Comparative Analysis of Gaussian Process Regression Based Extreme Learning Machine

Jing Zhou[1*], Rui Ying Liu[1], Xu Zhou[2], Rana Aamir Raza Ashfaq[3*]

[1] College of Science, Agricultural University of Hebei, Baoding 071001, China.
[2] Department of Basic Courses, Agricultural University of Hebei, Huanghua 061100, China.
[3] Department of Computer Science, Bahauddin Zakariya University, Multan, Pakistan.

* Corresponding authors Emails: csjzhou@126.com, aamir@bzu.edu.pk
Manuscript submitted August 7, 2016; accepted August 31, 2017.
doi: 10.17706/jsw.12.4.292-302

Abstract: It is an effective way to overcome the randomization sensibility of extreme learning machine (ELM) by using Gaussian process regression (GPR) to optimize the output-layer weights. The key of GPR based ELM (GPRELM) is the selection of kernel function which is used to measure the similarity between different hidden-layer output vectors. In this paper, we conduct an experimental analysis to compare the classification performances of radial basis function(RBF) kernel and polynomial (Poly) kernel based GPRELMs. The comparative results on 24 UCI data sets reveal that: (1) GPRELMs have the serious over-fitting; (2) GPRELMs can get the better classification accuracies with less hidden-layer nodes in comparison with the original ELM; and (3) the smaller regularization factors usually bring about the higher training accuracies for GPRELMs, while the larger regularization factors usually result in the higher testing accuracies. All these conclusions provide the useful enlightenments and instructions for the theoretical studies and practical applications of GPRELMs.

Key words: Extreme learning machine, gaussian process regression; radial basis function kernel; polynomial Kernel.

1. Introduction

Extreme learning machine (ELM) [8]-[10] is a simple training algorithm for single hidden-layer feed-forward neural network (SLFN), which randomly selects the input-layer weights and hidden-layer biases and analytically determines the output-layer weights. Thus, the training speed of ELM can be thousands of times faster than traditional back-propagation (BP) algorithm. Meanwhile, the theoretically proof guarantees the universal approximate capability of ELM. The lower computational complexity and better generalization performance makes ELM obtain a wide range of applications [2], [6], [7], [14], [15], [17].

However, every coin has two sides. ELM also has some obvious defects one of which is the sensibility of prediction result to random initialization. The researchers have conducted some representative works along this direction. For example, Ref. [16] proposed an evolutionary ELM (E-ELM) which uses the differential evolutionary algorithm to select the input weights and hidden biases for ELM. Then, [1] improved E-ELM and developed a self-adaptive evolutionary extreme learning machine (SaE-ELM) to optimize the hidden node parameters. Experimental results show SaE-ELM outperforms E-ELM. An optimized extreme learning machine (O-ELM) was designedin [12], which uses three different optimization algorithms to optimize the input weights, hidden biases, and regularization factor, simultaneously. Ref. [5] proposed two weight initialization schemes, i.e., binary ELM based on {0,1}-weights and ternary ELM based on {-1, 0, 1}-weights, to improve the diversity of neurons in the hidden layer. For binary/ternary ELMs, the necessary optimizations are also required to select the better

parameters.

These improvements indeed make ELM more stable, but they require high computational complexities because of the optimizations to input-layer weights and hidden-layer biases. Recently, a kind of optimization to ELM based on Bayesian prior knowledge, i.e., One-Hidden Layer Non-parametric Bayesian Kernel Machine (1HNBKM) [3], was proposed. Instead of the direct point-prediction, 1HNBKM estimates the posterior probability distribution of SLFN output so that the influence of random initialization is weakened. Due to avoid the time-consuming adjustment to random parameters, 1HNBKM saves a large amount of training time. In fact, 1HNBKM uses Gaussian Process Regression (GPR) to yield the prior distribution for output and thus stabilizes the ELM prediction. For the convenience of discussion, we call 1HNBKM as GPRELM in this paper. In [3], the authors investigated the classification error rate (classification task) and root mean square error (regression task) of GPRELM with Radial Basis Function (RBF) kernel. In order to determine the necessary parameters in GPRELM, the scaled conjugate gradient descent algorithm is employed to carry out the optimization task.

In this paper, we conduct a deeply experimental investigation to GPRELM, including its over-fitting characteristic and impacts of different kernels and learning parameters on classification performances (e.g., training accuracy, testing accuracy, training time and testing time) of GPRELM. The experimental results show that (1) the introduction of GPR leads to serious over-fitting for ELM although the randomization sensibility of ELM is weakened to some extent; (2) GPRELM can get the better classification accuracies with less hidden-layer nodes in comparison with original ELM; and (3) the smaller regularization factors usually bring about the higher training accuracies for GPRELM, while the larger regularization factors usually lead to the higher testing accuracies. These conclusions are useful to practical applications of 1HNBKM and can help users to select appropriate kernel and learning parameters for GPR based ELM.

2. Improving Extreme Learning Machine with Gaussian P rocess Regression

2.1. Original ELM

Given the training data set with N distinct instances $\mathrm{D}=\left\{\left(\mathrm{x}_i,\mathrm{y}_i\right)_{i=1}^{N}\middle|\mathrm{x}_i=\left(x_{i1},x_{i2},\cdots,x_{iD}\right),\mathrm{y}_i=\left(y_{i1},y_{i2},\cdots,y_{iM}\right)\right\}$, ELM [9], [10] calculates the output-layer weight matrix as

$$\beta=\mathrm{H}^{\dagger}\mathrm{Y}, \tag{1}$$

where

$$\mathrm{H}^{\dagger}=\begin{bmatrix} g\left(\mathrm{w}_1\mathrm{x}_1+b_1\right) & g\left(\mathrm{w}_2\mathrm{x}_1+b_2\right) & \cdots & g\left(\mathrm{w}_L\mathrm{x}_1+b_L\right) \\ g\left(\mathrm{w}_1\mathrm{x}_2+b_1\right) & g\left(\mathrm{w}_2\mathrm{x}_2+b_2\right) & \cdots & g\left(\mathrm{w}_L\mathrm{x}_2+b_L\right) \\ \vdots & \vdots & \ddots & \vdots \\ g\left(\mathrm{w}_1\mathrm{x}_N+b_1\right) & g\left(\mathrm{w}_2\mathrm{x}_N+b_2\right) & \cdots & g\left(\mathrm{w}_L\mathrm{x}_N+b_L\right) \end{bmatrix}$$

is Moore-Penrose generalized inverse of hidden-layer output matrix, $g\left(v\right)=\dfrac{1}{1+\exp\left(-v\right)}, v\in\left(-\infty,+\infty\right)$ is sigmoid activation function, L is the number of hidden-layer nodes of ELM, the input-layer weight matrix

$$\mathrm{W}=\left[\mathrm{w}_1,\mathrm{w}_2,\cdots,\mathrm{w}_L\right]=\begin{bmatrix} w_{11} & w_{21} & \cdots & w_{D1} \\ w_{12} & w_{22} & \cdots & w_{D2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1L} & w_{2L} & \cdots & w_{DL} \end{bmatrix}$$

and hidden-layer bias vector $b=\left(b_1,b_2,\cdots,b_L\right)$ are randomly selected according to any continuous probability distribution [10], and the training output matrix is

$$\mathrm{Y}=\begin{bmatrix}\mathrm{y}_1\\ \mathrm{y}_2\\ \vdots\\ \mathrm{y}_N\end{bmatrix}=\begin{bmatrix}y_{11} & y_{12} & \cdots & y_{1M}\\ y_{21} & y_{22} & \cdots & y_{2M}\\ \vdots & \vdots & \ddots & \vdots\\ y_{N1} & y_{N2} & \cdots & y_{NM}\end{bmatrix}.$$

For an unseen instance $\bar{\mathrm{x}}=(\bar{x}_1,\bar{x}_2,\cdots,\bar{x}_D)$, ELM predicts its output as:

$$\bar{\mathrm{y}}=\mathrm{h}(\bar{\mathrm{x}})\beta=\mathrm{h}(\bar{\mathrm{x}})\mathrm{H}^{+}\mathrm{Y}, \tag{2}$$

where $\mathrm{h}(\bar{\mathrm{x}})=\left(g(\mathrm{w}_1\bar{\mathrm{x}}+b_1),g(\mathrm{w}_2\bar{\mathrm{x}}+b_2),\cdots,g(\mathrm{w}_L\bar{\mathrm{x}}+b_L)\right)$ is the hidden-layer output vector of $\bar{\mathrm{x}}$. Due to avoid the iterative adjustments to weights and biases of SLFN, ELM's training speed can be thousands of times faster than BP [10]. ELM can achieve the equal generalization performances with Support Vector Machine (SVM) and Least Square SVM (LSSVM) [9]. From Eq. (2), we can find the predictive accuracy of ELM mainly depends on the calculation of H^{+}. Sometimes, the random selections to input-layer weights W and hidden- layer biases b can produce nonsingular hidden-layer output matrix H which causes no solution of linear system $\mathrm{H}\beta=\mathrm{Y}$ and lowers the predictive accuracy of ELM [13]. This makes the prediction of ELM unstable and indicates that ELM is sensitive to random initialization.

2.2. Gprelm

GPRELM[3] is a recently proposed method to improve ELM's random sensitivity, which predicts the output $\bar{\mathrm{y}}$ for unseen instance $\bar{\mathrm{x}}$ according to the following joint Gaussian distribution:

$$\begin{bmatrix}\mathrm{Y}\\ \bar{\mathrm{y}}\end{bmatrix}\sim\mathrm{N}\left(0,\begin{bmatrix}\mathrm{K}(\mathrm{H},\mathrm{H}) & \mathrm{k}^{\mathrm{T}}(\mathrm{h}(\bar{\mathrm{x}}),\mathrm{H})\\ \mathrm{k}(\mathrm{h}(\bar{\mathrm{x}}),\mathrm{H}) & \kappa(\mathrm{h}(\bar{\mathrm{x}}),\mathrm{h}(\bar{\mathrm{x}}))\end{bmatrix}\right), \tag{3}$$

where $\mathrm{h}(\mathrm{x}_i)=\left(g(\mathrm{w}_1\mathrm{x}_i+b_1),g(\mathrm{w}_2\mathrm{x}_i+b_2),\cdots,g(\mathrm{w}_L\mathrm{x}_i+b_L)\right)$ is the hidden-layer output vector of *i*-th training instance ($i=1,2,\cdots,N$),

$$\mathrm{K}(\mathrm{H},\mathrm{H})=\begin{bmatrix}\kappa(\mathrm{h}(\mathrm{x}_1),\mathrm{h}(\mathrm{x}_1)) & \kappa(\mathrm{h}(\mathrm{x}_1),\mathrm{h}(\mathrm{x}_2)) & \cdots & \kappa(\mathrm{h}(\mathrm{x}_1),\mathrm{h}(\mathrm{x}_N))\\ \kappa(\mathrm{h}(\mathrm{x}_2),\mathrm{h}(\mathrm{x}_1)) & \kappa(\mathrm{h}(\mathrm{x}_2),\mathrm{h}(\mathrm{x}_2)) & \cdots & \kappa(\mathrm{h}(\mathrm{x}_2),\mathrm{h}(\mathrm{x}_N))\\ \vdots & \vdots & \ddots & \vdots\\ \kappa(\mathrm{h}(\mathrm{x}_N),\mathrm{h}(\mathrm{x}_1)) & \kappa(\mathrm{h}(\mathrm{x}_N),\mathrm{h}(\mathrm{x}_2)) & \cdots & \kappa(\mathrm{h}(\mathrm{x}_N),\mathrm{h}(\mathrm{x}_N))\end{bmatrix}$$

is the kernel matrix, $\mathrm{k}(\mathrm{h}(\bar{\mathrm{x}}),\mathrm{H})=\left(\kappa(\mathrm{h}(\bar{\mathrm{x}}),\mathrm{h}(\mathrm{x}_1)),\kappa(\mathrm{h}(\bar{\mathrm{x}}),\mathrm{h}(\mathrm{x}_2)),\cdots,\kappa(\mathrm{h}(\bar{\mathrm{x}}),\mathrm{h}(\mathrm{x}_N))\right)$ is the kernel vector,

and $$\kappa_{\mathrm{RBF}}(\mathrm{u},\mathrm{v})=\exp\left(-\frac{\|\mathrm{u}-\mathrm{v}\|^2}{2\lambda^2}\right),\mathrm{u}=(u_1,u_2,\cdots,u_D),\mathrm{v}=(v_1,v_2,\cdots,v_D) \tag{4}$$

is Radial Basis Function (RBF) kernel.

From Eq. (3), we can derive the posterior distribution of predicted output $\bar{\mathrm{y}}$ as

$$\mathrm{P}(\bar{\mathrm{y}}|\mathrm{h}(\bar{\mathrm{x}}),\mathrm{H},\mathrm{Y})\sim\mathrm{N}(\mu,\sigma^2), \tag{5}$$

where the mean and variance of this Gaussian distribution are

$$\mu=\mathrm{k}(\mathrm{h}(\bar{\mathrm{x}}),\mathrm{H})\left[\mathrm{K}(\mathrm{H},\mathrm{H})+\sigma_N^2\mathrm{I}\right]\mathrm{Y} \tag{6}$$

and

$$\sigma^2 = \kappa\left(h\left(\bar{x}\right), h\left(\bar{x}\right)\right) - k\left(h\left(\bar{x}\right), H\right)\left[K\left(H,H\right) + \sigma_N^2 I\right] k^T\left(h\left(\bar{x}\right), H\right), \tag{7}$$

Table 1. Details of 24 UCI Data Sets

	Data sets	Attributes	Classes	Class distribution	Instances
1	Auto Mpg	5	3	245/79/68	392
2	Blood Transfusion	4	2	570/178	748
3	Breast Cancer	10	2	458/241	699
4	Breast Cancer WD	30	2	357/212	569
5	Breast Cancer WP	33	2	151/47	198
6	Cleveland	13	5	160/54/35/ 35/13	297
7	Credit Approval	15	2	383/307	690
8	Cylinder Bands	20	2	312/228	540
9	Ecoli	5	8	143/77/52/35/20/5/2/2	336
10	Glass Identification	9	7	76/70/29/17/13/9/0	214
11	Haberman's Survival	3	2	225/81	306
12	Heart Disease	13	2	150/ 120	270
13	Image Segment	19	7	330×7	2310
14	Ionosphere	33	2	225/126	351
15	Iris	4	3	50×3	150
16	Magic Telescope	10	2	12332/6688	19020 (10%)
17	New Thyroid Gland	5	3	150/35/30	215
18	Page Blocks	10	5	4913/329/115/88/28	5473 (10%)
19	Parkinsons	22	2	147/48	195
20	Pima Indian Diabetes	8	2	500/268	768
21	Sonar	60	2	111/97	208
22	SPECTF Heart	44	2	212/55	267
23	Vehicle Silhouettes	18	4	218/217/212/199	846
24	Vowel Recognition	10	11	48×11	528

respectively, I is a N-by-N identity matrix. In GPRELM μ is used as the prediction output of unseen instance $\bar{x}$, i.e., let

$$\bar{y} = k\left(h\left(\bar{x}\right), H\right)\left[K\left(H,H\right) + \sigma_N^2 I\right] Y. \tag{8}$$

Meanwhile, GPRELM also defines the 95% confidence region for the estimation of unknown $\bar{y}$ as

$$\left[\mu - 1.96\sigma, \mu + 1.96\sigma\right].$$

So far, there is a parameter about which we don't discuss, that is the regularization factor σ_N^2 in Eqs. (6)-(8). This parameter is related to Gaussian Process Regression (GPR) which assumes that

$$\bar{y} = h\left(\bar{x}\right)\beta + \varepsilon, \tag{9}$$

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

3. Experimental Analysis on Prediction Performance of GPRELM

3.1. Experimental Setup

In this comparative study, we use 24 UCI [11] classification data sets to validate the prediction performance of

GPRELM. The details of these 24 UCI data sets are summarized in Table 1. The data sets are firstly.

Table 2 Training Accuracies of ELM, GPRELM$_{RBF}$ and GPRELM$_{Poly}$ on 24 UCI Data Sets

	ELM		GPRELM$_{RBF}$		GPRELM$_{Poly}$	
	Training accuracy	(L,σ_N)	Training accuracy	(L,σ_N,λ^2)	Training accuracy	(L,σ_N,b)
1	0.918	(150, 2^{-19})	1.000	(10, 2^{-24}, 2^{-9})	0.949	(30, 2^{11}, 15)
2	0.824	(150, 2^{-24})	0.956	(50, 2^{-14}, 2^{-6})	0.834	(90, 2^{-4}, 10)
3	1.000	(150, 2^{-14})	1.000	(10, 2^{-24}, 2^{-9})	1.000	(10, 2^{-24}, 10)
4	1.000	(150, 2^{-14})	1.000	(10, 2^{-24}, 2^{-9})	1.000	(10, 2^{-24}, 95)
5	1.000	(90, 2^{-24})	1.000	(10, 2^{-24}, 2^{-9})	1.000	(20, 2^{-24}, 55)
6	0.869	(130, 2^{-24})	1.000	(10, 2^{-24}, 2^{-9})	0.976	(60, 2^{-9}, 25)
7	0.854	(120, 2^{-24})	1.000	(10, 2^{-24}, 2^{-9})	0.894	(150, 2^{6}, 15)
8	0.935	(130, 2^{-19})	1.000	(10, 2^{-24}, 2^{-9})	1.000	(30, 2^{-24}, 80)
9	0.946	(100, 2^{-24})	1.000	(10, 2^{-24}, 2^{-9})	0.991	(140, 2^{-4}, 45)
10	0.995	(130, 2^{-24})	1.000	(10, 2^{-24}, 2^{-9})	1.000	(10, 2^{-24}, 15)
11	0.846	(80, 2^{-19})	1.000	(60, 2^{-19}, 2^{-8})	0.892	(60, 2^{-14}, 30)
12	1.000	(100, 2^{-24})	1.000	(10, 2^{-24}, 2^{-9})	1.000	(10, 2^{-24}, 5)
13	0.974	(130, 2^{-14})	1.000	(10, 2^{-24}, 2^{-9})	0.992	(80, 2^{21}, 20)
14	0.946	(120, 2^{-24})	1.000	(50, 2^{-19}, 2^{-8})	0.997	(140, 2^{-14}, 100)
15	1.000	(50, 2^{-24})	1.000	(10, 2^{-24}, 2^{-9})	1.000	(10, 2^{-24}, 5)
16	0.913	(150, 2^{-24})	1.000	(10, 2^{-24}, 2^{-9})	0.976	(100, 2^{-4}, 40)
17	1.000	(50, 2^{-24})	1.000	(10, 2^{-24}, 2^{-9})	1.000	(10, 2^{-24}, 5)
18	0.985	(90, 2^{-24})	1.000	(10, 2^{-19}, 2^{-5})	0.995	(80, 2^{-19}, 15)
19	1.000	(50, 2^{-19})	1.000	(10, 2^{-24}, 2^{-9})	1.000	(10, 2^{-24}, 5)
20	0.887	(150, 2^{-19})	1.000	(10, 2^{-24}, 2^{-9})	0.987	(50, 2^{-19}, 25)
21	0.952	(150, 2^{-19})	1.000	(70, 2^{-24}, 2^{-9})	1.000	(140, 2^{-9}, 100)
22	0.794	(90, 2^{-4})	0.831	(140, 2^{-19}, 2^{-9})	0.794	(80, 2^{-4}, 95)
23	0.928	(150, 2^{-24})	1.000	(10, 2^{-24}, 2^{-9})	0.993	(130, 2^{11}, 15)
24	1.000	(130, 2^{-19})	1.000	(10, 2^{-24}, 2^{-9})	1.000	(10, 2^{-24}, 35)

preprocessed as the following procedures: (1) deleting the discrete-valued attributes. ELMs are mainly used to handle the classification and regression problems with continuous-valued attributes. (2) filling in the missing attribute-values. We use the unsupervised filter named *ReplaceMissingValues* in Weka[4] to fill in all the missing attribute-values in each data set. It replaces all missing values of continuous attributes with the means of the training data. (3) reducing the large data sets. To compromise the running time, we adopt the unsupervised filter named *Resample* with the *sampleSizePercent* 10 in Weka to randomly reduce the sizes of 2 large data sets: Magic Telescope and Page Blocks.

For ELM, we use the method proposed in[9] to calculate the Moore-Penrose generalized inverse $\mathrm{H}^{\dagger}$ of hidden-layer output matrix H as follows:

$$\mathrm{H}^{\dagger}=\begin{cases}\left(\sigma_N^2\mathrm{I}+\mathrm{H}^{\mathrm{T}}\mathrm{H}\right)^{-1}\mathrm{H}^{\mathrm{T}}, & \text{if} \quad N\geq L\\ \mathrm{H}^{\mathrm{T}}\left(\sigma_N^2\mathrm{I}+\mathrm{H}\mathrm{H}^{\mathrm{T}}\right)^{-1}, & \text{if} \quad N<L\end{cases}. \tag{10}$$

Besides RBF kernel used in GPRELM[8], we also consider using another kernel to construct GPRELM model, i.e., polynomial kernel

$$\kappa_{\mathrm{Poly}}\left(\mathrm{u},\mathrm{v}\right)=\left(\mathrm{u}\bullet\mathrm{v}+1\right)^{b},\mathrm{u}=\left(u_1,u_2,\cdots,u_D\right),\mathrm{v}=\left(v_1,v_2,\cdots,v_D\right). \tag{11}$$

Table 3. Testing Accuracies of ELM, GPRELM$_{RBF}$ and GPRELM$_{Poly}$ on 24 UCI Data Sets

	ELM		GPRELM$_{RBF}$		GPRELM$_{Poly}$	
	Testing accuracy	(L,σ_N)	Testing accuracy	(L,σ_N,λ^2)	Testing accuracy	(L,σ_N,b)
1	0.819	(30, 2^{-14})	0.824	(40, 2^{-4}, 2^{-4})	0.809	(30, 2^{1}, 5)
2	0.802	(80, 2^{-9})	0.806	(100, 2^{-24}, 2^{8})	0.803	(40, 2^{-9}, 10)
3	0.971	(130, 2^{1})	0.974	(140, 2^{11}, 2^{-5})	0.970	(90, 2^{16}, 5)
4	0.974	(150, 2^{-9})	0.977	(40, 2^{-4}, 2^{6})	0.977	(60, 2^{6}, 5)
5	0.763	(10, 2^{-14})	0.773	(50, 2^{-9}, 2^{4})	0.768	(40, 2^{6}, 5)
6	0.579	(20, 2^{-4})	0.592	(40, 2^{-4}, 2^{1})	0.579	(40, 2^{1}, 5)
7	0.777	(80, 2^{-19})	0.781	(80, 2^{-14}, 2^{3})	0.780	(10, 2^{1}, 10)
8	0.678	(150, 2^{-9})	0.687	(70, 2^{-9}, 2^{-1})	0.689	(40, 2^{11}, 10)
9	0.875	(30, 2^{-9})	0.881	(140, 2^{-9}, 2^{8})	0.878	(70, 2^{21}, 10)
10	0.673	(30, 2^{-9})	0.715	(150, 2^{-4}, 2^{-2})	0.701	(10, 2^{-4}, 5)
11	0.771	(120, 2^{-9})	0.775	(20, 2^{-14}, 2^{2})	0.771	(130, 2^{21}, 10)
12	0.837	(30, 2^{-4})	0.841	(100, 2^{-4}, 2^{1})	0.833	(20, 2^{6}, 5)
13	0.951	(130, 2^{-14})	0.969	(90, 2^{-4}, 2^{5})	0.961	(110, 2^{-9}, 5)
14	0.838	(50, 2^{-14})	0.846	(50, 2^{-14}, 2^{-4})	0.840	(40, 2^{-19}, 85)
15	0.980	(80, 2^{-9})	0.987	(100, 2^{-14}, 2^{7})	0.987	(100, 2^{6}, 5)
16	0.840	(40, 2^{-19})	0.855	(80, 2^{-4}, 2^{4})	0.845	(10, 2^{1}, 10)
17	0.953	(90, 2^{-14})	0.963	(140, 2^{1}, 2^{-9})	0.930	(100, 2^{1}, 5)
18	0.936	(110, 2^{-9})	0.947	(20, 2^{-9}, 2^{-1})	0.945	(110, 2^{1}, 5)
19	0.933	(140, 2^{-9})	0.938	(130, 2^{-24}, 2^{-1})	0.923	(100, 2^{21}, 10)
20	0.779	(90, 2^{-4})	0.784	(20, 2^{-9}, 2^{10})	0.783	(60, 2^{21}, 10)
21	0.745	(90, 2^{-14})	0.784	(150, 2^{-14}, 2^{3})	0.774	(140, 2^{16}, 20)
22	0.794	(90, 2^{-4})	0.794	(110, 2^{-9}, 2^{1})	0.794	(70, 2^{1}, 5)
23	0.768	(120, 2^{-14})	0.792	(110, 2^{-14}, 2^{5})	0.783	(90, 2^{1}, 5)
24	0.881	(150, 2^{-14})	0.956	(130, 2^{-24}, 2^{9})	0.911	(150, 2^{1}, 5)

We term GPRELMs with RBF kernel in Eq. (4) and polynomial kernel in Eq. (11) as GPRELM$_{RBF}$ and GPRELM$_{Poly}$, respectively.

There are four parameters that require to be determined in our experiment, i.e., the number L of hidden-layer nodes, regularization factor σ_N, λ^2 in RBF kernel, and b in polynomial kernel. We set them as: $L=\{10,20,\cdots,140,150\}$, $\sigma_N=\{2^{-24},2^{-19},\cdots,2^{16},2^{21}\}$, $\lambda^2=\{2^{-9},2^{-8},\cdots,2^{9},2^{10}\}$, and $b=\{5,10,\cdots,95,100\}$. We compare the training accuracy, testing accuracy, training time and testing time of ELM, GPRELM$_{RBF}$ and GPRELM$_{Poly}$. For any given (L,σ_N), (L,σ_N,λ^2), and (L,σ_N,b), the experimental results corresponding to ELM, GPRELM$_{RBF}$ and GPRELM$_{Poly}$ are obtained based on the procedure of 10-times 10-fold cross-validation.

3.2. Experimental Result and Analysis

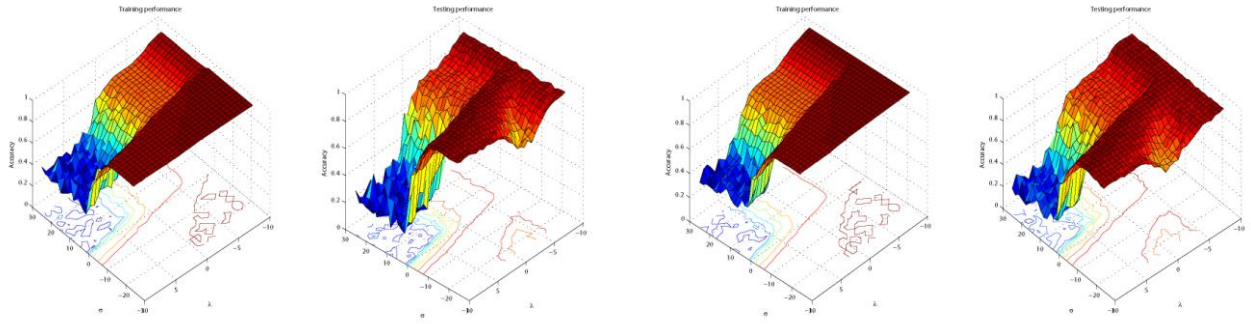
For 150 pairs of (L,σ_N), 3000 triples of (L,σ_N,λ^2), and 3000 triples of (L,σ_N,b), Tables 2 and 3 respectively present the best training and testing accuracies of ELM, GPRELM$_{RBF}$ and GPRELM$_{Poly}$. Meanwhile, Table 4 gives the training and testing times of ELM, GPRELM$_{RBF}$ and GPRELM$_{Poly}$ on 24 UCI data sets corresponding to the best training and testing accuracies. From Table 2, we can see that (1) GPRELM$_{RBF}$ and GPRELM$_{Poly}$ obtain the better training accuracies than ELM; (2) RBF kernel make GPRELM get the better training accuracy than polynomial kernel; (3) GPRELM$_{RBF}$ obtains the better training accuracy with less hidden-layer nodes than ELM and GPRELM$_{Poly}$ on 23 data sets; and (4) GPRELM$_{RBF}$ obtains the better training accuracy with smaller regularization factor σ_N than ELM and GPRELM$_{Poly}$. From Table 3, we can see that (1) GPRELM$_{RBF}$ and

GPRELM$_{Poly}$ obtain the better testing accuracies than ELM; (2) RBF kernel make GPRELM get the better testing accuracy than polynomial kernel; (3) GPRELM$_{RBF}$ obtains the better testing accuracy with more hidden-layer nodes than ELM and GPRELM$_{Poly}$ on 16 data sets; and (4) GPRELM$_{RBF}$ obtains the better training accuracy with smaller regularization factor σ_N than ELM and GPRELM$_{Poly}$. From Table 4, we can see the training and testing times of GPRELM$_{RBF}$ and GPRELM$_{Poly}$ are all higher than ELM, because the calculations of kernel matrix and kernel vector are time-consuming.

Table 4. Training/testing times of ELM, GPRELM$_{RBF}$ and GPRELM$_{Poly}$ on 24 UCI Data Sets

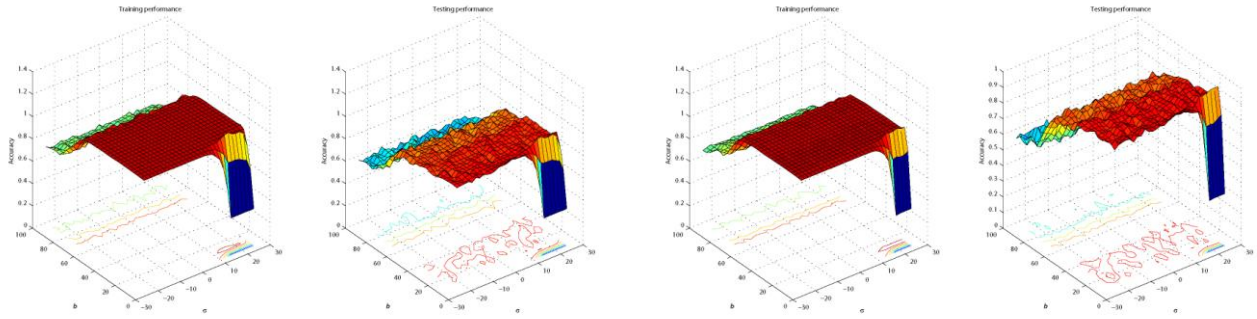
	ELM		GPRELM$_{RBF}$		GPRELM$_{Poly}$	
	Training time	Testing time	Training time	Testing time	Training time	Testing time
1	0.047	0.008	0.117	0.141	0.148	0.125
2	0.078	0.000	0.523	0.328	0.555	0.375
3	0.078	0.016	0.328	0.187	0.344	0.336
4	0.078	0.016	0.258	0.266	0.320	0.305
5	0.016	0.000	0.023	0.023	0.047	0.031
6	0.031	0.000	0.063	0.078	0.125	0.094
7	0.047	0.008	0.328	0.313	0.461	0.391
8	0.047	0.000	0.258	0.242	0.430	0.313
9	0.008	0.000	0.055	0.102	0.109	0.133
10	0.008	0.000	0.031	0.023	0.063	0.063
11	0.008	0.000	0.086	0.063	0.109	0.078
12	0.016	0.000	0.055	0.047	0.078	0.078
13	0.125	0.031	6.031	3.266	6.133	3.766
14	0.031	0.000	0.125	0.125	0.156	0.125
15	0.000	0.000	0.023	0.016	0.016	0.016
16	0.070	0.000	0.555	0.461	0.656	0.508
17	0.008	0.000	0.031	0.07	0.023	0.047
18	0.016	0.000	0.242	0.242	0.234	0.273
19	0.008	0.008	0.031	0.031	0.047	0.031
20	0.070	0.008	0.430	0.367	0.563	0.430
21	0.023	0.000	0.055	0.047	0.078	0.086
22	0.016	0.000	0.094	0.063	0.070	0.078
23	0.070	0.000	0.570	0.531	0.875	0.609
24	0.047	0.000	0.320	0.227	0.242	0.305

In addition, we also check the impact of different parameter pairs on the training and testing accuracies of GPRELM$_{RBF}$ and GPRELM$_{Poly}$. Taking the famous *Iris* data set as an example, we validate the impact of σ_N and λ^2 on GPRELM$_{RBF}$ (Fig. 1), impact of σ_N and b on GPRELM$_{Poly}$ (Fig. 2), impact of L and σ_N on GPRELM$_{RBF}$ (Fig. 3), impact of L and σ_N on GPRELM$_{Poly}$ (Fig. 4), impact of L and λ^2 on GPRELM$_{RBF}$ (Fig. 5), and impact of L and



(a) L=50 (b) L=150

Fig. 1.Impact of parameters σ_N and λ^2 on GPRELM$_{RBF}$ for *Iris* data set



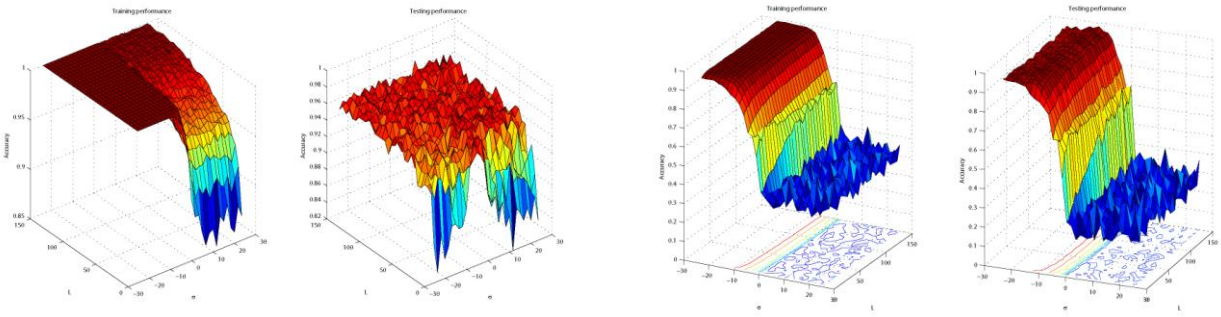
(a) L=50 (b) L=150

Fig. 2. Impact of parameters σ_N and b on GPRELM$_{Poly}$ for *Iris* data set

b on GPRELM$_{Poly}$ (Fig. 6). From Fig. 1, we can know the smaller regularization factor σ_N and λ^2 bring about the higher training and testing accuracies for GPRELM$_{RBF}$. From Fig. 2, we can find the regularization factor σ_N doesn't influence the training and testing accuracies for GPRELM$_{Poly}$ remarkably. The smaller b leads to the higher training and testing accuracies for GPRELM$_{Poly}$. By comparing the Fig. 3-(a) with Fig. 3-(b), we know that when λ^2 is small, the training and testing accuracies for GPRELM$_{RBF}$ are not remarkably impacted by L and σ_N. Fig. 4 tells us that when b is large, L and σ_N can't observably impact the training and testing accuracies for GPRELM$_{Poly}$. From Figs. 5 and 6, we know when the smaller σ_N is selected, the training and testing accuracies of GPRELM$_{RBF}$ and GPRELM$_{Poly}$ can't be influenced by parameters L, λ^2, and b.

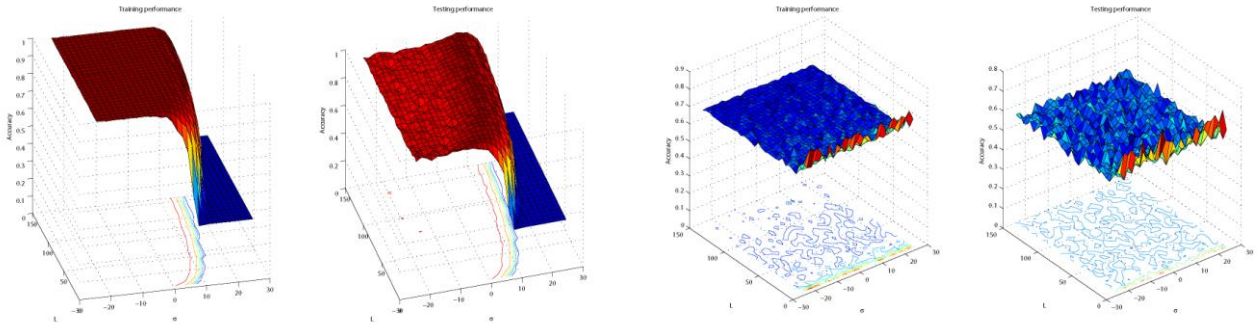
4. Conclusion

In this paper, we empirically investigate the classification performances of two kinds of Gaussian Process Regression based Extreme Learning Machine (GPRELM), i.e., GPRELM with Radial Basis Function kernel (GPRELM$_{RBF}$) and GPRELM with polynomial kernel (GPRELM$_{Poly}$). The final results tell us that (1) GPRELMs can obtain the better generalization performances than ELM and meanwhile exist the serious over-fitting; (2) the number of hidden-layer nodes can't remarkably impact the training and testing accuracies of GPRELM$_{RBF}$ and GPRELM$_{Poly}$; (3) the smaller regularization factors usually make the prediction of GPRELMs more stable. All these results provide the useful enlightenments and instructions for the theoretical studies and practical applications of GPRELMs.



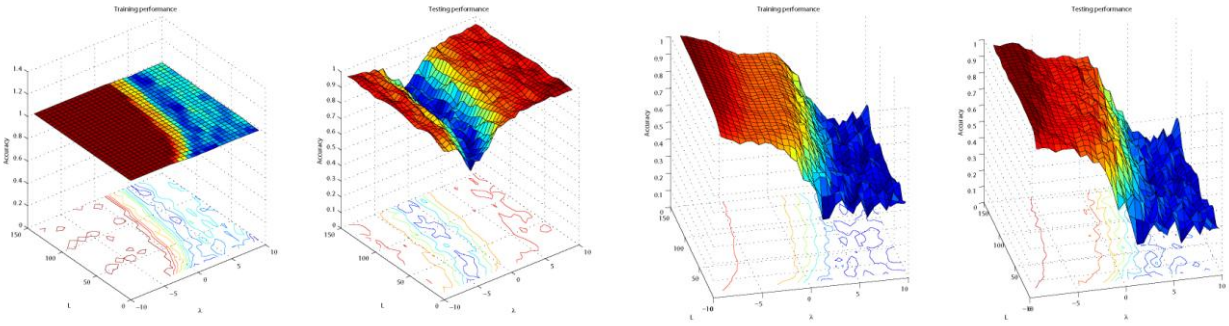
(a) $\lambda^2 = 2^{-9}$ (b) $\lambda^2 = 2^{10}$

Fig. 3.Impact of parameters$L$ andσ_N on GPRELM$_{RBF}$ for *Iris* data set



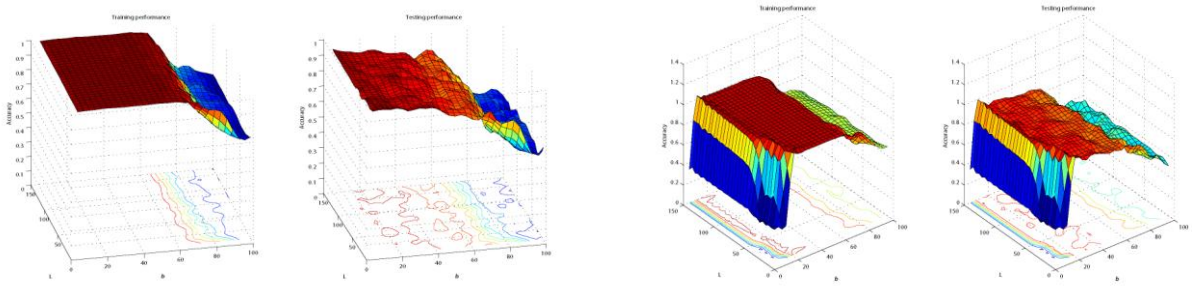
(a) b=5 (b) b=100

Fig. 4.Impact of parameters$L$ andσ_N on GPRELM$_{Poly}$ for *Iris* data set



(a) $\sigma_N = 2^{-24}$ (b) $\sigma_N = 2^{21}$

Fig. 5.Impact of parameters$L$ andλ^2 on GPRELM$_{RBF}$ for *Iris* data set



(a) $\sigma_N = 2^{-24}$ (b) $\sigma_N = 2^{21}$

Fig. 6.Impact of parameters$L$ and$b$ on GPRELM$_{Poly}$ for *Iris* data set

Acknowledgment

We thank the Editor and anonymous reviewers very much for their valuable comments which help us to improve this paper significantly. This paper was supported by Science and Technology Foundation of Agricultural University of Hebei (LG201634), China Postdoctoral Science Foundation (2016T90799), and National Natural Science Foundation of China (61503252).

References

[1] Cao, J., Lin, Z., & Huang, G. B. (2012). Self-adaptive evolutionary extreme learning machine. *Neural Processing Letters*, *36(3)*, 285-305.

[2] Chacko, B. P., Krishnan, R. V., Raju, G., & Anto, P. B. (2012). Handwritten character recognition using wavelet energy and extreme learning machine. *International Journal of Machine Learning and Cybernetics*, *3(2)*, 149-161.

[3] Chatzis, S. P., Korkinof, D., & Demiris, Y. (2011). The one-hidden layer non-parametric Bayesian kernel machine. *Proceedings of IEEE International Conference on Tools with Artificial Intelligence*.

[4] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten I. H. (2009). The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, *11(1)*, 10-18.

[5] Heeswijk, M.V., & Miche, Y. (2015). Binary/ternary extreme learning machines. *Neurocomputing*, 149, 187-197.

[6] Heeswijk, M., Miche, Y., Lindh-Knuutila, T., Hilbers, A. P. J., Honkela, T., Oja, E., & Lendasse, A. (2009). Adaptive ensemble models of extreme learning machines for time series prediction. *Lecture Notes in Computer Science*, 5769,305-314.

[7] Helmy, T., & Rasheed, Z. (2009). Multi-category bioinformatics dataset classification using extreme learning machine. *Proceedings of IEEE Congress on Evolutionary Computation*.

[8] Huang, G. B., Wang, D., & Lan, Y. (2011). Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, *2(2)*, 107-122.

[9] Huang, G. B., Zhou, H. M., Ding, J. X., & Zhang, R. (2012). Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *42(2)*, 513-529.

[10] Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing*, *70(1)*, 489-501.

[11] Lichman, M. (2013). UCI machine learning repository. Irvine, CA: University of California, School of Information and Computer Science.

[12] Matias, T., Souza, F., Araújo, R., & Antunes, C. H. (2014). Learning of a single-hidden layer feedforward neural network using an optimized extreme learning machine. *Neurocomputing*, *129*, 428-436.

[13] Wang, Y. G., Cao, F. L., & Yuan Y. B. (2011). A study on effectiveness of extreme learning machine. *Neurocomputing*, *74(16)*, 2483-2490.

[14] Zhang, Y., Xu, B., & Li., H. B. (2015). Adaptive neural control of a quadrotor helicopter with extreme learning machine. *Proceedings in Adaptation, Learning and Optimization*

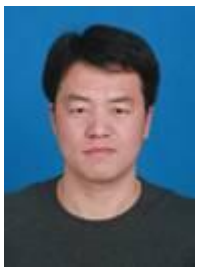
[15] Zheng, W. B., Qian, Y., & Lu, H. J. (2013). Text categorization based on regularization extreme learning machine. *Neural Computing and Applications*, *22(3-4)*, 447-456.

[16] Zhu, Q. Y., Qin, A., Suganthan, P., & Huang, G. B. (2005). Evolutionary extreme learning machine. *Pattern Recognition*, *38(10)*, 1759-1763.

[17] Zong, W. W., & Huang, G. B. (2011). Face recognition based on extreme learning machine. *Neurocomputing*, *74(16)*, 2541-2551.

Jing Zhou received her bachelor degree in mathematics and applied mathematics from Hebei Normal University in June 2003, and received her Master degree in Fundamental Mathematics from Hebei University in June 2010. She has been engaged in the teaching and research of mathematics in Hebei Agricultural University since July 2003. Her research interests include extreme learning machine, artificial neural networks, probability density function estimation, and Bayesian network.

Ruiying Liu received her bachelor degree in information and computing science from Yanshan University in June 2004,and received her Master degree in Applied Mathematics from Hebei University in June 2009. She has been teaching in Hebei Agricultural University since July 2004. Her research interests include artificial neural networks and their practical applications.



Xu Zhou received his Bachelor's degree in mathematics and application from Agricultural University of Hebei, Baoding, China, in June 2010. He received his Master degree in Application Mathematics from Hebei University of Hebei, Baoding, China, in June 2013. He is currently a mathematics teacher in Agricultural University of Hebei, Baoding, China. His research interests include artificial neural networks, machine learning, extreme learning machine, and support vector machine.

Rana Aamir Raza Ashfaq received his master degree in computer science from Blekinge Tekniska Hgskola (BTH), Sweden. He also received his Bachelor and Master Degrees in Computer Science from Bahauddin Zakariya University, Multan, Pakistan. Since 2010 he is working as Assistant Professor in Department of Computer Science, Bahauddin Zakariya University, Multan, Pakistan. He is currently a Ph.D. student in College of Computer Science & Software Engineering, Shenzhen University, Shenzhen, Guangdong, China. His main research interests include machine learning and big data mining.