Knowledge and Information Systems (2023) 65:2017–2042
https://doi.org/10.1007/s10115-022-01803-4

REGULAR PAPER

A novel correlation Gaussian process regression-based extreme learning machine

Xuan Ye[1] · Yulin He[1,2] · Manjing Zhang[1] · Philippe Fournier-Viger[2] · Joshua Zhexue Huang[1,2]

Received: 21 June 2022 / Revised: 24 September 2022 / Accepted: 27 November 2022 /
Published online: 10 January 2023

Abstract
An obvious defect of extreme learning machine (ELM) is that its prediction performance is sensitive to the random initialization of input-layer weights and hidden-layer biases. To make ELM insensitive to random initialization, GPRELM adopts the simple an effective strategy of integrating Gaussian process regression into ELM. However, there is a serious overfitting problem in kernel-based GPRELM (*k*GPRELM). In this paper, we investigate the theoretical reasons for the overfitting of *k*GPRELM and further propose a correlation-based GPRELM (*c*GPRELM), which uses a correlation coefficient to measure the similarity between two different hidden-layer output vectors. *c*GPRELM reduces the likelihood that the covariance matrix becomes an identity matrix when the number of hidden-layer nodes is increased, effectively controlling overfitting. Furthermore, *c*GPRELM works well for improper initialization intervals where ELM and *k*GPRELM fail to provide good predictions. The experimental results on real classification and regression data sets demonstrate the feasibility and superiority of *c*GPRELM, as it not only achieves better generalization performance but also has a lower computational complexity.

Keywords Extreme learning machine · Gaussian process regression · Kernel function · Correlation coefficient · Overfitting

✉ Yulin He
yulinhe@gml.ac.cn; csylhe@126.com

Xuan Ye
yexuan@gml.ac.cn

Manjing Zhang
zhangmanjing@gml.ac.cn

Philippe Fournier-Viger
philfv@szu.edu.cn

Joshua Zhexue Huang
zx.huang@szu.edu.cn

1 Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen 518107, China

2 College of Computer Science & Software Engineering, Shenzhen University, Shenzhen 518060, China

Abbreviations

ELM	Extreme learning machine
SLFN	Single hidden-layer feed-forward network
BP	Back-propagation
BLRELM	Bayesian linear regression-based ELM
GPR	Gaussian process regression
GPRELM	GPR-based ELM
1HNBKM	One hidden-layer nonparametric Bayesian kernel machine
RBF	Radial basis function
*k*GPRELM	Kernel-based GPRELM
*c*GPRELM	Correlation-based GPRELM
SVM	Support vector machine
LSSVM	Least square SVM
UCI	University of California, Irvine
KEEL	Knowledge extraction based on evolutionary learning
RMSE	Root-mean-square error

List of symbols

D	Training data set
N	Number of instances in D
x_i	Input of the i-th training instance
y_i	Output of the i-th training instance
D	Number of instance's condition attributes
M	Number of instance's decision attributes
W	Input-layer matrix of ELM
w_{dl}	Weight on the link between the d-th input-layer node and the l-th hidden-layer node
b	Hidden-layer bias vector of ELM
b_l	Bias of the l-th hidden-layer node
H	Hidden-layer output matrix of ELM
$\mathrm{h}(\mathrm{x}_i)$	Hidden-layer output of i-th training instance
β	Output-layer matrix of ELM
$g(\cdot)$	Activation function of hidden-layer node
$k(\cdot, \cdot)$	Kernel function
λ	Kernel radius of kernel function $k(\cdot, \cdot)$
μ	Mean of Gaussian distribution
σ_N^2	Variance of Gaussian distribution
ε	Gaussian noise
I	Identity matrix
C	Correlation matrix
$c(\cdot, \cdot)$	Correlation function
Cov (u, v)	Covariance between vectors u and v
Var (u)	Variance of vector u

1 Introduction

Extreme learning machine (ELM) [12, 13] is a type of noniterative training algorithm for single hidden-layer feed-forward neural network (SLFN). ELM randomly selects the input-layer

weights/hidden-layer biases and analytically calculates the output-layer weights [6]. Because ELM does not perform complex parameter adjustment (e.g., learning rate, learning epoches, stopping criteria) and time-consuming weight updates, ELM is simpler and faster than the traditional back-propagation (BP) algorithm [5, 10]. In addition, the theoretical universal approximation [9] also guarantees that ELM can obtain better generalization capability than BP [4]. ELM has been successfully applied to time series prediction [31], image recognition [27], smart city design [24], and COVID-19 severity detection [8].

However, random sensitivity is one of the obvious defects of ELM and it affects its generalization performance. The random initialization of input-layer weights and hidden-layer biases can result in nonfull rank of the hidden-layer output matrix and make ELM prediction unstable [29]. There are two main strategies to tackle this limitation: using an optimization algorithm and building an ensemble of ELMs. The former strategy employs different evolution algorithms to obtain optimal input weights and hidden biases (e.g., particle swarm optimization in [7] and differential evolution in [34]), whereas the latter integrates multiple ELMs to mitigate the negative effect of randomization on predictive stability (e.g., cross-validation-based ensemble in [19] and selective ensemble in [32]). More studies on this problem can be found in [2, 16, 17, 26, 33]. To some extent, these enhancements overcome ELM's random instability to a certain degree. However, using optimization algorithms and multi-ELMs training considerably increases the computational complexity of learning models. In addition, the optimized weights and biases usually cause ELMs to overfit.

Recently, integrating Bayesian prior knowledge into ELM has been proposed as an improvement to make ELM more insensitive to random initialization [3, 21, 28]. Instead of direct point prediction, these methods estimate the posterior probability distributions of ELM output, reducing the influence of random initialization. Soria-Olivas et al. [28] developed a Bayesian linear regression (BLR)-based ELM model (BLRELM). Then, Luo et al. [21] improved BLRELM and proposed a sparse BLRELM in which each weight has an independent regularization prior rather than all weights sharing a single prior as in BLRELM. Chatzis et al. [3] designed a One-Hidden layer Nonparametric Bayesian Kernel Machine (1HNBKM), which is a Gaussian process regression (GPR)-based ELM model (GPRELM). In probability theory and statistics, a Gaussian process [20] is a stochastic process represented by a collection of random variables indexed by time or space, and every finite collection of those random variables has a multivariate normal distribution. As pointed out by Chatzis et al. [3], BLRELM is a special case of GPRELM. When a linear kernel is considered in GPRELM, BLRELM is reduced to GPRELM. Thus, the majority of our analysis and improvement in this paper are focused on GPRELM, which is a more general form of Bayesian prior knowledge-based ELM. GPRELM is simpler and faster than the complex and time-consuming optimization and ensemble methods mentioned above. It provides a more efficient way of alleviating the random sensitivity of ELM. However, using a radial basis function (RBF) in GPRELM (kGPRELM) results in serious overfitting. In addition, improper initialization intervals (e.g., $[-100, 0]$ or $[0, 100]$) can significantly degrade kGPRELM's predictive accuracies.

The goal of this paper is to solve the overfitting and large interval unavailability problems of GPRELM by replacing the kernel trick in GPR with an alternative technique. The contributions of this cover four main aspects.

(1) The main reason why kGPRELM overfits is theoretically analyzed. Increasing the number of hidden-layer nodes tends to cause kernel inefficiency for kGPRELM.
(2) A correlation-based GPRELM (cGPRELM) method is proposed by calculating the similarity between two hidden-layer output vectors using a correlation index. This alleviates the overfitting problem of GPR-based ELM.

(3) A normalization strategy to normalize the hidden-layer inputs of *c*GPRELM is introduced to reduce the possibility that the hidden-layer output matrix elements become 0 or 1.
(4) Extensive experiments are conducted to compare the effectiveness of *c*GPRELM with standard ELM, *k*GPRELM, and multilayer ELM (ML-ELM) [15].

The remainder of this paper is organized as follows: In Sect. 2, we provide a brief overview of kernel-based GPRELM (*k*GPRELM). Section 3 introduces the proposed correlation-based GPRELM (*c*GPRELM) method. Experimental simulations are presented in Sect. 4. Finally, in Sect. 5, we conclude this paper and outline the main directions for future research.

2 *k*GPRELM: kernel-based GPRELM

This section first describes the fundamental concepts of ELM and *k*GPRELM and then analyzes *k*GPRELM's overfitting from a theoretical perspective.

2.1 Original ELM

Given a training data set

$$\mathrm{D} = \left\{(\mathrm{x}_i, \mathrm{y}_i)_{i=1}^{N} | \mathrm{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iD}),\ \mathrm{y}_i = (y_{i1}, y_{i2}, \ldots, y_{iM})\right\}, \tag{1}$$

which includes N distinct instances with D condition attributes, ELM [11, 13] calculates the output-layer weight matrix

$$\beta = \mathrm{H}^{\dagger}\mathrm{Y}, \tag{2}$$

where $\mathrm{H}^{\dagger}$ is the Moore-Penrose generalized inverse of the hidden-layer output matrix

$$\mathrm{H} = \begin{bmatrix} g(\mathrm{w}_1\mathrm{x}_1 + b_1) & g(\mathrm{w}_2\mathrm{x}_1 + b_2) & \cdots & g(\mathrm{w}_L\mathrm{x}_1 + b_L) \\ g(\mathrm{w}_1\mathrm{x}_2 + b_1) & g(\mathrm{w}_2\mathrm{x}_2 + b_2) & \cdots & g(\mathrm{w}_L\mathrm{x}_2 + b_L) \\ \vdots & \vdots & \ddots & \vdots \\ g(\mathrm{w}_1\mathrm{x}_N + b_1) & g(\mathrm{w}_2\mathrm{x}_N + b_2) & \cdots & g(\mathrm{w}_L\mathrm{x}_N + b_L) \end{bmatrix}, \tag{3}$$

$g(v) = \frac{1}{1+e^{-v}}$ is the sigmoid activation function, L is the number of hidden-layer nodes in SLFN, the input-layer weight matrix

$$\mathrm{W} = \begin{bmatrix} \mathrm{w}_1 \\ \mathrm{w}_2 \\ \vdots \\ \mathrm{w}_L \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} w_{11} & w_{21} & \cdots & w_{D1} \\ w_{12} & w_{22} & \cdots & w_{D2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1L} & w_{2L} & \cdots & w_{DL} \end{bmatrix} \tag{4}$$

and hidden-layer bias vector $\mathrm{b} = [b_1, b_2, \cdots, b_L]$ are randomly generated according to *any continuous probability distribution* [13], the training output is a matrix

$$\mathrm{Y} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1M} \\ y_{21} & y_{22} & \cdots & y_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{NM} \end{bmatrix} \quad (M > 1) \tag{5}$$

in the case of a classification problem and a vector

$$\mathrm{Y} = \begin{bmatrix} y_{11} \\ y_{21} \\ \vdots \\ y_{N1} \end{bmatrix} \quad (M = 1) \tag{6}$$

for a regression problem, M is the number of instance decision attributes. For an unseen instance $\hat{\mathrm{x}} = \left(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_D\right)$, ELM predicts its output $\hat{\mathrm{y}}$ as follows:

$$\hat{\mathrm{y}} = \mathrm{h}\left(\hat{\mathrm{x}}\right)\beta = \mathrm{h}\left(\hat{\mathrm{x}}\right)\mathrm{H}^{\dagger}\mathrm{Y}, \tag{7}$$

where $\mathrm{h}\left(\hat{\mathrm{x}}\right) = \left[g\left(\mathrm{w}_1\hat{\mathrm{x}} + b_1\right), \ldots, g\left(\mathrm{w}_L\hat{\mathrm{x}} + b_L\right)\right]$.

Because ELM does not require iterative adjustments to weights and biases like SLFN, ELM's training speed can be thousands of times faster than BP [13]. ELM can achieve equal generalization performances with support vector machine (SVM) and least square SVM (LSSVM) [11]. From Eq. (7), we can find that the predictive accuracy of ELM mainly depends on the calculation of $\mathrm{H}^{\dagger}$. Sometimes, the random selection of input-layer weights W and hidden-layer biases b can produce a nonsingular hidden-layer output matrix H, resulting in no solution for the linear system $\mathrm{H}\beta = \mathrm{Y}$ and lowering ELM's predictive accuracy [29]. This makes ELM's prediction unstable and indicates that it is sensitive to random initialization.

2.2 *k*GPRELM

*k*GPRELM is a recently proposed ELM variant that uses the Bayesian optimization method [23] to improve ELM's random sensitivity. It is called one-hidden layer Bayesian kernel machine (1HNBKM) [3]. It predicts the output $\hat{\mathrm{y}}$ for an unseen instance $\hat{\mathrm{x}}$ according to the following joint Gaussian distribution:

$$\begin{bmatrix} \mathrm{Y} \\ \hat{\mathrm{y}} \end{bmatrix} \sim N\left[0, \begin{bmatrix} \mathrm{K}\left(\mathrm{H}, \mathrm{H}\right) & \mathrm{k}^{\mathrm{T}}\left(\mathrm{h}\left(\hat{\mathrm{x}}\right), \mathrm{H}\right) \\ \mathrm{k}\left(\mathrm{h}\left(\hat{\mathrm{x}}\right), \mathrm{H}\right) & k\left(\mathrm{h}\left(\hat{\mathrm{x}}\right), \mathrm{h}\left(\hat{\mathrm{x}}\right)\right) \end{bmatrix}\right], \tag{8}$$

where

$$\mathrm{h}\left(\mathrm{x}_i\right) = \left[g\left(\mathrm{w}_1\mathrm{x}_i + b_1\right), \ldots, g\left(\mathrm{w}_L\mathrm{x}_i + b_L\right)\right] \tag{9}$$

is the hidden-layer output vector of the i-th ($i = 1, 2, \ldots, N$) training instance,

$$\mathrm{K}\left(\mathrm{H}, \mathrm{H}\right) = \begin{bmatrix} k\left(\mathrm{h}\left(\mathrm{x}_1\right), \mathrm{h}\left(\mathrm{x}_1\right)\right) & \cdots & k\left(\mathrm{h}\left(\mathrm{x}_1\right), \mathrm{h}\left(\mathrm{x}_N\right)\right) \\ \vdots & \ddots & \vdots \\ k\left(\mathrm{h}\left(\mathrm{x}_N\right), \mathrm{h}\left(\mathrm{x}_1\right)\right) & \cdots & k\left(\mathrm{h}\left(\mathrm{x}_N\right), \mathrm{h}\left(\mathrm{x}_N\right)\right) \end{bmatrix} \tag{10}$$

is the kernel matrix,

$$\mathrm{k}\left(\mathrm{h}\left(\hat{\mathrm{x}}\right), \mathrm{H}\right) = \left[k\left(\mathrm{h}\left(\hat{\mathrm{x}}\right), \mathrm{h}\left(\mathrm{x}_1\right)\right), \ldots, k\left(\mathrm{h}\left(\hat{\mathrm{x}}\right), \mathrm{h}\left(\mathrm{x}_N\right)\right)\right] \tag{11}$$

is the kernel vector, and

$$k\left(\mathrm{u}, \mathrm{v}\right) = \exp\left(-\frac{\|\mathrm{u} - \mathrm{v}\|^2}{2\lambda^2}\right) \tag{12}$$

is the radial basis function (RBF) kernel.

From Eq. (8), we can derive the posterior distribution of the predicted output $\hat{y}$ as

$$\mathrm{P}\left(\hat{y}\left|\mathrm{h}\left(\hat{x}\right), \mathrm{H}, \mathrm{Y}\right.\right) = N\left(\mu, \sigma^2\right), \tag{13}$$

where the mean and variance of this Gaussian distribution are

$$\mu = \mathrm{k}\left(\mathrm{h}\left(\hat{x}\right), \mathrm{H}\right)\left[\mathrm{K}\left(\mathrm{H}, \mathrm{H}\right) + \sigma_N^2 \mathrm{I}\right]^{-1} \mathrm{Y} \tag{14}$$

and

$$\sigma^2 = k\left(\mathrm{h}\left(\hat{x}\right), \mathrm{h}\left(\hat{x}\right)\right) - \mathrm{k}\left(\mathrm{h}\left(\hat{x}\right), \mathrm{H}\right)\left[\mathrm{K}\left(\mathrm{H}, \mathrm{H}\right) + \sigma_N^2 \mathrm{I}\right]^{-1} \mathrm{k}^{\mathrm{T}}\left(\mathrm{h}\left(\hat{x}\right), \mathrm{H}\right), \tag{15}$$

respectively, and I is a N-by-N identity matrix. In 1HNBKM, μ is used as the predictive output of the unseen instance $\hat{x}$, i.e., let

$$\hat{y} = \mathrm{k}\left(\mathrm{h}\left(\hat{x}\right), \mathrm{H}\right)\left[\mathrm{K}\left(\mathrm{H}, \mathrm{H}\right) + \sigma_N^2 \mathrm{I}\right]^{-1} \mathrm{Y}. \tag{16}$$

$[\mu - 1.96\sigma, \mu + 1.96\sigma]$ be a 95% confidence region for the estimation of the unknown $\hat{y}$. So far, there is one parameter that we have not discussed, which is σ_N^2 in Eqs. (14)–(16). This parameter is related to Gaussian process regression (GPR) [22], which assumes that

$$\hat{y} = \mathrm{h}\left(\hat{x}\right)\beta + \varepsilon, \tag{17}$$

where the noise ε obeys a Gaussian distribution $N\left(0, \sigma_N^2\right)$.

From the aforementioned reasoning process, we can clearly see that 1HNBKM actually introduces GPR into ELM, i.e., using RBF kernel-based GPR to solve the linear system in Eq. (17) and predict the output for $\hat{x}$. Thus, in this paper, the so-called one-hidden layer Bayesian kernel machine is called kernel-based GPRELM (*k*GPRELM) to indicate that 1HNBKM is an improved variant of the original ELM.

2.3 Overfitting analysis of *k*GPRELM

ELM [13] is sensitive to random initialization and thus generates unstable predictions. *k*GPRELM [3] improves the usability of ELM, but it suffers from serious overfitting. The following derivation process provides a theoretical analysis of *k*GPRELM's overfitting.

For a given λ, SLFN with L_1 hidden-layer nodes calculates the hidden-layer output vectors for any two instances $\mathrm{x}_i, \mathrm{x}_j \in \mathrm{D}$ as

$$\mathrm{h}_1\left(\mathrm{x}_i\right) = \left[g\left(\mathrm{w}_1 \mathrm{x}_i + b_1\right), \ldots, g\left(\mathrm{w}_{L_1} \mathrm{x}_i + b_{L_1}\right)\right] \tag{18}$$

and

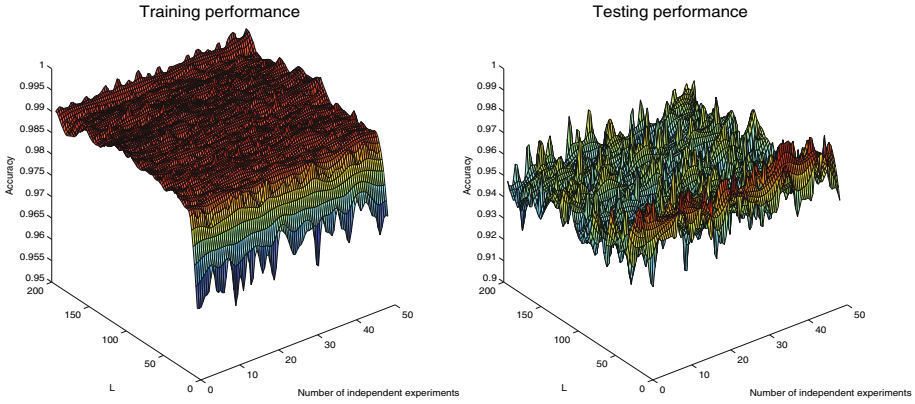
$$\mathrm{h}_1\left(\mathrm{x}_j\right) = \left[g\left(\mathrm{w}_1 \mathrm{x}_j + b_1\right), \ldots, g\left(\mathrm{w}_{L_1} \mathrm{x}_j + b_{L_1}\right)\right], \tag{19}$$

respectively. When the hidden-layer nodes are incrementally increased to L_2 ($L_2 > L_1$), the hidden-layer output vectors of x_i and x_j are changed into

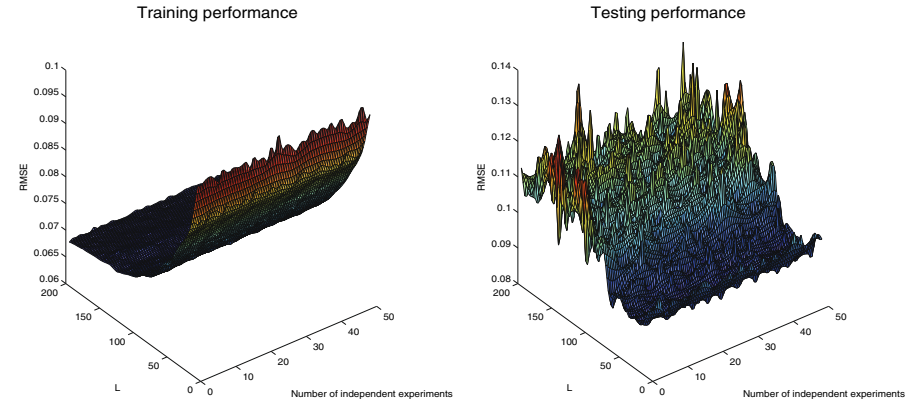
$$\mathrm{h}_2\left(\mathrm{x}_i\right) = \left[g\left(\mathrm{w}_1 \mathrm{x}_i + b_1\right), \ldots, g\left(\mathrm{w}_{L_1} \mathrm{x}_i + b_{L_1}\right),\ g\left(\mathrm{w}_{L_1+1} \mathrm{x}_i + b_{L_1+1}\right), \ldots, g\left(\mathrm{w}_{L_2} \mathrm{x}_i + b_{L_2}\right)\right] \tag{20}$$

and

$$\mathrm{h}_2\left(\mathrm{x}_j\right) = \left[g\left(\mathrm{w}_1 \mathrm{x}_j + b_1\right), \ldots, g\left(\mathrm{w}_{L_1} \mathrm{x}_j + b_{L_1}\right),\ g\left(\mathrm{w}_{L_1+1} \mathrm{x}_j + b_{L_1+1}\right), \ldots, g\left(\mathrm{w}_{L_2} \mathrm{x}_j + b_{L_2}\right)\right]. \tag{21}$$



(a) On *Iris* data set

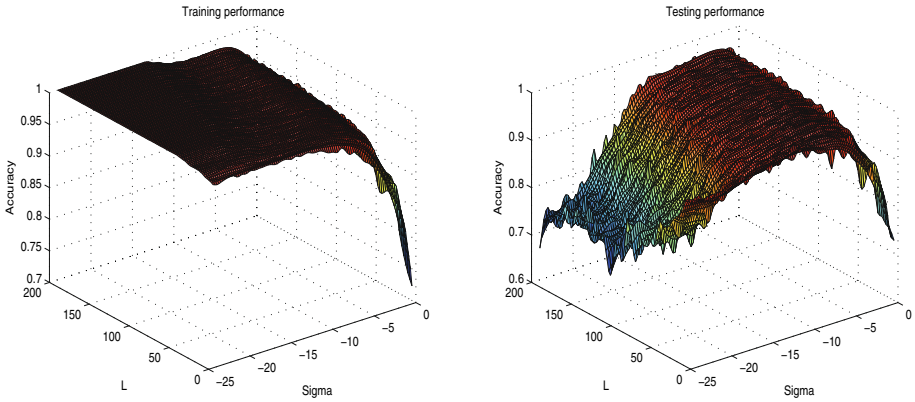


(b) On *DEE* data set

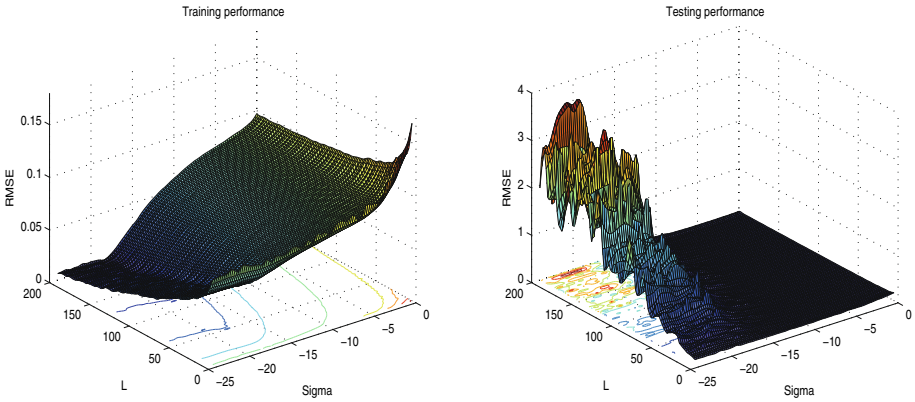
Fig. 1 ELM's instability

We can calculate

$$k\left(\mathrm{h}_1\left(\mathrm{x}_i\right), \mathrm{h}_1\left(\mathrm{x}_j\right)\right) = \exp\left[-\frac{\left\|\mathrm{h}_1\left(\mathrm{x}_i\right) - \mathrm{h}_1\left(\mathrm{x}_j\right)\right\|^2}{2\lambda^2}\right]$$
$$= \exp\left[-\frac{\sum_{l=1}^{L_1}\left[g\left(\mathrm{w}_l\mathrm{x}_i + b_l\right) - g\left(\mathrm{w}_l\mathrm{x}_j + b_l\right)\right]^2}{2\lambda^2}\right] \quad (22)$$



(a) On *Iris* data set



(b) On *DEE* data set

Fig. 2 kGPRELM's overfitting (for a fixed $\lambda = 1$)

and

$$k\left(\mathrm{h}_2\left(\mathrm{x}_i\right), \mathrm{h}_2\left(\mathrm{x}_j\right)\right) = \exp\left[-\frac{\left\|\mathrm{h}_2\left(\mathrm{x}_i\right) - \mathrm{h}_2\left(\mathrm{x}_j\right)\right\|^2}{2\lambda^2}\right]$$
$$= \exp\left[\begin{array}{c} -\frac{\sum_{l=1}^{L_1}\left[g\left(\mathrm{w}_l\mathrm{x}_i+b_l\right)-g\left(\mathrm{w}_l\mathrm{x}_j+b_l\right)\right]^2}{2\lambda^2} \\ -\frac{\sum_{l=L_1+1}^{L_2}\left[g\left(\mathrm{w}_l\mathrm{x}_i+b_l\right)-g\left(\mathrm{w}_l\mathrm{x}_j+b_l\right)\right]^2}{2\lambda^2} \end{array}\right] \tag{23}$$

Then, the following inequality

$$k\left(\mathrm{h}_1\left(\mathrm{x}_i\right), \mathrm{h}_1\left(\mathrm{x}_j\right)\right) > k\left(\mathrm{h}_2\left(\mathrm{x}_i\right), \mathrm{h}_2\left(\mathrm{x}_j\right)\right) \tag{24}$$

is derived. This indicates that as the number of hidden-layer nodes is increased in SLFN, $k\left(\mathrm{h}\left(\mathrm{x}_i\right), \mathrm{h}\left(\mathrm{x}_j\right)\right)$ is monotonically decreasing and converges toward 0, i.e., for the fixed λ,

$$\lim_{L\to+\infty} k\left(\mathrm{h}\left(\mathrm{x}_i\right), \mathrm{h}\left(\mathrm{x}_j\right)\right) = 0. \tag{25}$$

Equation (25) leads to $\mathrm{K}(\mathrm{H}, \mathrm{H}) \to \mathrm{I}$. For the training input matrix

$$\mathrm{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix}, \tag{26}$$

its predictive output matrix $\hat{\mathrm{Y}}$ can be easily calculated with kGPRELM as

$$\hat{\mathrm{Y}} = \mathrm{K}(\mathrm{H}, \mathrm{H}) \left[\mathrm{K}(\mathrm{H}, \mathrm{H}) + \sigma_N^2 \mathrm{I}\right]^{-1} \mathrm{Y} \to \mathrm{Y}, \tag{27}$$

when $L \to +\infty$ and $\sigma_N^2 \to 0$. Equation (27) shows that kGPRELM will seriously overfit for a large number of hidden-layer nodes and small noise.

We present a simple experimental simulation to demonstrate the instability of ELM and the overfitting of kGPRELM using the *Iris* classification and *Daily Electricity Energy (DEE)* regression data sets, which are, respectively, selected from the UCI [18] and KEEL [1] machine learning repositories. Classification accuracy and root-mean-square error (RMSE) are used to assess the classification and regression performance of ELM [13] and kGPRELM [3]. Each experimental result was obtained using a 10 times twofold cross-validation procedure. SLFN weights and biases are *Uniform* random numbers selected in the [0, 1] interval. The highly fluctuating learning curves in the right sub-figures of Fig. 1 demonstrate ELM's instability. Higher training accuracy/lower training RMSE and lower testing accuracy/higher testing RMSE in Fig. 2 show that kGPRELM is seriously overfitting, but it is more stable than ELM.

3 *c*GPRELM: correlation-based GPRELM

According to the analysis of Sect. 2.3, the main cause of kGPRELM's overfitting is the RBF kernel $k\left(\mathrm{h}(\mathrm{x}_i), \mathrm{h}\left(\mathrm{x}_j\right)\right) \to 0$ for any $i \neq j$ when $L \to +\infty$ (or $\lambda^2 \to 0$) and $\sigma_N^2 \to 0$. In the classical GPR, the kernel function is introduced to guarantee the positive semi-definition of the Gaussian distribution's covariance matrix in Eq. (8), which evaluates the correlation between two outputs by measuring the similarity between the corresponding inputs [30]. This similarity between input pairs is represented by the inner product, which is calculated in a high-dimensional projected space. It is very difficult to determine the specific form of the original input in high-dimensional space. Thus, a kernel trick is applied in the original input space to calculate the inner product in projected space. That is to say, the kernel operation includes a one-time projection from the original space to a high-dimensional space. In fact, there is another projection in kGPRELM, i.e., from the D-dimensional input x_i to the L-dimensional hidden-layer output $\mathrm{h}(\mathrm{x}_i)$ of ELM. The relationship between these two projections can be represented as

$$\left.\begin{array}{l} \mathrm{x}_i \overset{\text{ELM}}{\to} \mathrm{h}(\mathrm{x}_i) \\ \mathrm{x}_j \overset{\text{ELM}}{\to} \mathrm{h}\left(\mathrm{x}_j\right) \end{array}\right\} \overset{\text{RBF Kernel}}{\to} k\left(\mathrm{h}(\mathrm{x}_i), \mathrm{h}\left(\mathrm{x}_j\right)\right). \tag{28}$$

The kernel's role in kGPRELM is to measure the similarity between two different hidden-layer output vectors $\mathrm{h}(\mathrm{x}_i)$ and $\mathrm{h}\left(\mathrm{x}_j\right)$. The second projection for hidden-layer outputs with RBF kernel leads to overfitting and redundancy if an alternative criterion can be used to measure the similarity between hidden-layer outputs obtained by the first projection.

In this paper, we propose a correlation-based GPRELM (*c*GPRELM) which uses a correlation coefficient defined as

$$c\left(\mathrm{h}\left(\mathrm{x}_i\right), \mathrm{h}\left(\mathrm{x}_j\right)\right) = \frac{1}{2}\left[\frac{\mathrm{Cov}\left(\mathrm{h}\left(\mathrm{x}_i\right), \mathrm{h}\left(\mathrm{x}_j\right)\right)}{\mathrm{Var}\left(\mathrm{h}\left(\mathrm{x}_i\right)\right)\mathrm{Var}\left(\mathrm{h}\left(\mathrm{x}_j\right)\right)} + 1\right], \tag{29}$$

to measure the similarity between $\mathrm{h}\left(\mathrm{x}_i\right)$ and $\mathrm{h}\left(\mathrm{x}_j\right)$, where $\mathrm{Cov}\left(\mathrm{h}\left(\mathrm{x}_i\right), \mathrm{h}\left(\mathrm{x}_j\right)\right)$ is the covariance between $\mathrm{h}\left(\mathrm{x}_i\right)$ and $\mathrm{h}\left(\mathrm{x}_j\right)$, and $\mathrm{Var}\left(\mathrm{h}\left(\mathrm{x}_i\right)\right)$ and $\mathrm{Var}\left(\mathrm{h}\left(\mathrm{x}_j\right)\right)$ are variances of $\mathrm{h}\left(\mathrm{x}_i\right)$ and $\mathrm{h}\left(\mathrm{x}_j\right)$, respectively. *c*GPRELM predicts the output $\hat{\mathrm{y}}$ for an unseen instance $\hat{\mathrm{x}}$ based on the following formulation

$$\hat{\mathrm{y}} = \mathrm{c}\left(\mathrm{h}\left(\hat{\mathrm{x}}\right), \mathrm{H}\right)\left[\mathrm{C}\left(\mathrm{H}, \mathrm{H}\right) + \sigma_N^2\mathrm{I}\right]^{-1}\mathrm{Y}, \tag{30}$$

where

$$\mathrm{C}\left(\mathrm{H}, \mathrm{H}\right) = \begin{bmatrix} c\left(\mathrm{h}\left(\mathrm{x}_1\right), \mathrm{h}\left(\mathrm{x}_1\right)\right) & \cdots & c\left(\mathrm{h}\left(\mathrm{x}_1\right), \mathrm{h}\left(\mathrm{x}_N\right)\right) \\ \vdots & \ddots & \vdots \\ c\left(\mathrm{h}\left(\mathrm{x}_N\right), \mathrm{h}\left(\mathrm{x}_1\right)\right) & \cdots & c\left(\mathrm{h}\left(\mathrm{x}_N\right), \mathrm{h}\left(\mathrm{x}_N\right)\right) \end{bmatrix} \tag{31}$$

is the correlation matrix and

$$\mathrm{c}\left(\mathrm{h}\left(\hat{\mathrm{x}}\right), \mathrm{H}\right) = \left[c\left(\mathrm{h}\left(\hat{\mathrm{x}}\right), \mathrm{h}\left(\mathrm{x}_1\right)\right), \cdots, c\left(\mathrm{h}\left(\hat{\mathrm{x}}\right), \mathrm{h}\left(\mathrm{x}_N\right)\right)\right] \tag{32}$$

is the correlation vector.

The covariance matrix in Eq. (8) for the joint Gaussian distribution should be positive semi-definite [25]; thus, GPR requires a kernel matrix $\mathrm{K}\left(\mathrm{H}, \mathrm{H}\right)$ in *k*GPRELM that is also positive semi-definite. Hence, the positive semi-definition property of the correlation matrix $\mathrm{C}\left(\mathrm{H}, \mathrm{H}\right)$ is also necessary and important for the rationality of *c*GPRELM. The requirement of the positive semi-definite property ensures the existence of an inverse matrix for $\mathrm{C}\left(\mathrm{H}, \mathrm{H}\right) + \sigma_N^2\mathrm{I}$ in Eq. (30). Theorem 1 proves the positive semi-definition property of $\mathrm{C}\left(\mathrm{H}, \mathrm{H}\right)$ and then guarantees the rationality of Eq. (30).

Theorem 1 $\mathrm{zC}\left(\mathrm{H}, \mathrm{H}\right)\mathrm{z}^{\mathrm{T}} \geq 0$ holds for any nonzero row vector $\mathrm{z} = (z_1, z_2, \ldots, z_N)$.

Proof We can calculate

$$\begin{aligned} \mathrm{zC}\left(\mathrm{H}, \mathrm{H}\right)\mathrm{z}^{\mathrm{T}} &= \sum_{j=1}^{N}\sum_{i=1}^{N}\left[z_i c\left(\mathrm{h}\left(\mathrm{x}_i\right), \mathrm{h}\left(\mathrm{x}_j\right)\right) z_j\right] \\ &= \sum_{j=1}^{N}\sum_{i=1}^{N}\left[z_i \frac{1}{2}\left[\frac{\mathrm{Cov}\left(\mathrm{h}\left(\mathrm{x}_i\right), \mathrm{h}\left(\mathrm{x}_j\right)\right)}{\mathrm{Var}\left(\mathrm{h}\left(\mathrm{x}_i\right)\right)\mathrm{Var}\left(\mathrm{h}\left(\mathrm{x}_j\right)\right)} + 1\right] z_j\right] \\ &= \frac{1}{2}\sum_{j=1}^{N}\sum_{i=1}^{N}\left[\frac{z_i}{\mathrm{Var}\left(\mathrm{h}\left(\mathrm{x}_i\right)\right)}\mathrm{Cov}\left(\mathrm{h}\left(\mathrm{x}_i\right), \mathrm{h}\left(\mathrm{x}_j\right)\right)\frac{z_j}{\mathrm{Var}\left(\mathrm{h}\left(\mathrm{x}_j\right)\right)}\right] + \frac{1}{2}\left[\sum_{j=1}^{N}\left[z_j\sum_{i=1}^{N} z_i\right]\right] \\ &= \hat{\mathrm{z}}\mathrm{COV}\hat{\mathrm{z}}^{\mathrm{T}} + \frac{1}{2}\left[\sum_{i=1}^{N} z_i\right]^2, \end{aligned} \tag{33}$$

where

$$\hat{\mathrm{z}} = \left[\frac{z_1}{\mathrm{Var}\left(\mathrm{h}\left(\mathrm{x}_1\right)\right)}, \frac{z_2}{\mathrm{Var}\left(\mathrm{h}\left(\mathrm{x}_2\right)\right)}, \ldots, \frac{z_N}{\mathrm{Var}\left(\mathrm{h}\left(\mathrm{x}_N\right)\right)}\right] \tag{34}$$

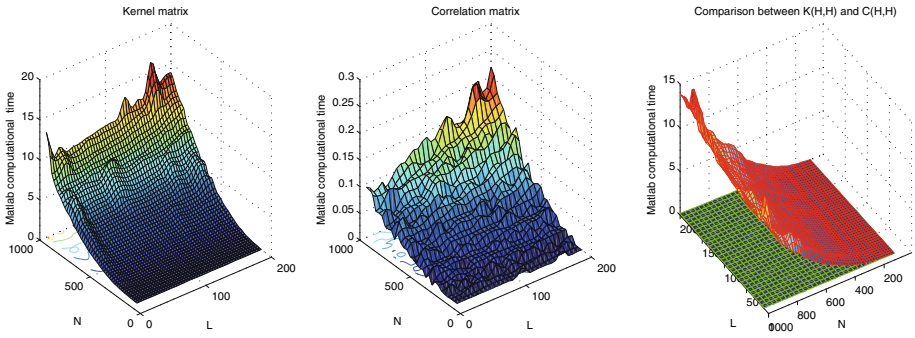


Fig. 3 Comparison of MATLAB computational time between kernel and correlation matrices in *k*GPRELM and *c*GPRELM

and

$$\mathrm{COV} = \begin{bmatrix} \mathrm{Cov}\left(\mathrm{h}\left(\mathrm{x}_1\right), \mathrm{h}\left(\mathrm{x}_1\right)\right) & \cdots & \mathrm{Cov}\left(\mathrm{h}\left(\mathrm{x}_1\right), \mathrm{h}\left(\mathrm{x}_N\right)\right) \\ \vdots & \ddots & \vdots \\ \mathrm{Cov}\left(\mathrm{h}\left(\mathrm{x}_N\right), \mathrm{h}\left(\mathrm{x}_1\right)\right) & \cdots & \mathrm{Cov}\left(\mathrm{h}\left(\mathrm{x}_N\right), \mathrm{h}\left(\mathrm{x}_N\right)\right) \end{bmatrix} \tag{35}$$

is the covariance matrix. Because the covariance matrix of any real random vector is always positive semi-definite, then we can get

$$\hat{\mathrm{z}}\mathrm{COV}\hat{\mathrm{z}}^{\mathrm{T}} \geq 0. \tag{36}$$

Moreover, $\frac{1}{2}\left[\sum_{i=1}^{N} z_i\right]^2 \geq 0$ holds for any nonzero row vector z. In conclusion, we obtain $\mathrm{zC}\left(\mathrm{H}, \mathrm{H}\right)\mathrm{z}^{\mathrm{T}} \geq 0$. □

For the given parameter $\sigma_N^2 > 0$ and any nonzero row vector z, $\mathrm{C}\left(\mathrm{H}, \mathrm{H}\right) + \sigma_N^2\mathrm{I}$ is a positive definite matrix because

$$\mathrm{z}\left[\mathrm{C}\left(\mathrm{H}, \mathrm{H}\right) + \sigma_N^2\mathrm{I}\right]\mathrm{z}^{\mathrm{T}} = \mathrm{zC}\left(\mathrm{H}, \mathrm{H}\right)\mathrm{z}^{\mathrm{T}} + \sigma_N^2\mathrm{zz}^{\mathrm{T}} > 0 \tag{37}$$

holds according to the conclusion of Theorem 1. Applying the correlation coefficient shown in Eq. (29) has the following advantages.

(1) The purpose of Eq. (29) is to assess the correlation between two hidden-layer outputs $\mathrm{h}\left(\mathrm{x}_i\right)$ and $\mathrm{h}\left(\mathrm{x}_j\right)$ as $c\left(\mathrm{h}\left(\mathrm{x}_i\right), \mathrm{h}\left(\mathrm{x}_j\right)\right) \in [0, 1]$. When there is a perfectly linear negative correlation between $\mathrm{h}\left(\mathrm{x}_i\right)$ and $\mathrm{h}\left(\mathrm{x}_j\right)$, $c\left(\mathrm{h}\left(\mathrm{x}_i\right), \mathrm{h}\left(\mathrm{x}_j\right)\right) = 0$. That is $c\left(\mathrm{h}\left(\mathrm{x}_i\right), \mathrm{h}\left(\mathrm{x}_j\right)\right) \to 0$ is determined by the linear correlation between $\mathrm{h}\left(\mathrm{x}_i\right)$ and $\mathrm{h}\left(\mathrm{x}_j\right)$ rather than the number L of hidden-layer nodes in SLFN. Thus, Eq. (29) can more effectively control overfitting than the RBF kernel in *k*GPRELM.

(2) There is no user-specified parameter in Eq. (29), whereas the RBF kernel in *k*GPRELM includes an undetermined parameter, i.e., λ. In the following experimental validation, we find that λ seriously influences the rank of $\mathrm{K}\left(\mathrm{H}, \mathrm{H}\right) + \sigma_N^2\mathrm{I}$.

(3) The computational time of $c\left(\mathrm{h}\left(\mathrm{x}_i\right), \mathrm{h}\left(\mathrm{x}_j\right)\right)$ is lower than $k\left(\mathrm{h}\left(\mathrm{x}_i\right), \mathrm{h}\left(\mathrm{x}_j\right)\right)$ in the MATLAB environment. A series of hidden-layer output matrices $\mathrm{H}_{N\times L}$ were randomly generated corresponding to 1800 different pairs of (N, L), where $N = \{110, 120, \ldots, 1000\}$ and $L = \{10, 20, \ldots, 200\}$. We compared the computational time of $\mathrm{K}\left(\mathrm{H}, \mathrm{H}\right)$ and $\mathrm{C}\left(\mathrm{H}, \mathrm{H}\right)$ in *k*GPRELM and *c*GPRELM. The experimental results are presented in Fig. 3,

which show that the computational time of *c*GPRELM (green curves in the third picture) is far less than that of *k*GPRELM (red curves in the third picture).

We now introduce the main framework of the novel *c*GPRELM method, which uses correlation-based GPR to handle the hidden-layer outputs of SLFN. Here, the hidden-layer output $\mathrm{h}\left(\hat{\mathrm{x}}_i\right)$ corresponding to the i-th training instance $\hat{\mathrm{x}}_i \in \mathrm{D}$ depends on the input-layer weights W and hidden-layer biases b. Huang et al. [13] demonstrated that for any W and b selected from any intervals, $\|\mathrm{H}\beta - \mathrm{Y}\| < \varepsilon$ holds for any small positive value ε with probability 1. However, we find that improper w_{jl} and b_l ($j = 1, 2, \ldots, D; l = 1, 2, \ldots, L$) can seriously degrade the classification accuracies or regression RMSEs of ELM, *k*GPRELM, and *c*GPRELM. Assume that the inputs and output (only for regression) of instances are all normalized into the [0, 1] interval, in this paper. If w_{jl} and b_l were randomly selected from interval $[-100, 0]$ or $[0, 100]$, we would obtain

$$g\left(\mathrm{w}_l \mathrm{x}_i + b_l\right) \to 0 \text{ or } g\left(\mathrm{w}_l \mathrm{x}_i + b_l\right) \to 1. \tag{38}$$

This is caused by the mathematical property of the sigmoid activation function $g(v) = \frac{1}{1+e^{-v}}$. In MATLAB, when $v \leq -12.206$, $g(v) \leq 0.00001$; when $v \geq 11.108$, $g(v) \geq 0.99999$. The improper initialization intervals easily result in

$$\mathrm{w}_l \mathrm{x}_i + b_l \leq -12.206 \text{ or } \mathrm{w}_l \mathrm{x}_i + b_l \geq 11.108. \tag{39}$$

For ELM, this leads to

$$\mathrm{H} \to \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \text{ or } \mathrm{H} \to \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{40}$$

and results in very inaccurate training and testing. For *k*GPRELM and *c*GPRELM, it brings about

$$\mathrm{K}(\mathrm{H}, \mathrm{H}) \to \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{41}$$

and

$$\mathrm{C}(\mathrm{H}, \mathrm{H}) \to \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}, \tag{42}$$

respectively. As a consequence, GPRELM is unable to accurately measure the similarity between pairs of hidden-layer outputs corresponding to different SLFN inputs.

In this paper, we design a normalization method to normalize the hidden-layer inputs of SLFN in *c*GPRELM, which calculates $\mathrm{h}(\mathrm{x}_i)$ as

$$\mathrm{h}(\mathrm{x}_i) = [h_{i1}, h_{i2}, \ldots, h_{iL}], \tag{43}$$

where

$$h_{il} = g\left[\frac{\mathrm{w}_l \mathrm{x}_i + b_l}{(\mathrm{D}+1) \times \max\left[\mathrm{abs}(\mathrm{W}), \mathrm{abs}(\mathrm{b})\right]}\right], l = 1, 2, \ldots, L \tag{44}$$

and max [abs (W) , abs (b)] is the maximum of the absolute components from the input-layer weight matrix W and hidden-layer bias vector b. Then, Cov $\left(\mathrm{h}\left(\mathrm{x}_i\right), \mathrm{h}\left(\mathrm{x}_j\right)\right)$, Var $\left(\mathrm{h}\left(\mathrm{x}_i\right)\right)$ and Var $\left(\mathrm{h}\left(\mathrm{x}_j\right)\right)$ for Eq. (30) in *c*GPRELM are

$$\mathrm{Cov}\left(\mathrm{h}\left(\mathrm{x}_i\right), \mathrm{h}\left(\mathrm{x}_j\right)\right) = \sum_{l=1}^{L}\left[\left(h_{il} - \bar{h}_i\right)\left(h_{jl} - \bar{h}_j\right)\right], \tag{45}$$

$$\mathrm{Var}\left(\mathrm{h}\left(\mathrm{x}_i\right)\right) = \sqrt{\sum_{l=1}^{L}\left(h_{il} - \bar{h}_i\right)^2}, \tag{46}$$

and

$$\mathrm{Var}\left(\mathrm{h}\left(\mathrm{x}_j\right)\right) = \sqrt{\sum_{l=1}^{L}\left(h_{jl} - \bar{h}_j\right)^2}, \tag{47}$$

where $\bar{h}_i = \frac{\sum_{l=1}^{L} h_{il}}{L}$ and $\bar{h}_j = \frac{\sum_{l=1}^{L} h_{jl}}{L}$. Assume that random weights and biases are selected for interval $[\mathrm{R}_1, \mathrm{R}_2]$. We can derive

$$\begin{aligned} &\frac{\mathrm{w}_l \mathrm{x}_i + b_l}{(D+1) \times \max\left[\mathrm{abs}\left(\mathrm{W}\right), \mathrm{abs}\left(\mathrm{b}\right)\right]} \\ &\in \begin{cases} \left[\frac{\mathrm{R}_1}{\mathrm{abs}(\mathrm{R}_1)}, \frac{\mathrm{R}_2}{\mathrm{abs}(\mathrm{R}_1)}\right] \subset [-1, 0), & \text{if } \mathrm{abs}\left(\mathrm{R}_1\right) \geq \mathrm{abs}\left(\mathrm{R}_2\right) \cdot \\ \left[\frac{\mathrm{R}_1}{\mathrm{abs}(\mathrm{R}_2)}, \frac{\mathrm{R}_2}{\mathrm{abs}(\mathrm{R}_2)}\right] \subset (0, 1], & \text{if } \mathrm{abs}\left(\mathrm{R}_1\right) < \mathrm{abs}\left(\mathrm{R}_2\right) \end{cases} \end{aligned} \tag{48}$$

Then, we can get

$$h_{il} \in \left[\frac{1}{1+e}, \frac{1}{1+e^{-1}}\right] = [0.2689, 0.7311]\,. \tag{49}$$

This indicates that the normalization in Eq. (44) makes it possible for *c*GPRELM to effectively handle classification and regression tasks for any random initialization. Table 1 summarizes the main differences between ELM, *k*GPRELM, and *c*GPRELM when predicting the label for an unseen instance $\hat{\mathrm{x}} = \left(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_D\right)$.

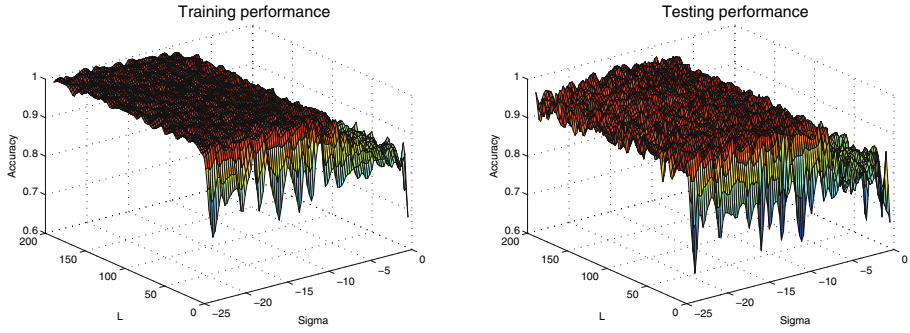
We briefly discuss the time complexity of *c*GPRELM. It was shown [14] that the time complexities of standard ELM are $O\left(NDL\right)$ and $O\left(L^3 + L^2N + LNM\right)$ for calculating the hidden-layer output matrix and output-layer weight matrix, where N is the number of instances, D is the number of instance condition attributes, M is the number of instance decision attributes, and L is the number of hidden-layer nodes. Because of the introduction of the correlation matrix as shown in Eq. (31), the time complexity for calculating $\mathrm{C}\left(\mathrm{H}, \mathrm{H}\right) + \sigma_N^2 \mathrm{I}$ is $O\left(NDL + N^2\right)$ for *c*GPRELM. Then, the time complexity of calculating $\left[\mathrm{C}\left(\mathrm{H}, \mathrm{H}\right) + \sigma_N^2 \mathrm{I}\right]^{-1} \mathrm{Y}$ is $O\left(N^3 + N^2 M\right)$.

4 Experimental simulations

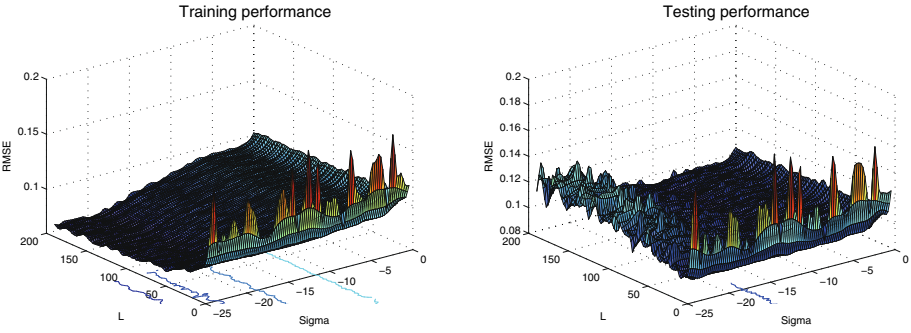
This section presents the results of a series of experiments conducted to validate the predictive performance of the proposed *c*GPRELM method. These experiments include (1) evaluating the training/testing accuracy/RMSE of *c*GPRELM on the Iris and DEE data sets; (2) comparing the confidence intervals of *k*GPRELM and *c*GPRELM on the *SinC* data set;

Table 1 Main differences between ELM, *k*GPRELM, and *c*GPRELM

Algorithm	Predicted output	Hidden-layer transformation	Hidden-layer normalization
ELM	$\hat{y} = h(\hat{x})\, H^{\dagger} Y$	No	No
*k*GPRELM	$\hat{y} = k(h(\hat{x}), H)\left[K(H,H) + \sigma_N^2 I\right]^{-1} Y$	RBF kernel in Eq. (12)	No
*c*GPRELM	$\hat{y} = c(h(\hat{x}), H)\left[C(H,H) + \sigma_N^2 I\right]^{-1} Y$	Correlation in Eq. (29)	Yes



(a) On *Iris* data set



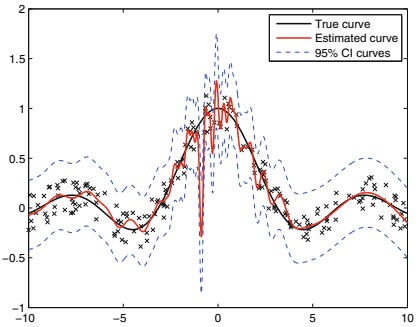
(b) On *DEE* data set

Fig. 4 Training and testing performances of *c*GPRELM on the *Iris* and *DEE* data sets

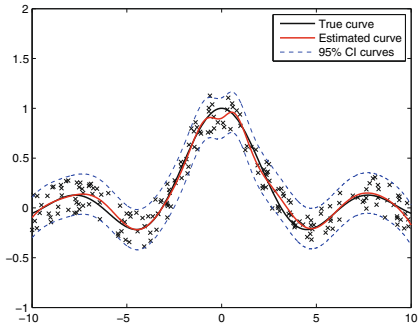
(3) comparing the generalization capabilities of ELM, *k*GPRELM, *c*GPRELM, and multilayer ELM on 19 UCI [18] and 10 KEEL [1] data sets; and (4) comparing the ranks of $K(H,H) + \sigma_N^2 I$ and $C(H,H) + \sigma_N^2 I$. All the experiments were implemented using MATLAB and ran on a Thinkpad SL410K computer with Windows XP Professional, a Pentium Dual-core T4400 2.20 GHz processor and 4 GB of RAM.

4.1 On the *Iris* and *DEE* data sets

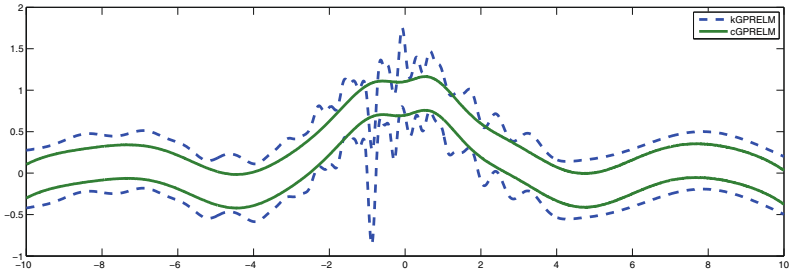
Figures 1 and 2 of Sect. 2.3 illustrate the instability of ELM and overfitting of *k*GPRELM, respectively. In this experiment, we also plot the learning curves of *c*GPRELM for the *Iris*



(a) kGPRELM



(b) cGPRELM



(c) CIs of kGPRELM and cGPRELM

Fig. 5 Predictive performances of kGPRELM ($(L, \sigma_N, \lambda^2) = (190, 2^{-20}, 2^{-9})$) and cGPRELM ($(L, \sigma_N) = (80, 2^{-20})$) on 200 *SinC* instances

and *DEE* data sets. The experimental parameters were setup as follows: the number of hidden-layer nodes in SLFNs is $L = \{10, 20, \ldots, 190, 200\}$; σ_N in Eqs. (5) and (12) is $\sigma_N = \{2^{-24}, 2^{-23.5}, \ldots, 2^{-0.5}, 2^0\}$; and λ^2 in RBF kernel is $\lambda^2 = \{2^{-9}, 2^{-8.5}, \ldots, 2^{9.5}, 2^{10}\}$. Figure 4 presents the training/testing accuracies of cGPRELM on *Iris* and training/testing RMSEs on *DEE*. In that figure, we can clearly see that the learning curves of cGPRELM do not sharply fluctuate and that the testing accuracies and RMSEs of cGPRELM do not obviously decrease and increase with an increase of the training accuracies and decrease of training RMSEs. This is because the proposed cGPRELM method is more stable than ELM and more effectively control overfitting than kGPRELM.

4.2 On the *SinC* data set

The second experiment compares the 95% confidence intervals (CIs) $[\mu - 1.96\sigma, \mu + 1.96\sigma]$ of kGPRELM and cGPRELM, where μ and σ^2 are the mean and variance of the predictive output. In total, 200 instances (x_i, y_i), $i = 1, 2, \ldots, 200$ were randomly generated according to the following *SinC* function

$$y_i = \begin{cases} \frac{\sin x_i}{x_i} + \varepsilon_i \ , & x_i \neq 0 \\ 1 + \varepsilon_i \ , & x_i = 0 \end{cases}, \tag{50}$$

Table 2 Details of 19 UCI [18] classification data sets

No.	Data sets	Attributes	Classes	Instances
1	Auto Mpg	5	3	392
2	Blood transfusion	4	2	748
3	Breast cancer W-P	33	2	198
4	Cleveland	13	5	297
5	Credit approval	15	2	690
6	Cylinder bands	20	2	540
7	Ecoli	5	8	336
8	Glass identification	9	7	214
9	Haberman's survival	3	2	306
10	Image segment	19	7	2310
11	Ionosphere	33	2	351
12	Iris	4	3	150
13	Magic telescope	10	2	19020(5%)
14	New thyroid gland	5	3	215
15	Page blocks	10	5	5473
16	Pima Indian diabetes	8	2	768
17	Sonar	60	2	208
18	Vehicle silhouettes	18	4	846
19	Vowel recognition	10	11	528

Table 3 Details of 10 KEEL [1] regression data sets

No.	Data sets	Attributes	Instances
1	Daily electricity energy	7	365
2	Delta ailerons	6	7129
3	Electrical maintenance	5	1056
4	Friedman function	6	1200
5	Laser generated	5	993
6	Mortgage	16	1049
7	Stock prices	10	950
8	Treasury	16	1049
9	Weather Ankara	10	321
10	Weather Izmir	10	1461

where x_i and ε_i are random numbers drawn uniformly in intervals $[-10, 10]$ and $[-0.2, 0.2]$, respectively. The predictive curves and corresponding CIs of kGPRELM and cGPRELM are presented in Fig. 5. We selected the best learning parameters corresponding to the lowest training RMSEs for kGPRELM (i.e., $\left(190, 2^{-20}, 2^{-9}\right)$) and cGPRELM (i.e., $\left(80, 2^{-20}\right)$) from parameter spaces $L \times \sigma_N \times \lambda^2$ and $L \times \sigma_N$, respectively, where $L = \{10, 20, \ldots, 190, 200\}$, $\sigma_N = \left\{2^{-25}, 2^{-20}, \ldots, 2^{15}, 2^{20}\right\}$, and $\lambda^2 = \left\{2^{-9}, 2^{-8}, \ldots, 2^{9}, 2^{10}\right\}$. In Fig. 5, we can find that kGPRELM overfits, while the proposed cGPRELM does not suffer from serious overfitting. Figure 5c compares different CIs of kGPRELM and cGPRELM. We can find that the CIs of cGPRELM are almost all contained in those of kGPRELM. This indicates that cGPRELM obtains smaller CIs than kGPRELM. From a statistical perspective, the smaller a

CI is, the more accurate the estimation is. This indicates that the proposed *c*GPRELM method can obtain a more accurate predictive output than *k*GPRELM. The next experiments on real data sets also support this empirical conclusion.

4.3 On UCI and KEEL data sets

In another experiment, we selected 19 UCI classification data sets (Table 2) and 10 KEEL regression data sets (Table 3) to compare the performances of ELM, *k*GPRELM, *c*GPRELM, and multilayer ELM (ML-ELM) [15]. We measured the training accuracy/RMSE, testing accuracy/RMSE, training time, and testing time of these three methods. The inputs of all 29 data sets and outputs of 10 regression data sets are normalized in the [0, 1] interval. The learning parameters for ELM, *k*GPRELM, and *c*GPRELM are $L = \{10, 20, \ldots, 190, 200\}$, $\sigma_N = \left\{2^{-25}, 2^{-20}, \ldots, 2^{15}, 2^{20}\right\}$, and $\lambda^2 = \left\{2^{-9}, 2^{-8}, \ldots, 2^{9}, 2^{10}\right\}$. To find the best training models, ELM, *k*GPRELM, and *c*GPRELM were, respectively, trained 20, $20\times10\times20$, and 20×10 times for each data set. For each L, $\left(L, \sigma_N, \lambda^2\right)$, and (L, σ_N), we used 10 times twofold cross-validation to train and test ELM, *k*GPRELM, and *c*GPRELM. For ML-ELM with three hidden-layers, a fixed learning parameter is used to train and test the algorithm based on the same training and testing data sets as ELM, *k*GPRELM, and *c*GPRELM. Accuracies and RMSEs are listed in Tables 4, 5. Learning time on classification and regression data sets is summarized in Tables 6, 7.

The experimental results in Tables 4, 5 support the theoretical analysis of *k*GPRELM's overfitting presented in Sect. 2.3. Its maximal training accuracies on 18 UCI classification data sets are 1.000, and minimal training RMSEs on 9 KEEL regression data sets are 0.000. However, the corresponding testing accuracies and RMSEs of *k*GPRELM are the worst. Smaller λ^2s (e.g., 2^{-9}) lead to overfitting for classification tasks and larger Ls (e.g., 160, 170, 180, 190, and 200) for regression tasks. Though the proposed *c*GPRELM method does not achieve the highest training accuracies or the lowest training RMSEs, its testing accuracies and RMSEs are the best in comparison with ELM and *k*GPRELM. *c*GPRELM, in particular, achieves comparable testing results with ML-ELM: *c*GPRELM obtains the highest testing accuracies on 14 classification data sets and lower testing RMSEs on 7 regression data sets. It indicates that correlation GPR can help ELM in achieving good generalization performances under the condition of shallow learning. Tables 6, 7 compare the training and testing times of ELM, *k*GPRELM, *c*GPRELM, and ML-ELM. It is found that *c*GPRELM requires less training time than *k*GPRELM. This is because calculating K (H, H) is more time-consuming than C (H, H). This observation is also supported by the computational time comparison for kernel and correlation matrices in Fig. 3.

4.4 Ranks of K (H, H) $+ \sigma_N^2$I and C (H, H) $+ \sigma_N^2$I

We use two classification (*Iris* and *New Thyroid Gland*) and two regression (*DEE* and *Wankara*) data sets to validate the effects of ranks of K (H, H) $+ \sigma_N^2$I and C (H, H) $+ \sigma_N^2$I on the predictive performances of *k*GPRELM and *c*GPRELM, respectively. For *k*GPRELM, we set $\sigma_N = 2^{-24}$, $L = \{10, 20, \ldots, 190, 200\}$, and $\lambda^2 = \left\{2^{-9}, 2^{-8.5}, \ldots, 2^{9.5}, 2^{10}\right\}$. For *c*GPRELM, we let $L = \{10, 20, \cdots, 190, 200\}$, and $\sigma_N = \left\{2^{-24}, 2^{-23.5}, \ldots, 2^{-0.5}, 2^{0}\right\}$. Figures 6 and 7 show the relationships between ranks and predictive performances on classification and regression data sets, respectively. In the RBF kernel of *k*GPRELM, there is a user-specified parameter, i.e., λ^2. From Figs. 6a, c, 7a, and c, we can find that smaller λ^2s lead to full ranks of K (H, H) $+ \sigma_N^2$I and thus higher training accuracies (e.g., 1.000) and lower

Table 4 Maximal training accuracies of ELM, *k*GPRELM, *c*GPRELM, and ML-ELM and corresponding testing accuracies

No.	ELM			*k*GPRELM			*c*GPRELM			MLELM		
	Training	L	Testing	Training	(L, σ_N, λ^2)	Testing	Training	(L, σ_N)	Testing	Training	(L_1, L_2, L_3)	Testing
1	0.923	160	0.773	1.000	$(10, 2^{-25}, 2^{-9})$	0.648	0.893	$(120, 2^{-20})$	0.809	0.696	(50, 50, 100)	0.687
2	0.821	40	0.761	0.956	$(110, 2^{-15}, 2^{-7})$	0.648	0.816	$(90, 2^{-25})$	0.786	0.767	(50, 50, 100)	0.765
3	1.000	90	0.535	1.000	$(10, 2^{-25}, 2^{-9})$	0.551	1.000	$(80, 2^{-25})$	0.697	0.972	(50, 50, 100)	0.970
4	0.892	140	0.424	1.000	$(10, 2^{-25}, 2^{-9})$	0.387	0.832	$(190, 2^{-25})$	0.512	0.539	(50, 50, 100)	0.539
5	0.862	160	0.714	1.000	$(10, 2^{-25}, 2^{-9})$	0.597	0.829	$(200, 2^{-20})$	0.759	0.626	(50, 50, 100)	0.620
6	0.980	200	0.628	1.000	$(10, 2^{-25}, 2^{-9})$	0.539	0.987	$(200, 2^{-25})$	0.639	0.638	(50, 50, 100)	0.561
7	0.955	140	0.774	1.000	$(10, 2^{-25}, 2^{-9})$	0.598	0.940	$(200, 2^{-25})$	0.807	0.661	(50, 50, 100)	0.645
8	1.000	150	0.514	1.000	$(10, 2^{-25}, 2^{-9})$	0.481	0.967	$(130, 2^{-20})$	0.603	0.448	(50, 50, 100)	0.402
9	0.850	170	0.690	1.000	$(120, 2^{-20}, 2^{-7})$	0.650	0.837	$(180, 2^{-25})$	0.735	0.735	(50, 50, 100)	0.735
10	0.982	190	0.948	1.000	$(10, 2^{-25}, 2^{-9})$	0.867	0.959	$(180, 2^{-20})$	0.948	0.929	(50, 50, 100)	0.921
11	0.954	150	0.766	1.000	$(70, 2^{-25}, 2^{-9})$	0.727	1.000	$(140, 2^{-25})$	0.783	0.895	(50, 50, 100)	0.854
12	1.000	20	0.927	1.000	$(10, 2^{-25}, 2^{-9})$	0.940	1.000	$(30, 2^{-20})$	0.947	0.829	(50, 50, 100)	0.803
13	0.919	180	0.803	1.000	$(10, 2^{-25}, 2^{-9})$	0.668	0.885	$(190, 2^{-25})$	0.833	0.763	(50, 50, 100)	0.757
14	1.000	40	0.902	1.000	$(10, 2^{-25}, 2^{-9})$	0.860	1.000	$(70, 2^{-20})$	0.925	0.711	(50, 50, 100)	0.707
15	0.989	190	0.899	1.000	$(10, 2^{-25}, 2^{-9})$	0.775	0.976	$(100, 2^{-20})$	0.925	0.886	(50, 50, 100)	0.886
16	0.910	200	0.693	1.000	$(10, 2^{-25}, 2^{-9})$	0.635	0.857	$(170, 2^{-20})$	0.740	0.742	(50, 50, 100)	0.730
17	0.966	200	0.644	1.000	$(60, 2^{-25}, 2^{-9})$	0.587	1.000	$(70, 2^{-25})$	0.716	0.831	(50, 50, 100)	0.742
18	0.943	190	0.741	1.000	$(10, 2^{-25}, 2^{-9})$	0.593	0.885	$(190, 2^{-20})$	0.757	0.551	(50, 50, 100)	0.517
19	1.000	130	0.792	1.000	$(10, 2^{-25}, 2^{-9})$	0.843	0.994	$(180, 2^{-25})$	0.848	0.378	(50, 50, 100)	0.309

Table 5 Minimal training RMSEs of ELM, *k*GPRELM, *c*GPRELM, and ML-ELM and corresponding testing RMSEs

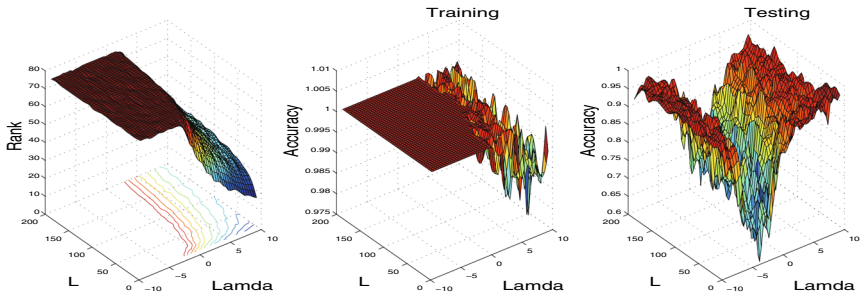
No.	ELM			*k*GPRELM			*c*GPRELM			MLELM		
	Training	L	Testing	Training	(L, σ_N, λ^2)	Testing	Training	(L, σ_N)	Testing	Training	(L_1, L_2, L_3)	Testing
1	0.053	170	0.159	0.000	$(190, 2^{-25}, 2^{-9})$	0.371	0.064	$(150, 2^{-25})$	0.113	0.102	(50, 50, 100)	0.104
2	0.042	120	0.174	0.000	$(150, 2^{-25}, 2^{-9})$	0.354	0.053	$(90, 2^{-20})$	0.109	0.039	(50, 50, 100)	0.039
3	0.009	170	0.046	0.001	$(140, 2^{-15}, 2^{-6})$	0.121	0.010	$(120, 2^{-20})$	0.020	0.051	(50, 50, 100)	0.052
4	0.020	190	0.074	0.000	$(130, 2^{-25}, 2^{-7})$	0.286	0.038	$(160, 2^{-20})$	0.045	0.055	(50, 50, 100)	0.058
5	0.007	100	0.052	0.000	$(180, 2^{-25}, 2^{-9})$	0.201	0.018	$(180, 2^{-25})$	0.040	0.089	(50, 50, 100)	0.090
6	0.001	190	0.038	0.000	$(190, 2^{-25}, 2^{-9})$	0.125	0.002	$(200, 2^{-20})$	0.011	0.014	(50, 50, 100)	0.015
7	0.013	190	0.072	0.000	$(170, 2^{-25}, 2^{-9})$	0.213	0.029	$(180, 2^{-20})$	0.039	0.065	(50, 50, 100)	0.067
8	0.003	200	0.032	0.000	$(160, 2^{-25}, 2^{-9})$	0.119	0.009	$(200, 2^{-20})$	0.012	0.019	(50, 50, 100)	0.019
9	0.007	200	0.151	0.000	$(200, 2^{-25}, 2^{-9})$	0.325	0.012	$(120, 2^{-20})$	0.039	0.065	(50, 50, 100)	0.067
10	0.015	200	0.133	0.000	$(140, 2^{-25}, 2^{-9})$	0.214	0.013	$(130, 2^{-25})$	0.043	0.040	(50, 50, 100)	0.041

Table 6 Training and testing times of ELM, *k*GPRELM, *c*GPRELM, and ML-ELM on 19 classification data sets

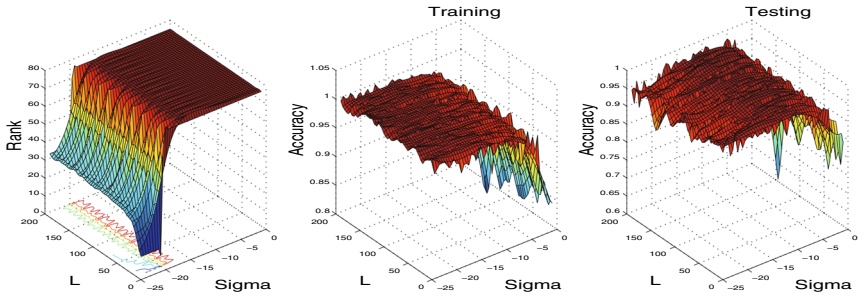
No.	ELM		*k*GPRELM		*c*GPRELM		MLELM	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
1	0.031	0.000	0.148	0.156	0.000	0.039	0.003	0.000
2	0.000	0.000	0.484	0.344	0.102	0.031	0.004	0.000
3	0.000	0.000	0.023	0.031	0.000	0.016	0.004	0.000
4	0.016	0.000	0.063	0.078	0.000	0.031	0.003	0.000
5	0.016	0.047	0.367	0.289	0.063	0.047	0.004	0.000
6	0.047	0.000	0.227	0.180	0.070	0.039	0.004	0.000
7	0.023	0.000	0.094	0.055	0.000	0.039	0.003	0.000
8	0.016	0.000	0.047	0.031	0.000	0.016	0.003	0.000
9	0.016	0.000	0.102	0.094	0.000	0.016	0.003	0.000
10	0.070	0.023	6.273	2.750	2.453	0.391	0.007	0.000
11	0.031	0.000	0.125	0.109	0.016	0.000	0.004	0.000
12	0.000	0.000	0.016	0.016	0.000	0.000	0.003	0.000
13	0.031	0.000	0.641	0.383	0.164	0.070	0.004	0.000
14	0.000	0.000	0.047	0.031	0.016	0.000	0.003	0.000
15	0.063	0.039	2.250	2.141	1.247	0.243	0.004	0.000
16	0.047	0.000	0.453	0.367	0.117	0.055	0.004	0.000
17	0.016	0.000	0.047	0.047	0.016	0.000	0.005	0.000
18	0.031	0.039	0.438	0.344	0.109	0.055	0.004	0.000
19	0.000	0.000	0.172	0.234	0.047	0.039	0.004	0.000

Table 7 Training and testing times of ELM, *k*GPRELM, *c*GPRELM, and ML-ELM on 10 regression data sets

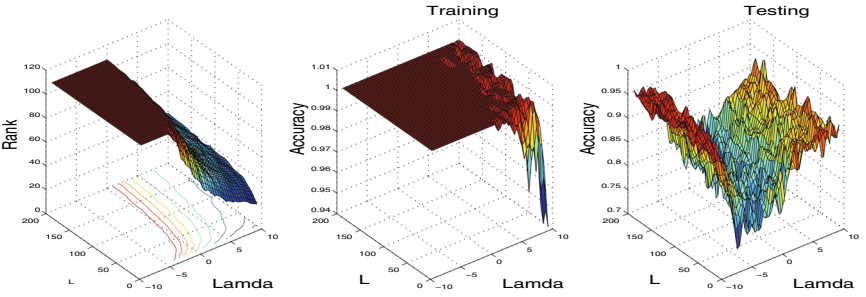
No.	ELM		*k*GPRELM		*c*GPRELM		MLELM	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
1	0.016	0.000	0.164	0.203	0.031	0.000	0.003	0.000
2	0.086	0.055	2.969	3.125	0.313	0.000	0.018	0.003
3	0.039	0.000	0.417	0.365	0.211	0.070	0.004	0.000
4	0.130	0.000	0.469	0.495	0.305	0.109	0.005	0.000
5	0.023	0.000	0.488	0.508	0.195	0.086	0.004	0.000
6	0.031	0.023	0.469	0.313	0.242	0.078	0.004	0.000
7	0.031	0.000	0.508	0.430	0.172	0.078	0.004	0.000
8	0.070	0.000	0.469	0.417	0.227	0.117	0.005	0.000
9	0.031	0.000	0.117	0.156	0.031	0.000	0.003	0.000
10	0.047	0.016	0.688	0.594	0.531	0.141	0.005	0.000



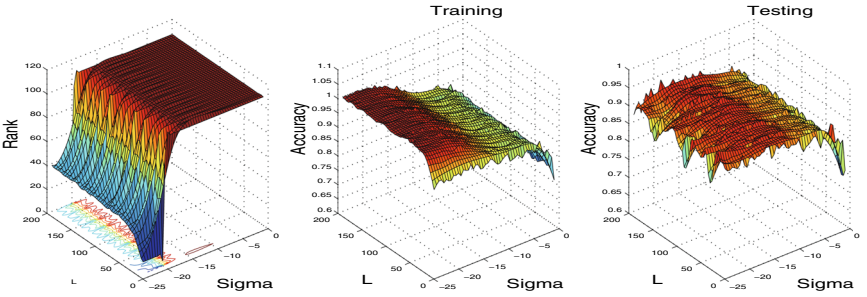
(a) kGPRELM on *Iris* data set



(b) cGPRELM on *Iris* data set

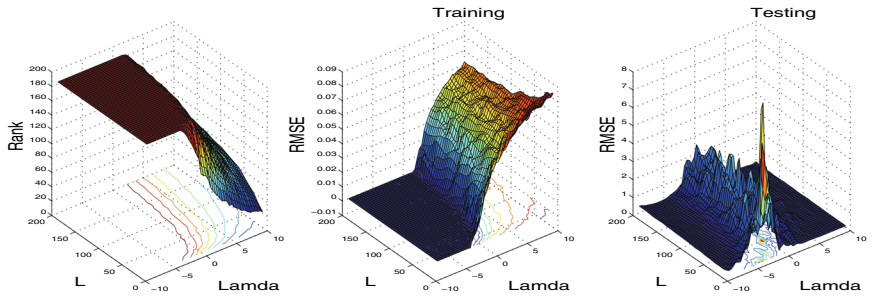


(c) kGPRELM on *New Thyroid Gland* data set

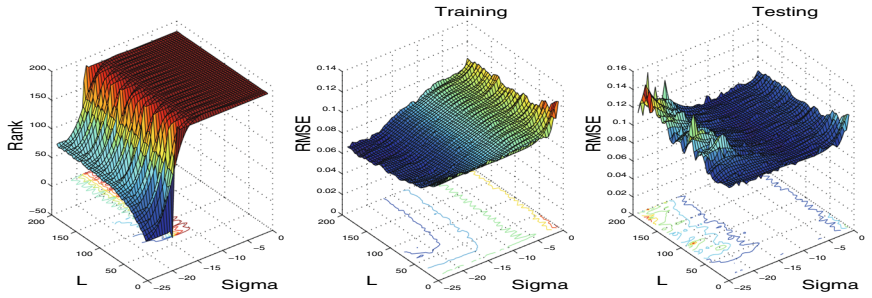


(d) cGPRELM on *New Thyroid Gland* data set

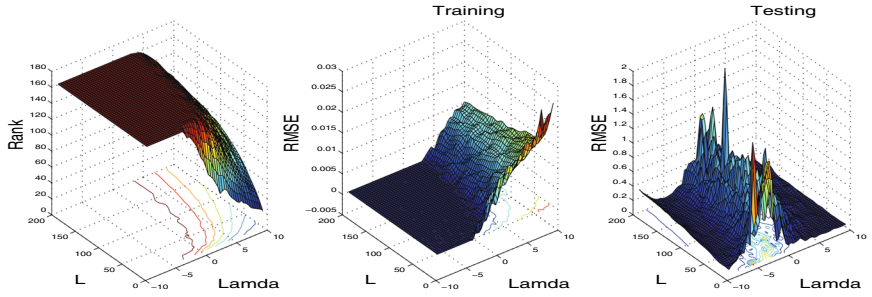
Fig. 6 Ranks of K (H, H) $+ \sigma_N^2$I and C (H, H) $+ \sigma_N^2$I on two representative classification data sets



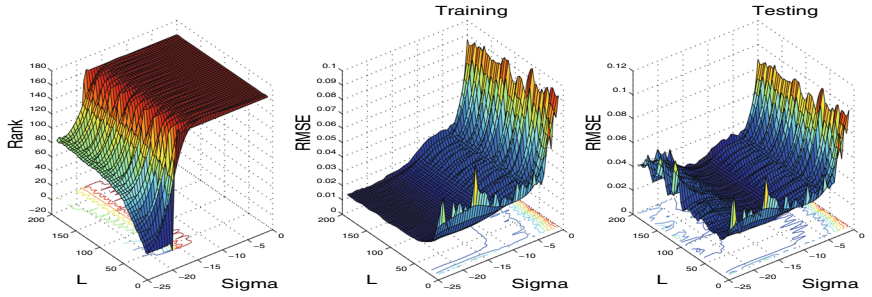
(a) kGPRELM on *DEE* data set



(b) cGPRELM on *DEE* data set



(c) kGPRELM on *Wankara* data set



(d) cGPRELM on *Wankara* data set

Fig. 7 Ranks of K (H, H) $+ \sigma_N^2$I and C (H, H) $+ \sigma_N^2$I on two representative regression data sets

training RMSEs (e.g., 0.000). However, as λ^2 increases, the testing accuracies for *k*GPRELM decrease initially and increase afterward (the testing RMSEs increase initially and decrease afterward). This indicates that *k*GPRELM easily overfits and λ^2 has a clear influence on the testing accuracy and RMSE of *k*GPRELM. As the designed *c*GPRELM method has no pre-determined parameters in Eq. (30), *c*GPRELM is more simple to use than *k*GPRELM. It is observed in Figs. 6b, d, 7b, and d that the rank of C (H, H) $+ \sigma_N^2$I mainly depends on σ_N. Usually, smaller σ_Ns leads to better training and testing performances for *c*GPRELM even though C (H, H) $+ \sigma_N^2$I is not full rank.

5 Conclusion

This paper investigated the primary cause of overfitting in kernel-based Gaussian process regression for extreme learning machine (*k*GPRELM) and proposed a correlation-based GPRELM (*c*GPRELM) method. It provides stable prediction and effectively control overfitting by using a correlation coefficient rather than a kernel function to measure the similarity between different hidden-layer outputs of the single hidden-layer feed-forward neural network (SLFN) with random input-layer weights and hidden-layer biases. The experimental results demonstrated the feasibility and effectiveness of the proposed *c*GPRELM method, which not only outperforms *k*GPRELM in terms of generalization performance but also has a lower computational complexity. In future work, we will (1) provide a theoretical proof for the overfitting of the Gaussian process-based random weight network; (2) investigate the robustness of *c*GPRELM for noisy data and outliers; and (3) use *c*GPRELM to handle prediction tasks with fuzzy-in and fuzzy-out data.

Acknowledgements The authors would like to thank the editors and three anonymous reviewers whose meticulous readings and valuable suggestions have helped to improve the paper significantly after two rounds of review. This paper was supported by National Natural Science Foundation of China (61972261), Natural Science Foundation of Guangdong Province (2314050006683), Key Basic Research Foundation of Shenzhen (JCYJ20220818100205012), and Basic Research Foundations of Shenzhen (JCYJ20210324093609026, JCYJ20200813091134001).

Author Contributions Data Curation, Writing-Review and Editing, XY; Methodology, Writing-Original Draft Preparation, Writing-Review and Editing, YH; Formal Analysis, Writing-Review and Editing, MZ; Investigation, Writing-Review and Editing, PF-V; Supervision, Funding acquisition, JZH.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

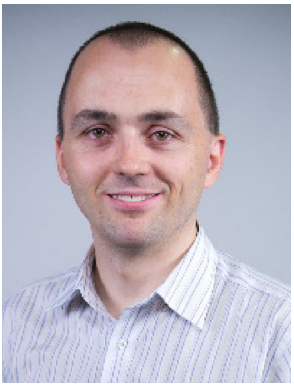
1. Alcalá-Fdez J, Fernandez A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011) KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Mult-Valued Logic Soft Comput 17(2–3):255–287
2. Cao JW, Lin ZP, Huang GB (2012) Self-adaptive evolutionary extreme learning machine. Neural Process Lett 36(3):285–305
3. Chatzis SP, Korkinof D, Demiris Y (2011) The one-hidden layer non-parametric Bayesian kernel machine, In: Proceedings of IEEE international conference on tools with artificial intelligence, pp 825–831
4. Fernández-Delgado M, Cernadas E, Barro S, Amorim D (2014) Do we need hundreds of classifiers to solve real world classification problems? J Mach Learn Res 15(1):3133–3181
5. Fu AM, Dong CR, Wang LS (2015) An experimental study on stability and generalization of extreme learning machines. Int J Mach Learn Cybern 6:1

6. Fu AM, Wang XZ, He YL, Wang LS (2014) A study on residence error of training an extreme learning machine and its application to evolutionary algorithms. Neurocomputing 146:75–82
7. Han F, Yao HF, Ling QH (2013) An improved evolutionary extreme learning machine based on particle swarm optimization. Neurocomputing 116:87–93
8. Hu J, Heidari AA, Shou Y, Ye H, Wang L, Huang X, Wu P (2022) Detection of COVID-19 severity using blood gas analysis parameters and Harris hawks optimized extreme learning machine. Comput Biol Med 142:105166
9. Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892
10. Huang GB, Wang DH, Lan Y (2011) Extreme learning machines: a survey. Int J Mach Learn Cybern 2(2):107–122
11. Huang GB, Zhou HM, Ding XJ, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern B Cybern 42(2):513–529
12. Huang GB, Zhu QY, Siew CK (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. Proc Int Joint Conf Neural Netw 2:985–990
13. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1):489–501
14. Iosifidis A, Tefas A, Pitas I (2015) On the kernel extreme learning machine classifier. Pattern Recogn Lett 54:11–17
15. Kasun LLC, Zhou H, Huang GB et al (2013) Representational learning with ELMs for big data. IEEE Intell Syst 28(6):31–34
16. Lan Y, Soh YC, Huang GB (2009) Ensemble of online sequential extreme learning machine. Neurocomputing 72(13–15):3391–3395
17. Larrea M, Porto A, Irigoyen E et al (2021) Extreme learning machine ensemble model for time series forecasting boosted by PSO: application to an electric consumption problem. Neurocomputing 452:465–472
18. Lichman M (2013) UCI machine learning repository. University of California, School of Information and Computer Science, Irvine, CA
19. Liu N, Wang H (2010) Ensemble based extreme learning machine. IEEE Signal Process Lett 17(8):754–757
20. Lukasik M, Bontcheva K, Cohn T et al (2019) Gaussian processes for rumour stance classification in social media. ACM Trans Inf Syst 37(2):1–24
21. Luo JH, Vong CM, Wong PK (2014) Sparse Bayesian extreme learning machine for multi-classification. IEEE Trans Neural Netw Learn Syst 25(4):836–843
22. Mair S, Brefeld U (2018) Distributed robust Gaussian process regression. Knowl Inf Syst 55(2):415–435
23. Nguyen V, Gupta S, Rana S, Li C, Venkatesh S (2019) Filtering Bayesian optimization approach in weakly specified search space. Knowl Inf Syst 60(1):385–413
24. Peng X, Song R, Cao Q, Li Y, Cui D, Jia X, Lin Z, Huang GB (2022) Real-time illegal parking detection algorithm in urban environments. IEEE Trans Intel Transp Syst. https://doi.org/10.1109/TITS.2022.3180225
25. Rasmussen CE, Williams CKI (2006) Gaussian processes for machine learning. The MIT Press
26. Silva DNG, Pacifico LDS, Ludermir TB (2011) An evolutionary extreme learning machine based on group search optimization, In: Proceedings of the IEEE congress on evolutionary computation, pp 574–580
27. Song G, Dai Q, Han X, Guo L (2020) Two novel ELM-based stacking deep models focused on image recognition. Appl Intel 50(5):1345–1366
28. Soria-Olivas E, Gomez-Sanchis J, Jarman IH, Vila-Frances J (2011) BELM: Bayesian extreme learning machine. IEEE Trans Neural Netw 22(3):505–509
29. Wang YG, Cao FL, Yuan YB (2011) A study on effectiveness of extreme learning machine. Neurocomputing 74(16):2483–2490
30. Wilson AG, Adams RP (2013) Gaussian process kernels for pattern discovery and extrapolation. Proc Int Conf Mach Learn 3:1067–1075
31. Xue J, Zhou S, Liu Q, Liu X, Yin J (2018) Financial time series prediction using $l2$, 1RF-ELM. Neurocomputing 277:176–186
32. Xue XW, Yao M, Wu ZH, Yang JH (2014) Genetic ensemble of extreme learning machine. Neurocomputing 129:175–184
33. Zhai JH, Xu HY, Wang XZ (2012) Dynamic ensemble extreme learning machine based on sample entropy. Soft Comput 16(9):1493–1502
34. Zhu QY, Qin AK, Suganthan PN, Huang GB (2005) Evolutionary extreme learning machine. Pattern Recogn 38(10):1759–1763

Xuan Ye received his B.Sc. and M.Sc. degrees from Shenzhen University, Shenzhen, China, in 2019 and 2022, respectively. He is currently a software engineer in Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen, China. His current research interests are in probability density estimation technology, nonlinear optimization method, and big data system computing technology.

Yulin He was born in 1982. He received the Ph.D. degree from Hebei University, China, in 2014. From 2011 to 2014, he has served as a Research Assistant with the Department of Computing, Hong Kong Polytechnic University, Hong Kong, China. From 2014 to 2017, he worked as a Post-doctoral Fellow in the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. He is currently a Research Associate with Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen, China. His main research interests include big data approximate computing technologies, multi-sample statistical analysis theories and methods, and data mining/machine learning algorithms and their applications. He has published over 100+ research papers in ACM Transactions, CAAI Transactions, IEEE Transactions, Elsevier, Springer Journals and PAKDD, IJCNN, CEC, DASFAA conferences. Dr. He is an ACM member, CAAI member, CCF member, IEEE member, and the Editorial Review Board members of several international journals.

Manjing Zhang received her Ph.D. degree in management sciences from City University of Hong Kong, Hong Kong. She is currently an associate research fellow in Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen, China. Her research interests mainly focus on stochastic simulation, data mining, and machine learning.



Philippe Fournier-Viger was born in 1980. He is a distinguished professor with the College of Computer Science & Software Engineering at the Shenzhen University (China). He obtained a title of national talent from the National Science Foundation of China. He has published more than 300 research papers related to data mining, big data, intelligent systems, and applications, which have received more than 8000 citations (H-Index 45). He is the associate editor-in-chief of the Applied Intelligence journal (SCI, Q1) and editor-in-chief of the Data Science and Pattern Recognition journal. He is the founder of the SPMF data mining library, offering more than 200 algorithms, used in more than 1,000 research papers. He is co-founder of the UDML and MLiSE series workshop held at the ICDM, PKDD, and KDD conferences. His interests are data mining, algorithm design, pattern mining, sequence mining, big data, and applications.

Joshua Zhexue Huang was born in 1959. He received the Ph.D. degree from The Royal Institute of Technology, Stockholm, Sweden, in 1993. He is currently a Distinguished Professor with the College of Computer Science & Software Engineering, Shenzhen University, Shenzhen, China. He is also the Director of Big Data Institute, Shenzhen, China, and the Deputy Director of National Engineering Laboratory for Big Data System Computing Technology. He has published over 200 research papers in conferences and journals. His main research interests include big data technology and applications. Prof. Huang received the first PAKDD Most Influential Paper Award in 2006. He is known for his contributions to the development of a series of k-means type clustering algorithms in data mining, such as k-modes, fuzzy k-modes, k-prototypes, and w-k-means that are widely cited and used, and some of which have been included in commercial software. He has extensive industry expertise in business intelligence and data mining and has been involved in numerous consulting projects in many countries.