \theta)$
Sinc function with WGN of power 10 dBW	Sigmoid	L_2–ELM	7.2980 ± 0.0051	0.8127 ± 0.0037	0.0419 ± 0.0006	–, 2^{-3}, –, –, –
		CELM	7.0489 ± 0.0045	0.9362 ± 0.0026	0.0410 ± 0.0008	–, 2^{-3}, 2^{-2}, –, –
		L_1–ELM	7.8532 ± 0.0057	0.7789 ± 0.0048	0.0392 ± 0.0003	2^{-34}, –, –, –, –
		CDRELM	**6.7739 ± 0.0031**	**0.9408 ± 0.0069**	**0.0303 ± 0.0001**	2^{-30}, 2^{-4}, 2^{-2}, –, –
		BPNN	8.2298 ± 0.0046	0.8820 ± 0.0044	1.3491 ± 0.0003	–, –, –, –, –
		DRELM	7.5742 ± 0.0127	0.9318 ± 0.0031	0.0309 ± 0.0005	2^{-31}, 2^{-3}, –, –, –
	Sine	L_2–ELM	8.0492 ± 0.0035	0.8190 ± 0.0090	0.0421 ± 0.0010	–, 2^{-3}, –, –, –
		CELM	7.8835 ± 0.0049	0.9358 ± 0.0045	0.0299 ± 0.0009	–, 2^{-5}, 2^{-2}, –, –
		L_1–ELM	9.1019 ± 0.0131	0.8798 ± 0.0032	0.0367 ± 0.0008	2^{-29}, –, –, –, –
		CDRELM	**7.2014 ± 0.0030**	**0.9580 ± 0.0017**	**0.0298 ± 0.0003**	2^{-32}, 2^{-4}, 2^{-2}, –, –
		BPNN	8.2388 ± 0.0074	0.8161 ± 0.0056	1.3489 ± 0.0011	–, –, –, –, –
		DRELM	7.5688 ± 0.0198	0.9377 ± 0.0024	0.0309 ± 0.0005	2^{-31}, 2^{-3}, –, –, –
		SVR	8.2706 ± 0.0215	0.8813 ± 0.0016	0.0742 ± 0.0014	–, –, –, 2^{6}, 2^{-1}

CDRELM. According to the situations of all the functions with noises, CDRELM can maintain the accuracy and stability required to solve regression problems without interference.

Performance on Benchmark Datasets

To further verify the performance of seven algorithms, five real-world datasets from the Kaggle datasets and UCI

Table 5 Experimental results on linear regression and linear regression with WGN

Dataset	Activation function	Algorithm	MSE	R^2	Time(s)	Parameter $(\eta, \xi, \sigma, C, \theta)$
Linear regression	Sigmoid	L_2–ELM	1.0184 ± 0.0571	**1.0000 ± 0.0000**	0.0609 ± 0.0001	–, 2^{-36}, –, –, –
		CELM	1.0171 ± 0.0379	0.9983 ± 0.0003	0.0293 ± 0.0003	–, 2^{-35}, 2^{-2}, –, –
		L_1–ELM	1.2473 ± 0.0900	0.9986 ± 0.0003	0.0318 ± 0.0002	2^{-10}, –, –, –, –
		CDRELM	**0.7126 ± 0.0419**	0.9998 ± 0.0003	**0.0279 ± 0.0002**	2^{-10}, 2^{-30}, 2^{-2}, –, –
		BPNN	1.9875 ± 0.0814	0.9935 ± 0.0002	1.2352 ± 0.0003	–, –, –, –, –
		DRELM	0.7349 ± 0.0998	0.9995 ± 0.0003	0.0301 ± 0.0004	2^{-10}, 2^{-30}, –, –, –
	Sine	L_2–ELM	1.2570 ± 0.0304	**1.0000 ± 0.0000**	0.0483 ± 0.0012	–, 2^{-35}, –, –, –
		CELM	1.1566 ± 0.0798	0.9998 ± 0.0000	0.0212 ± 0.0009	–, 2^{-35}, 2^{-2}, –, –
		L_1–ELM	1.4813 ± 0.1023	0.9981 ± 0.0002	0.0219 ± 0.0006	2^{-9}, –, –, –, –
		CDRELM	**0.6988 ± 0.0099**	**1.0000 ± 0.0000**	**0.0175 ± 0.0008**	2^{-8}, 2^{-30}, 2^{-2}, –, –
		BPNN	1.9734 ± 0.0516	0.9992 ± 0.0000	1.2366 ± 0.0012	–, –, –, –, –
		DRELM	0.7129 ± 0.0020	0.9995 ± 0.0000	0.0203 ± 0.0011	2^{-10}, 2^{-30}, –, –, –
		SVR	1.0281 ± 0.0452	0.9997 ± 0.0000	0.0504 ± 0.0005	–, –, –, 2^{8}, 2^{0}
Linear regression with WGN	Sigmoid	L_2–ELM	1.7961 ± 0.0321	0.9607 ± 0.0030	0.0419 ± 0.0009	–, 2^{-10}, –, –, –
		CELM	1.1899 ± 0.0076	0.9708 ± 0.0015	0.0267 ± 0.0007	–, 2^{-10}, 2^{-2}, –, –
		L_1–ELM	1.3040 ± 0.0085	0.9681 ± 0.0028	0.0308 ± 0.0007	2^{-10}, –, –, –, –
		CDRELM	**1.1897 ± 0.0057**	**0.9712 ± 0.0034**	**0.0229 ± 0.0006**	2^{-6}, 2^{-8}, 2^{-2}, –, –
		BPNN	1.8835 ± 0.0045	0.9438 ± 0.0025	1.2333 ± 0.0012	–, –, –, –, –
		DRELM	1.5381 ± 0.0043	0.9522 ± 0.0027	0.0233 ± 0.0007	2^{-10}, 2^{-7}, –, –, –
	Sine	L_2–ELM	1.7277 ± 0.0055	0.9483 ± 0.0020	0.0417 ± 0.0006	–, 2^{-10}, –, –, –
		CELM	1.5363 ± 0.0039	0.9460 ± 0.0034	0.0249 ± 0.0005	–, 2^{-10}, 2^{-2}, –, –
		L_1–ELM	1.5382 ± 0.0068	0.9604 ± 0.0059	0.0276 ± 0.0007	2^{-9}, –, –, –, –
		CDRELM	**1.4973 ± 0.0093**	**0.9606 ± 0.0013**	**0.0231 ± 0.0003**	2^{-8}, 2^{-8}, 2^{-2}, –, –
		BPNN	1.8643 ± 0.0022	0.9441 ± 0.0013	1.2572 ± 0.0005	–, –, –, –, –
		DRELM	1.5381 ± 0.0110	0.9522 ± 0.0032	0.0233 ± 0.0004	2^{-7}, 2^{-7}, –, –, –
		SVR	1.5183 ± 0.0107	0.9479 ± 0.0040	0.0897 ± 0.0007	–, –, –, 2^{8}, 2^{3}

Table 6 Experimental results on self-defining function and self–defining function with WGN

Dataset	Activation function	Algorithm	MSE	R^2	Time(s)	Parameter $(\eta, \xi, \sigma, C, \theta)$
Self-defining function	Sigmoid	L_2–ELM	1.9383 ± 0.0301	0.9068 ± 0.0007	0.0559 ± 0.0005	–, 2^{-18}, –, –, –
		CELM	3.9911 ± 0.0098	0.9007 ± 0.0021	0.0237 ± 0.0001	–, 2^{-20}, 2^{-2}, –, –
		L_1–ELM	2.2779 ± 0.0104	0.8949 ± 0.0002	0.0263 ± 0.0006	2^{-10}, –, –, –, –
		CDRELM	**1.1320 ± 0.0107**	**0.9966 ± 0.0008**	**0.0212 ± 0.0008**	2^{-11}, 2^{-16}, 2^{-2}, –, –
		BPNN	3.1340 ± 0.0432	0.8564 ± 0.0032	1.4991 ± 0.0077	–, –, –, –, –
		DRELM	2.0635 ± 0.0236	0.9304 ± 0.0008	0.0235 ± 0.0007	2^{-11}, 2^{-17}, –, –, –
	Sine	L_2–ELM	2.1640 ± 0.0533	0.9265 ± 0.0003	0.0521 ± 0.0007	–, 2^{-20}, –, –, –
		CELM	1.5682 ± 0.0309	0.9082 ± 0.0004	0.0226 ± 0.0004	–, 2^{-15}, 2^{-2}, –, –
		L_1–ELM	1.7750 ± 0.0313	0.8751 ± 0.0003	0.0268 ± 0.0004	2^{-9}, –, –, –, –
		CDRELM	**1.4828 ± 0.0114**	**0.9984 ± 0.0007**	**0.0166 ± 0.0002**	2^{-12}, 2^{-16}, 2^{-2}, –, –
		BPNN	3.1290 ± 0.0128	0.8563 ± 0.0010	1.4998 ± 0.0008	–, –, –, –, –
		DRELM	2.0729 ± 0.0336	0.9152 ± 0.0007	0.0236 ± 0.0007	2^{-10}, 2^{-17}, –, –, –
		SVR	6.0281 ± 0.0256	0.8021 ± 0.0188	0.1060 ± 0.0018	–, –, –, 2^{7}, 2^{0}
Self-defining function with WGN	Sigmoid	L_2–ELM	3.0632 ± 0.0369	0.8092 ± 0.0006	0.0640 ± 0.0013	–, 2^{-12}, –, –, –
		CELM	3.3166 ± 0.1107	0.8219 ± 0.0005	0.0232 ± 0.0008	–, 2^{-13}, 2^{-2}, –, –
		L_1–ELM	7.8308 ± 0.1098	0.7858 ± 0.0009	0.0270 ± 0.0006	2^{-18}, –, –, –, –
		CDRELM	**1.4385 ± 0.0111**	**0.9315 ± 0.0007**	**0.0207 ± 0.0004**	2^{-20},2^{-14},2^{-2}, –, –
		BPNN	3.9584 ± 0.0389	0.7952 ± 0.0007	1.6511 ± 0.0004	–, –, –, –, –
		DRELM	3.1968 ± 0.0422	0.8592 ± 0.0008	0.0242 ± 0.0007	2^{-17}, 2^{-14}, –, –, –
	Sine	L_2–ELM	3.4683 ± 0.1429	0.7063 ± 0.0008	0.0533 ± 0.0007	–, 2^{-12}, –, –, –
		CELM	3.5432 ± 0.0966	0.7432 ± 0.0017	0.0255 ± 0.0006	–, 2^{-13}, 2^{-2}, –, –
		L_1–ELM	8.4988 ± 0.0865	0.7901 ± 0.0022	0.0302 ± 0.0003	2^{-19}, –, –, –, –
		CDRELM	**1.6300 ± 0.0438**	**0.9314 ± 0.0013**	**0.0158 ± 0.0003**	2^{-18}, 2^{-14}, 2^{-2}, –, –
		BPNN	3.9459 ± 0.0098	0.8494 ± 0.0026	1.6513 ± 0.0008	–, –, –, –, –
		DRELM	3.1968 ± 0.0428	0.8592 ± 0.0032	0.0242 ± 0.0006	2^{-18}, 2^{-14}, –, –, –
		SVR	6.0943 ± 0.0031	0.6981 ± 0.0026	0.1127 ± 0.0046	–, –, –, 2^{7}, 2^{3}

machine learning repository were tested. The datasets were of different types, including low, medium, and high dimensions and small, medium, and large sizes.

We randomly divide the benchmark datasets into two subsets (80% + 20%), the former for training and the latter for testing. Table 8 shows the data descriptions of the five datasets, which can be classified into three groups of data:

1. Datasets with relatively small size and high dimensions: the octane number dataset which contains 60 gasoline samples. The samples used Fourier transform near-infrared spectroscopy to scan. Each sample has 401 features, and each feature is a wavelength point from the scanning range of 900 to 1700 nm.
2. Datasets with medium size and medium dimensions: The Boston housing dataset has 506 samples with 13 features concerning housing prices in the suburbs of Boston. The life expectancy dataset includes 958 samples with 18 features for analyzing the factors that affect the average life expectancy in different countries.
3. Datasets with large size and small dimensions: The energy consumption dataset covers 2208 instances with five features that show the hourly energy consumption from October 1 to December 31 during the years 2008 to 2012. The air quality dataset contains 9358 samples of hourly averaged responses with eight reference analyzers.

As can be seen from Figs. 11, 12, 13, 14, and 15, six algorithms were compared with CDRELM on different types of datasets. Figure 11 shows the predictive ability of seven algorithms on the octane number dataset. The performances of seven algorithms on the Boston housing, life expectancy, energy consumption, and air quality datasets are shown in Figs. 12, 13, 14, and 15, respectively. It is clear

Table 7 Experimental results on two-moon dataset

Dataset	Activation function	Algorithm	MSE	R^2	Time(s)	Parameter $(\eta, \xi, \sigma, C, \theta)$
Moon1	Sigmoid	L_2–ELM	0.0109 ± 0.0003	0.8972 ± 0.0002	0.0535 ± 0.0008	–, 2^{-13}, –, –, –
		CELM	0.0108 ± 0.0002	0.8970 ± 0.0003	0.0210 ± 0.0008	–, 2^{-14}, 2^{-2}, –, –
		L_1–ELM	0.0188 ± 0.0003	0.8853 ± 0.0004	0.0243 ± 0.0008	2^{-10}, –, –, –, –
		CDRELM	**0.0095 ± 0.0003**	**0.9068 ± 0.0002**	**0.0182 ± 0.0002**	2^{-10}, 2^{-12}, 2^{-2}, –, –
		BPNN	0.0112 ± 0.0003	0.8960 ± 0.0004	1.2620 ± 0.0004	–, –, –, –, –
		DRELM	0.0117 ± 0.0003	0.8992 ± 0.0003	0.0239 ± 0.0002	2^{-10}, 2^{-14}, –, –, –
	Sine	L_2–ELM	**0.0100 ± 0.0000**	0.8937 ± 0.0003	0.0520 ± 0.0002	–, 2^{-12}, –, –, –
		CELM	0.0109 ± 0.0005	0.8887 ± 0.0002	0.0235 ± 0.0004	–, 2^{-9}, 2^{-2}, –, –
		L_1–ELM	0.0153 ± 0.0005	0.8935 ± 0.0001	0.0230 ± 0.0008	2^{-9}, –, –, –, –
		CDRELM	0.0104 ± 0.0004	**0.8997 ± 0.0001**	**0.0104 ± 0.0001**	2^{-8}, 2^{-12}, 2^{-2}, –, –
		BPNN	0.0110 ± 0.0008	0.8958 ± 0.0003	1.2589 ± 0.0004	–, –, –, –, –
		DRELM	0.0117 ± 0.0003	0.8992 ± 0.0003	0.0239 ± 0.0004	2^{-8}, 2^{-13}, –, –, –
		SVR	0.0281 ± 0.0005	0.8981 ± 0.0003	0.0721 ± 0.0003	–, –, –, 2^{8}, 2^{2}
Moon2	Sigmoid	L_2–ELM	0.0116 ± 0.0003	0.8863 ± 0.0003	0.0498 ± 0.0002	–, 2^{-10}, –, –, –
		CELM	0.0122 ± 0.0008	0.8921 ± 0.0001	0.0244 ± 0.0001	–, 2^{-14}, 2^{-2}, –, –
		L_1–ELM	0.0116 ± 0.0002	0.8864 ± 0.0003	0.0240 ± 0.0002	2^{-10}, –, –, –, –
		CDRELM	**0.0115 ± 0.0003**	**0.9066 ± 0.0001**	**0.0169 ± 0.0001**	2^{-10}, 2^{-10}, 2^{-2}, –, –
		BPNN	0.0125 ± 0.0001	0.8897 ± 0.0002	1.5250 ± 0.0001	–, –, –, –, –
		DRELM	0.0120 ± 0.0001	0.8990 ± 0.0002	0.0239 ± 0.0004	2^{-10}, 2^{-11}, –,–, –
	Sine	L_2–ELM	0.0117 ± 0.0001	0.8990 ± 0.0003	0.0546 ± 0.0003	–, 2^{-10}, –, –, –
		CELM	0.0116 ± 0.0003	0.8988 ± 0.0003	0.0240 ± 0.0002	–, 2^{-9}, 2^{-2}, –, –
		L_1–ELM	0.0135 ± 0.0001	0.8994 ± 0.0008	0.0239 ± 0.0002	2^{-9}, –, –, –, –
		CDRELM	**0.0116 ± 0.0000**	**0.8998 ± 0.0004**	**0.0187 ± 0.0001**	2^{-10}, 2^{-9}, 2^{-2}, –
		BPNN	0.0126 ± 0.0002	0.8881 ± 0.0003	1.5261 ± 0.0003	–, –, –, –, –
		DRELM	0.0119 ± 0.0003	0.8992 ± 0.0005	0.0253 ± 0.0001	2^{-10}, 2^{-10}, –,–, –
		SVR	0.0214 ± 0.0001	0.8702 ± 0.0003	0.0794 ± 0.0001	–, –, –, 2^{8}, 2^{2}

that CDRELM has the greatest performance and can fit the true value best. Table 9 provides the detailed comparisons of seven algorithms on three small- or medium-sized datasets. Table 10 lists the performance results of seven algorithms on two large-sized datasets. It can be seen that the proposed algorithm achieves better generalization performance in three types of datasets at much higher learning speeds. Figures 16, 17, and 18 show the comparison of R^2, MSE, and time among seven algorithms on each dataset and show the performance of seven algorithms with two different activate functions on various sizes of datasets. In Fig. 16, the histogram represents the performance index R^2, including two situations with different activate functions. The comparison of MSE among seven algorithms on the datasets divided into large-, small-, or medium-sized datasets is shown in Fig. 17. Interestingly, CDRELM can solve the problems of small- or medium-sized datasets much better than those of large-sized datasets. L_2–ELM, L_1–ELM, CELM, DRELM, and CDRELM obtain a slightly larger value of MSE than SVR and BPNN on large datasets. Hence, CDRELM is better at processing small datasets than other types. Figure 18 compares the overall trends in terms of the time of all the datasets. It can be seen that CDRELM is significantly faster than BPNN and SVR. Figure 18a is not clear to see the advantage of CDRELM. Thus, we completed the Fig. 18b to compare the time intuitionally. CDRELM is more efficient than DRELM, CELM, L_2–ELM, and L_1–ELM. It is clear that the proposed CDRELM obtains the fastest and the most stable performance with the highest accuracy.

Table 8 Details of benchmark datasets

Dataset	Number of training samples	Number of testing samples	Number of features
Octane number	48	12	401
Boston housing	405	101	13
Life expectancy	766	192	18
Energy consumption	1766	442	5
Air quality	7486	1872	8

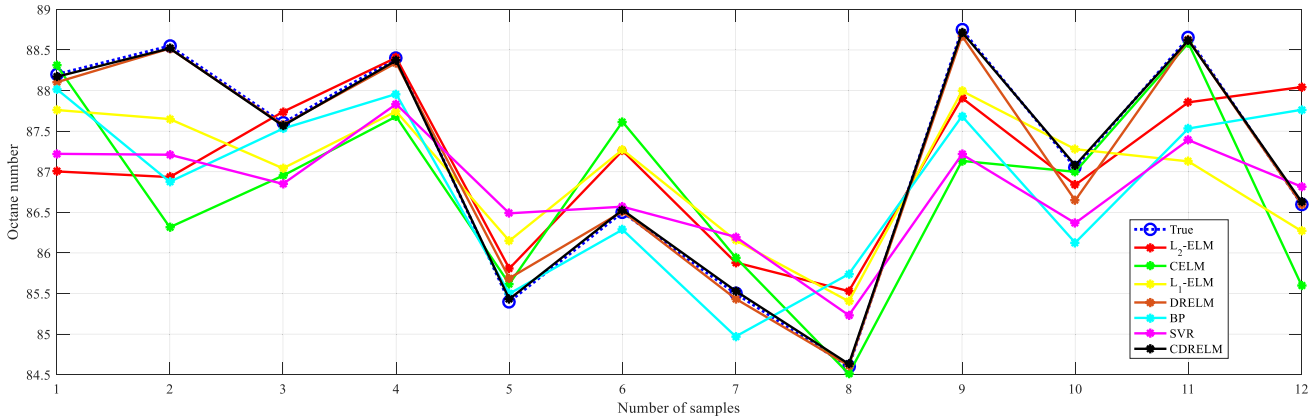


Fig. 11 Regression results on octane number dataset

To analyze the statistical accuracy more clearly, Table 11 lists the average ranks which are computed by the average value of R^2 in Tables 9 and 10 for each algorithm with two activate functions. As seen in Table 11, CDRELM is ranked first, and DRELM, CELM, BPNN, SVR, and L_1–ELM are ranked in turn. The experimental results also verify the expected achievement of each algorithm. L_1–ELM aims at reducing the learning time. DRELM has 1-norm and 2-norm penalties with high learning speeds and the ability to prevent overfitting. CELM aims at improving accuracy and robustness. Having the advantages of both CELM and DRELM, CDRELM can attain better generalization ability at a faster learning speed by introducing the C-loss function, L_2 norm and L_1 norm into ELM.

To obtain various precision and credible results, the Friedman statistical method was used to determine whether all the algorithms have the same performance. Let N be the number of datasets and m denotes the counts of algorithms. Meanwhile, R_i represents the average ranks in Table 11. Friedman statistic [34] follows the distribution of χ_F^2 with $m-1$ degrees of freedom, which is defined as follows:

$$\chi_F^2 = \frac{12N}{m(m+1)}\left[\sum_i R_i^2 - \frac{m(m+1)^2}{4}\right] \tag{31}$$

Based on Eq. (31), Iman et al. [35] proposed a better statistic:

$$F_F = \frac{(N-1)\chi_F^2}{N(m-1)-\chi_F^2} \tag{32}$$

which follows the F-distribution with $m-1$ and $(m-1)(N-1)$ degrees of freedom. According to Tables 9 and 10, $\chi_F^2 = 44.52$, $F_F \approx 25.88$, and $F_{0.05}(6,54)$ are 2.272 by referring to the F-distribution critical value table. It is clear that $F_F = 25.88 > F_{0.05}(6,54) = 2.272$, so the null

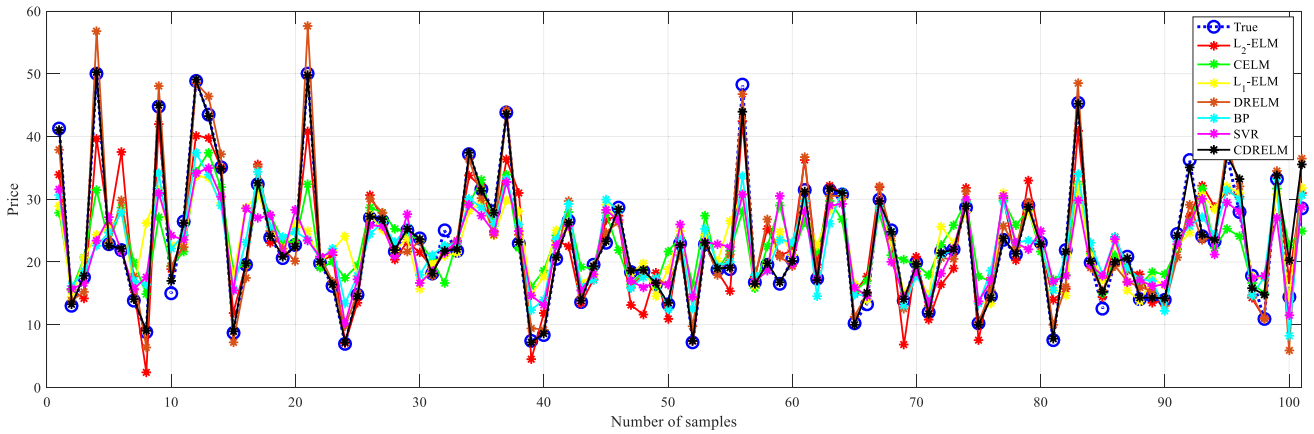


Fig. 12 Regression results on Boston housing dataset

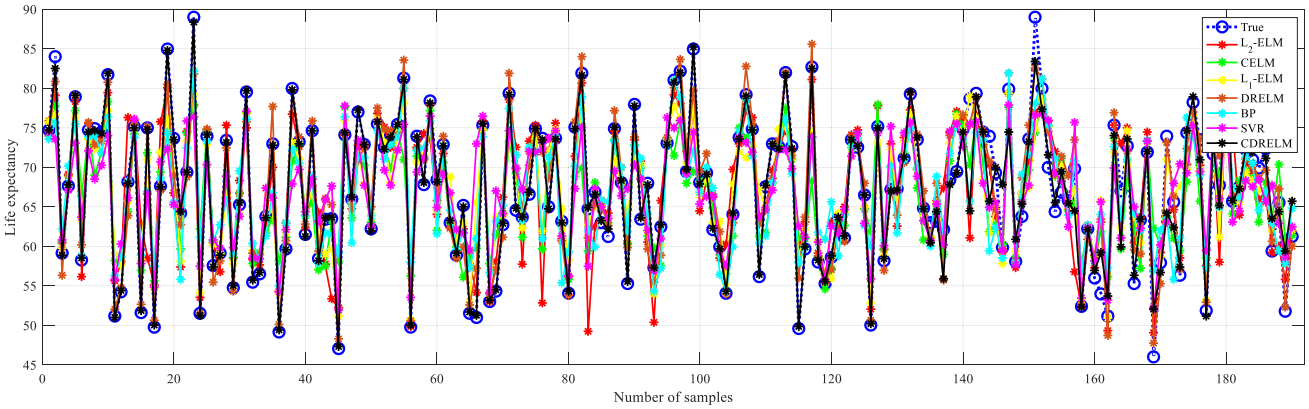


Fig. 13 Regression results on life expectancy dataset

hypothesis is rejected. Hence, it is shown that the performance of the algorithms is significantly different.

To further differentiate the algorithms, the Nemenyi test [36] is used to pairwise compare seven algorithms. It is defined as follows:

$$CD = q_\alpha \sqrt{\frac{m(m+1)}{6N}} \tag{33}$$

where q_α is the critical value of Tukey distribution. When $\alpha = 0.05$ and $m = 7$, $q_\alpha = 2.949$ according to the inspection table of Nemenyi. The null hypothesis that two algorithms have the same performance is rejected if the corresponding average ranks differ by at least the critical difference $CD \approx 2.8439$. Due to the average rank difference between CDRELM and L_1–ELM which is $5.9 - 1 = 4.9$ and is much larger than the critical difference 2.8439, the performance of CDRELM is substantially better than that of L_1–ELM. Similarly, the performance of CDRELM is much superior to that of SVR ($5.6 - 1 = 4.1 > 2.8439$). As a result of $5.1 - 1 = 4.1 > 2.8439$, the Nemenyi test can detect significant difference between CDRELM and BPNN. The result $4.6 - 1 = 3.6 > 2.8439$ makes it clear that CDRELM performs much better than CELM. It is clear that $3.8 - 1 = 2.8 < 2.8439$ and $2 - 1 = 1 < 2.8439$. Thus, CDRELM has slightly better performance than DRELM and L_2–ELM. The above comparison can be visually shown using the Friedman test chart. In Fig. 19, the vertical axis represents each algorithm. For the horizontal axis, dot is the value of average rank, and the horizontal line segment centered on a dot represents the value of CD. If there are overlaps between the horizontal line segment of two algorithms, then, there is no remarkable difference between the two algorithms. Hence, it can be clearly seen that CDRELM has significantly better performance than CELM, L_1–ELM, BPNN, and SVR. CDRELM is slightly better than DRELM and L_2–ELM. Of the seven algorithms, CDRELM has the best performance and L_1–ELM has the worst accuracy.

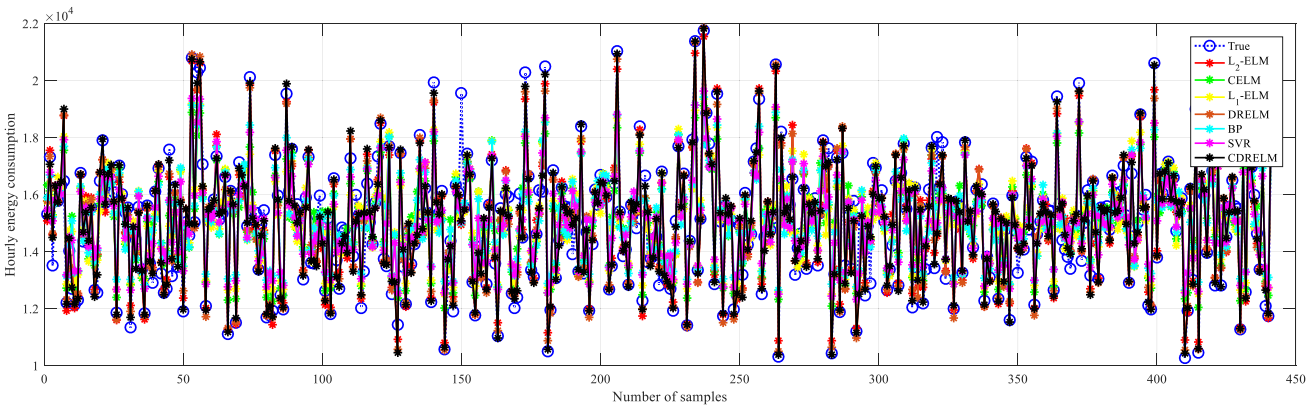


Fig. 14 Regression results on hourly energy consumption dataset

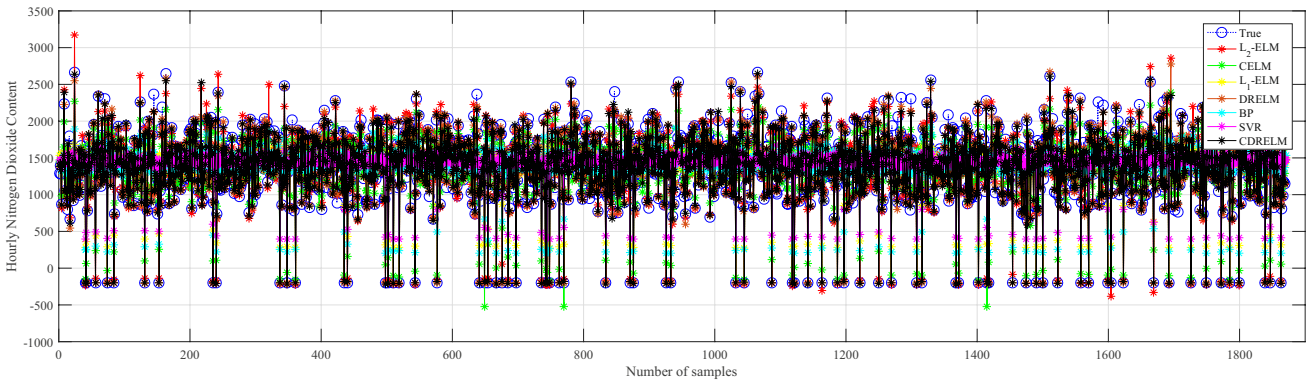


Fig. 15 Regression results on air quality dataset

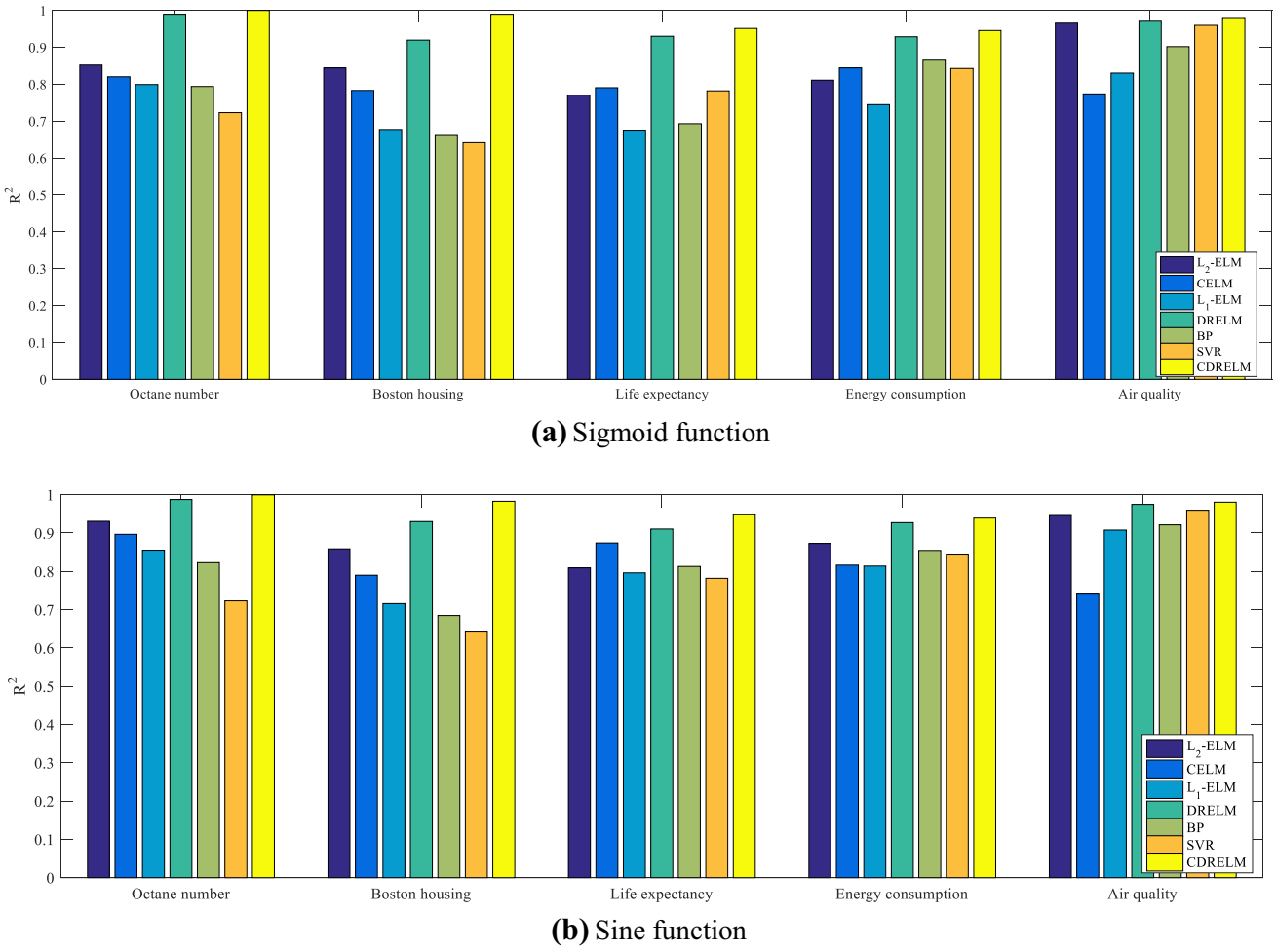
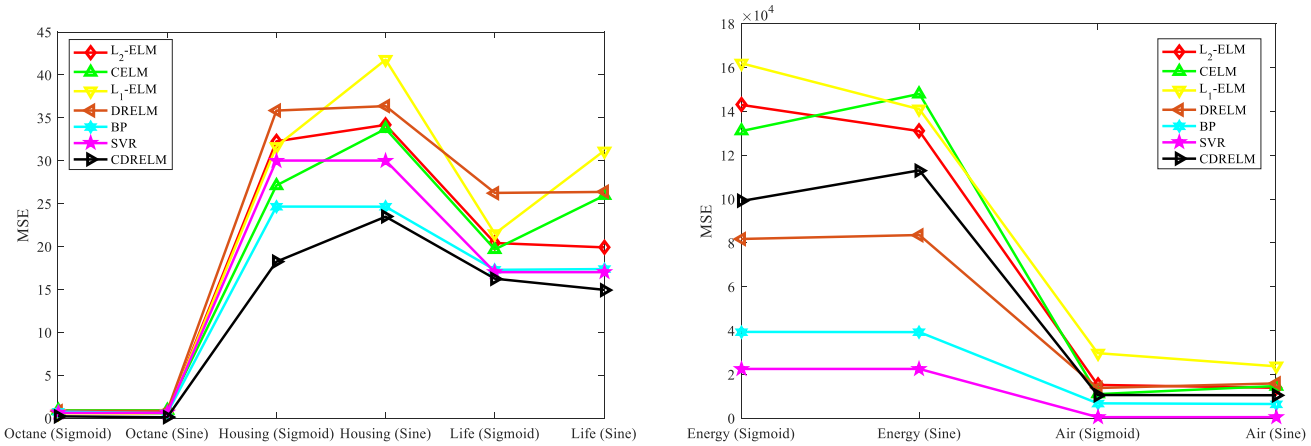
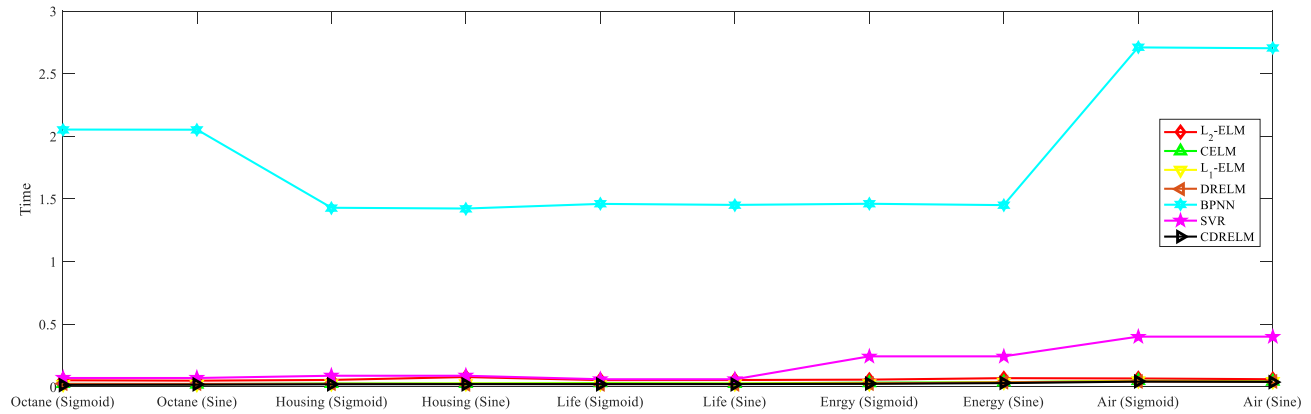


Fig. 16 Comparison of R^2 on all the datasets using different activation function

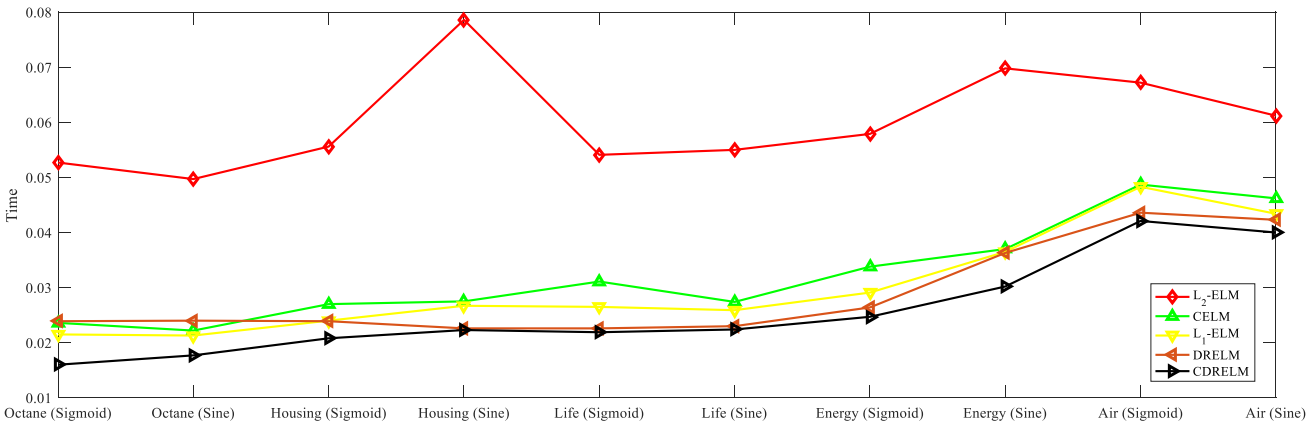


(a) Small and medium sized datasets **(b)** Large sized datasets

Fig. 17 Comparison of MSE on all of datasets



(a) L_2–ELM, L_1–ELM, CELM, DRELM, BPNN, SVR and CDRELM



(b) L_2–ELM, L_1–ELM, CELM, DRELM and CDRELM

Fig. 18 Comparison of time on all of datasets

Table 9 Experimental results on small samples of benchmark dataset

Dataset	Activation function	Algorithm	MSE	R^2	Time(s)	Parameter $(\eta, \xi, \sigma, C, \theta)$
Octane number	Sigmoid	L_2–ELM	0.7735 ± 0.0012	0.8519 ± 0.0006	0.0527 ± 0.0003	–, 2^{-34}, –, –, –
		CELM	0.9208 ± 0.0012	0.8201 ± 0.0004	0.0236 ± 0.0002	–, 2^{-34}, 2^{-2}, –, –
		L_1–ELM	0.5857 ± 0.0027	0.7991 ± 0.0005	0.0215 ± 0.0003	2^{-17}, –, –, –, –
		CDRELM	**0.2226 ± 0.0004**	**0.9998 ± 0.0000**	**0.0160 ± 0.0001**	2^{18}, 2^{-34}, 2^{-2}, –, –
		BPNN	0.7716 ± 0.0044	0.7938 ± 0.0007	2.0541 ± 0.0003	–, –, –, –, –
		DRELM	0.8476 ± 0.0009	0.9894 ± 0.0002	0.0239 ± 0.0003	2^{-18}, 2^{-30}, –, –, –
	Sine	L_2–ELM	0.6140 ± 0.0022	0.9306 ± 0.0004	0.0497 ± 0.0002	–, 2^{-34}, –, –, –
		CELM	0.9239 ± 0.0013	0.8966 ± 0.0003	0.0222 ± 0.0001	–, 2^{-34}, 2^{-2}, –, –
		L_1–ELM	0.5352 ± 0.0035	0.8553 ± 0.0007	0.0213 ± 0.0001	2^{-17}, –, –, –, –
		CDRELM	**0.1130 ± 0.0005**	**0.9997 ± 0.0000**	**0.0177 ± 0.0001**	2^{-18}, 2^{-34}, 2^{-2}, –, –
		BPNN	0.5823 ± 0.0033	0.8229 ± 0.0004	2.0529 ± 0.0012	–, –, –, –, –
		DRELM	0.8476 ± 0.0024	0.9875 ± 0.0001	0.0240 ± 0.0002	2^{-17}, 2^{-31}, –, –, –
		SVR	0.6214 ± 0.0033	0.7229 ± 0.0000	0.0710 ± 0.0003	–, –, –, 2^{7}, 2^{2}
Boston housing	Sigmoid	L_2–ELM	32.2671 ± 0.0305	0.8441 ± 0.0005	0.0556 ± 0.0002	–, 2^{-16}, –, –, –
		CELM	27.1208 ± 0.0543	0.7831 ± 0.0008	0.0270 ± 0.0002	–, 2^{-14}, 2^{-2}, –, –
		L_1–ELM	31.6840 ± 0.0778	0.6773 ± 0.0005	0.0240 ± 0.0001	2^{-10}, –, –, –, –
		CDRELM	**18.2421 ± 0.0132**	**0.9894 ± 0.0004**	**0.0208 ± 0.0001**	2^{-10}, 2^{-20}, 2^{-2}, –, –
		BPNN	24.6598 ± 0.0099	0.6609 ± 0.0006	1.4298 ± 0.0008	–, –, –, –, –
		DRELM	35.8476 ± 0.0214	0.9194 ± 0.0003	0.0239 ± 0.0005	2^{-12}, 2^{-15}, –, –, –
	Sine	L_2–ELM	34.1789 ± 0.0313	0.8588 ± 0.0005	0.0786 ± 0.0005	–, 2^{-16}, –, –, –
		CELM	33.7223 ± 0.0407	0.7901 ± 0.0004	0.0275 ± 0.0003	–, 2^{-9}, 2^{-2}, –, –
		L_1–ELM	41.7759 ± 0.0521	0.7158 ± 0.0005	0.0267 ± 0.0002	2^{-9}, –, –, –, –
		CDRELM	**23.4839 ± 0.0045**	**0.9829 ± 0.0001**	**0.0223 ± 0.0000**	2^{-10}, 2^{-20}, 2^{-2}, –, –
		BPNN	24.6477 ± 0.0022	0.6847 ± 0.0007	1.4237 ± 0.0021	–, –, –, –, –
		DRELM	36.3641 ± 0.0094	0.9299 ± 0.0005	0.0226 ± 0.0002	2^{-12}, 2^{-15}, –, –, –
		SVR	30.0214 ± 0.0005	0.6415 ± 0.0005	0.0890 ± 0.0005	–, –, –, 2^{8}, 2^{0}
Life expectancy	Sigmoid	L_2–ELM	20.4141 ± 0.0321	0.7704 ± 0.0003	0.0541 ± 0.0003	–, 2^{2}, –, –, –
		CELM	19.6719 ± 0.0425	0.7905 ± 0.0006	0.0311 ± 0.0002	–, 2^{2}, 2^{-2}, –, –
		L_1–ELM	21.5021 ± 0.0578	0.6754 ± 0.0005	0.0265 ± 0.0001	2^{-9}, –, –, –, –
		CDRELM	**16.2593 ± 0.0155**	**0.9510 ± 0.0002**	**0.0219 ± 0.0000**	2^{-16}, 2^{2}, 2^{-2}, –, –
		BPNN	17.2907 ± 0.0094	0.6930 ± 0.0004	1.4608 ± 0.0011	–, –, –, –, –
		DRELM	26.2532 ± 0.0077	0.9299 ± 0.0002	0.0226 ± 0.0002	2^{-10}, 2^{2}, –, –, –
	Sine	L_2–ELM	19.9088 ± 0.0235	0.8093 ± 0.0006	0.0550 ± 0.0001	–, 2^{2}, –, –, –
		CELM	25.9338 ± 0.0453	0.8742 ± 0.0006	0.0274 ± 0.0002	–, 2^{3}, 2^{-2}, –, –
		L_1–ELM	31.1602 ± 0.0316	0.7963 ± 0.0004	0.0259 ± 0.0001	2^{-9}, –, –, –, –
		CDRELM	**14.9626 ± 0.0046**	**0.9476 ± 0.0000**	**0.0224 ± 0.0000**	2^{-16}, 2^{2}, 2^{-2}, –, –
		BPNN	17.3802 ± 0.0065	0.8130 ± 0.0004	1.4527 ± 0.0033	–, –, –, –, –
		DRELM	26.3675 ± 0.0101	0.9104 ± 0.0004	0.0230 ± 0.0001	2^{-12}, 2^{2}, –, –, –
		SVR	17.0214 ± 0.0086	0.7820 ± 0.0006	0.0610 ± 0.0003	–, –, –, 2^{8}, 2^{1}

Table 10 Experimental results on large samples of benchmark dataset

Dataset	Activation function	Algorithm	MSE	R^2	Time(s)	Parameter $(\eta, \xi, \sigma, C, \theta)$
Energy consumption	Sigmoid	L_2–ELM	14.3000 ± 0.0135	0.8109 ± 0.0003	0.0579 ± 0.0006	–, 2^{-16}, –, –, –
		CELM	13.1000 ± 0.0096	0.8442 ± 0.0003	0.0338 ± 0.0046	–, 2^{-14}, 2^{-2}, –, –
		L_1–ELM	16.2000 ± 0.0107	0.7447 ± 0.0002	0.0291 ± 0.0004	2^{0}, –, –, –, –
		CDRELM	9.9100 ± 0.0036	**0.9454 ± 0.0002**	**0.0247 ± 0.0001**	2^{1}, 2^{-15}, 2^{-2}, –, –
		BPNN	3.9478 ± 0.0044	0.8650 ± 0.0003	1.4617 ± 0.0023	–, –, –, –, –
		DRELM	8.1835 ± 0.0057	0.9285 ± 0.0006	0.0264 ± 0.0002	2^{0}, 2^{-15}, –, –, –
	Sine	L_2–ELM	13.1000 ± 0.0159	0.8729 ± 0.0004	0.0698 ± 0.0006	–, 2^{-16}, –, –, –
		CELM	14.8100 ± 0.0175	0.8165 ± 0.0016	0.0370 ± 0.0002	–, 2^{-10}, 2^{-2}, –, –
		L_1–ELM	14.1221 ± 0.0099	0.8140 ± 0.0005	0.0365 ± 0.0001	2^{2}, –, –, –, –
		CDRELM	11.3000 ± 0.0032	**0.9389 ± 0.0004**	**0.0302 ± 0.0001**	2^{1}, 2^{-18}, 2^{-2}, –, –
		BPNN	3.9351 ± 0.0012	0.8547 ± 0.0003	1.4510 ± 0.0003	–, –, –, –, –
		DRELM	8.3629 ± 0.0033	0.9271 ± 0.0003	0.0363 ± 0.0003	2^{1}, 2^{-15}, –, –, –
		SVR	**2.2591 ± 0.0031**	0.8426 ± 0.0004	0.2438 ± 0.0004	–, –, –, 2^{6}, 2^{3}
Air quality	Sigmoid	L_2–ELM	1.5230 ± 0.0031	0.9654 ± 0.0006	0.0672 ± 0.0001	–, 2^{-16}, –, –, –
		CELM	1.1018 ± 0.0026	0.7737 ± 0.0004	0.0487 ± 0.0002	–, 2^{-17}, 2^{-2}, –, –
		L_1–ELM	2.9731 ± 0.0033	0.8301 ± 0.0005	0.0483 ± 0.0003	2^{-3}, –, –, –, –
		CDRELM	1.0600 ± 0.0022	**0.9805 ± 0.0007**	**0.0421 ± 0.0001**	2^{-8}, 2^{-15}, 2^{-2}, –, –
		BPNN	0.6879 ± 0.0009	0.9018 ± 0.0004	2.7098 ± 0.0006	–, –, –, –, –
		DRELM	1.3885 ± 0.0015	0.9705 ± 0.0003	0.0436 ± 0.0004	2^{-8}, 2^{-13}, –, –, –
	Sine	L_2–ELM	1.3951 ± 0.0037	0.9457 ± 0.0006	0.06120 ± .0003	–, 2^{-17}, –, –, –
		CELM	1.4790 ± 0.0016	0.7405 ± 0.0006	0.0462 ± 0.0003	–, 2^{-17}, 2^{-2}, –, –
		L_1–ELM	2.3717 ± 0.0022	0.9077 ± 0.0004	0.0434 ± 0.0002	2^{-5}, –, –, –, –
		CDRELM	1.0561 ± 0.0018	**0.9805 ± 0.0002**	**0.0400 ± 0.0004**	2^{-9},2^{-15},2^{-2}, –, –
		BPNN	0.6513 ± 0.0010	0.9213 ± 0.0002	2.7031 ± 0.0003	–, –, –, –, –
		DRELM	1.5933 ± 0.0007	0.9747 ± 0.0002	0.0423 ± 0.0003	2^{-8}, 2^{-13}, –, –, –
		SVR	**0.0512 ± 0.0006**	0.9594 ± 0.0002	0.4011 ± 0.0003	–, –, –, 2^{4}, 2^{2}

Table 11 Accuracy average ranks

Dataset	Activation function	L_2-ELM	CELM	L_1-ELM	BPNN	SVR	DRELM	CDRELM
Octane number	Sigmoid	3	4	5	6	7	2	1
	Sine	3	4	5	6	7	2	1
Boston housing	Sigmoid	3	4	5	6	7	2	1
	Sine	3	4	5	6	7	2	1
Life expectancy	Sigmoid	5	3	6	4	7	2	1
	Sine	5	3	7	6	4	2	1
Energy consumption	Sigmoid	6	4	7	3	5	2	1
	Sine	3	6	7	4	5	2	1
Air quality	Sigmoid	3	7	6	5	4	2	1
	Sine	4	7	6	5	3	2	1
Average rank		3.8	4.6	5.9	5.1	5.6	2	1

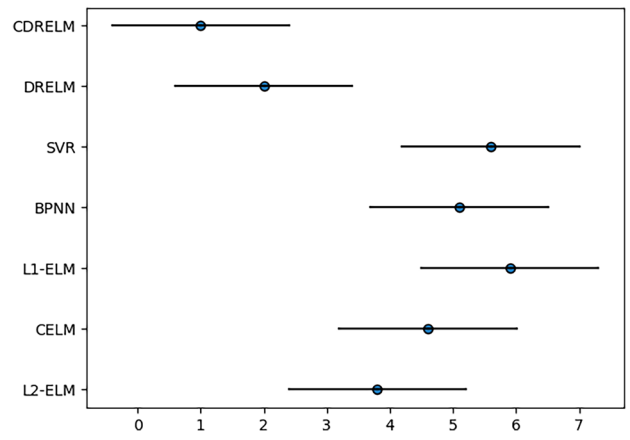


Fig. 19 Friedman test

Conclusion

The traditional ELM with the square loss function has the disadvantages of overfitting and high sensibility to outliers. A new algorithm called CDRELM, which has a nonconvex and bounded C-loss function and embeds L_1 norm and L_2 norm in objective function, is proposed. CDRELM is used to solve the problems with many outliers, high dimension, and small or medium samples. It also offers a new embedded feature selection which has a strong capability for dimensionality reduction. Furthermore, CDRELM can complete the two processes of prediction and dimensionality reduction at the same time, so that it can speed up training efficiency. The improved PGD algorithm can also make the solving process more rapid and more accurate. Experiments on artificial datasets and benchmark datasets show that CDRELM has better generalization ability and more robustness at a higher learning speed than BPNN, SVR, DRELM, CELM, L_2–ELM, and L_1–ELM.

It should be noted that we only verified the performance for regression. In future work, we will attempt to verify the classification capacity of the proposed algorithm. Based on the comparison of MSE, it is clear that CDRELM obtains a slightly larger value on large datasets. Therefore, how to improve the accuracy and robustness for large sized datasets will be another focus of our future research.

Acknowledgements The authors thank the anonymous reviewers for their constructive comments and suggestions. This work was supported in part by the National Natural Science Foundation of China under Grant 51875457, the Key Research Project of Shaanxi Province (2022GY-050), the Natural Science Foundation of Shaanxi Province of China (2022JQ-636, 2021JQ–701), and the Special Scientific Research Plan Project of Shaanxi Province Education Department (21JK0905).

Declarations

Informed Consent Informed consent was not required as no human beings or animals were involved.

Human and Animal Rights This article does not contain any studies with human or animal subjects performed by any of the authors.

Conflict of Interest The authors declare no competing interests.

References

1. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back–propagating errors. Nature. 1986;323:533–6.
2. Vapnik V, Golowich S, Smola A. Support vector method for function approximation, regression estimation, and signal processing. The 9th Int Conf Neural Inform Proc Sys. 1996;281–287.
3. Furfaro R, Barocco R, Linares R, Topputo F, Reddy V, Simo J, et al. Modeling irregular small bodies gravity field via extreme learning machines and Bayesian optimization. Adv Space Res. 2020;67(1):617–38.
4. Huang GB, Zhu QY, Siew CK. Extreme learning machine: theory and applications. Neurocomputing. 2006;70(1–3):489–501.
5. Kaleem K, Wu YZ, Adjeisah M. Consonant phoneme based extreme learning machine (ELM) recognition model for foreign accent identification. The World Symp Software Eng. 2019;68–72.
6. Liu X, Huang H, Xiang J. A personalized diagnosis method to detect faults in gears using numerical simulation and extreme learning machine. Knowl Based Syst. 2020;195(1): 105653.
7. Fellx A, Daniela G, Liviu V, Mihaela–Alexandra P. Neural network approaches for children's emotion recognition in intelligent learning applications. The 7th Int Conf Education and New Learning Technol. 2015;3229–3239.
8. Huang GB, Zhou H, Ding X. Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern B. 2011;42(2):513–29.
9. Huang S, Zhao G, Chen M. Tensor extreme learning design via generalized Moore-Penrose inverse and triangular type–2 fuzzy sets. Neural Comput Applical. 2018;31:5641–51.
10. Bai Z, Huang GB, Wang D. Sparse Extreme learning machine for classification. IEEE Trans Cybern. 2014;44(10):1858–70.
11. Wang Y, Yang L, Yuan C. A robust outlier control framework for classification designed with family of homotopy loss function. Neural Netw. 2019;112:41–53.
12. Deng WY, Zheng Q, Lin C. Regularized extreme learning machine. IEEE symposium on computational intelligence and data mining. 2009;2009:389–95.
13. Balasundaram S, Gupta D. 1–Norm extreme learning machine for regression and multiclass classification using Newton method. Neurocomputing. 2014;128:4–14.
14. Christine DM, Ernesto DV, Lorenzo R. Elastic–net regularization in learning theory. J complexity. 2009;25(2):201–30.
15. Luo X, Chang XH, Ban XJ. Regression and classification using extreme learning machine based on L-$_1$-norm and L-$_2$-norm. Neurocomputing. 2016;174:179–86.
16. Abhishek S, Rosha P, Jose P. The C–loss function for pattern classification. Pattern Recognit. 2014;47(1):441–53.
17. Zhao YP, Tan JF, Wang JJ. C–loss based extreme learning machine for estimating power of small–scale turbojet engine. Aerosp Sci Technol. 2019;89(6):407–19.

18. Jing TT, Xia HF, and Ding ZM. Adaptively-accumulated knowledge transfer for partial domain adaptation. In Proceedings of the 28th ACM International Conference on Multimedia. 2020;1606–1614.
19. Fu YY, Zhang M, Xu X, et al. Partial feature selection and alignment for multi-source domain adaptation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021;16654–16663.
20. Khalajmehrabadi A, Gatsis N, Pack D. A joint indoor WLAN localization and outlier detection scheme using LASSO and Elastic-Net optimization techniques. IEEE Trans Mob Comput. 2017;16(8):1–1.
21. Boyd S, Vandenberghe L, Faybusovich L. Convex optimization IEEE Trans Automat Contr. 2006;51(11):1859.
22. Huang GB, Wang DH, Lan Y. Extreme learning machines: a survey. Int J Mach Learn Cyb. 2011;2(2):107–22.
23. Peng HY, Liu CL. Discriminative feature selection via employing smooth and robust hinge loss. IEEE T Neur Net Lear. 2019;99:1–15.
24. Lei Z, Mammadov MA. Yearwood J. From convex to nonconvex: a loss function analysis for binary classification. 2010 IEEE International Conference On Data Mining Workshops. 2010;1281–1288.
25. Hajiabadi H, Molla D, Monsefi R, et al. Combination of loss functions for deep text classification. Int J Mach Learn Cyb. 2019;11:751–61.
26. Hajiabadi H, Monsefi R, Yazdi HS. RELF: robust regression extended with ensemble loss function. Appl Intell. 2018;49:473.
27. Zou H, Hastie T. Addendum: Regularization and variable selection via the elastic net. J Roy Stat Soc. 2010;67(5):768–768.
28. Golub GH, Loan CFV. Matrix computations 3rd edition. Johns Hopkins studies in mathematical sciences. 1996.
29. Dinoj S "Swiss roll datasets", http://people.cs.uchicago.edu/~dinoj/manifold/swissroll.html, accessed on 12 Apr 2021.
30. UCI machine learning repository http://archive.ics.uci.edu/ml/datasets.php, accessed on 12 Apr 2021
31. Kaggle datasets https://www.kaggle.com/, accessed on 12 April 2021
32. Hua XG, Ni YQ, Ko JM, et al. Modeling of temperature–frequency correlation using combined principal component analysis and support vector regression technique. J Comput Civil Eng. 2007;21(2):122–35.
33. Frost P, Kailath T. An innovations approach to least–squares estimation—part III: nonlinear estimation in white Gaussian noise. IEEE Trans Automat Contr. 2003;16(3):217–26.
34. Demšar J. Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res. 2006;7:1–30.
35. Iman L, Davenport JM. Approximations of the critical region of the Friedman statistic. Commun Stat–Simul C. 1998;571–595.
36. Fei Z, Webb GI, Suraweera P, et al. Subsumption resolution: an efficient and effective technique for semi–naive Bayesian learning. Mach Learn. 2012;87(1):93–125.