Use Correlation Coefficients in Gaussian Process to Train Stable ELM Models

Yulin He[1(✉)], Joshua Zhexue Huang[1], Xizhao Wang[1], and Rana Aamir Raza[2]

[1] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
{yulinhe,zx.huang,xzwang}@szu.edu.cn
[2] College of Information and Communication Engineering, Shenzhen University, Shenzhen 518060, China
aamir@szu.edu.cn

Abstract. This paper proposes a new method to train stable extreme learning machines (ELM). The new method, called StaELM, uses correlation coefficients in Gaussian process to measure the similarities between different hidden layer outputs. Different from kernel operations such as linear or RBF kernels to handle hidden layer outputs, using correlation coefficients can quantify the similarity of hidden layer outputs with real numbers in $(0, 1]$ and avoid covariance matrix in Gaussian process to become a singular matrix. Training through Gaussian process results in ELM models insensitive to random initialization and can avoid over-fitting. We analyse the rationality of StaELM and show that existing kernel-based ELMs are special cases of StaELM. We used real world datasets to train both regression and classification StaELM models. The experiment results have shown that StaELM models achieved higher accuracies in both regression and classification in comparison with traditional kernel-based ELMs. The StaELM models are more stable with respect to different random initializations and less over-fitting. The training process of StaELM models is also faster.

Keywords: Extreme learning machine · Correlation coefficient · Gaussian process · Neural network

1 Introduction

Extreme learning machine (ELM) is a special single-hidden layer feed-forward neural network (SLFN) [6]. Due to its lower computational complexity and better generalization performance, ELM has recently attracted a lot of interests in research and industry and is used in a wide range of applications [5]. ELM uses a random method to determine input weights/hidden layer biases and analytically computes the output weights. Therefore, it is extremely fast to train an ELM model. It has also been proved that ELM can guarantee the universal approximate capability of ELM [3].

T. Cao et al. (Eds.): PAKDD 2015, Part I, LNAI 9077, pp. 405–417, 2015.
DOI: 10.1007/978-3-319-18038-0_32

Currently, ELM has two main problems in practical applications. The first problem is that the trained ELM model is sensitive to the random initial settings [15]. Different initial settings often result in different performances, which implies that the training process produces instable ELM models from different initial settings. The second problem is over-fitting [3], which is usually caused by the numerous hidden layer nodes specified to best approximate the training data set. A number of improvements have been proposed to tackle these problems. One approach is to optimize the random weights with different evolutionary algorithms. Examples include E-ELM [15], SaE-ELM [1], and O-ELM [9]. Another approach is to select better architectures for ELM, for instance, I-ELMs [3,4], OP-ELM [10], and localized generalization error ELM [13]. Although the literatures reported the better performances of these improved ELM models, the higher computational complexity makes them impractical to deal with the regression and classification tasks with a large number of training instances.

A different direction to improve ELM without increase of computational complexity is to estimate the prior probability distribution of ELM models. Soria-Olivas *et al.* [12] designed a Bayesian ELM (BELM). Luo *et al.* [8,14] proposed sparse Bayesian ELM (SBELM).[1]. Chatzis *et al.* [2] proposed the one-hidden-layer nonparametric Bayesian kernel machine (1HNBKM). Because BELM and 1HNBKM used linear and RBF kernels to handle the hidden layer outputs, we call them kernel-based ELMs in this paper. The empirical analysis shows that kernel-based ELMs are still sensitive to random initialization. For example, there is an obvious difference between the predictive results of BELM and 1HNBKM on *Libras Movement* dataset[2] with random input weights in intervals $[0, 1]$ and $[-1, 1]$. In addition, the over-fitting still exists for 1HNBKM.

In this paper, we propose to use Gaussian process to train ELM models and present a stable extreme learning machine algorithm, StaELM. In this algorithm, we use correlation coefficients in Gaussian process to measure the similarity between different hidden layer outputs with real numbers in $(0, 1]$. The advantages of using Gaussian process in ELM model training over aforementioned training methods are that the training process is fast and the trained ELM models are insensitive to random initialization and can avoid over-fitting. In the training process, we use correlation coefficients to avoid the covariance matrix to become a singular matrix and make the inverse of covariance matrix solvable.

We have used 12 UCI and KEEL[3] datasets to conduct the experiments and compared the performances of accuracy and running time of StaELM, orginal ELM, BELM, and 1HNBKM. The experimental results show that StaELM models achieved higher accuracies and lower running time in both regression and

[1] The main difference between SBELM and BELM is that the independent regularization priors in SBELM are imposed on each weight instead of one shared prior for all weights in BELM. Because SBELM and BELM are homologous, we only discuss and analyse BELM in this paper due to its simplicity.

[2] http://archive.ics.uci.edu/ml/

[3] http://sci2s.ugr.es/keel/datasets.php

classification than other methods. The StaELM models are stable with respect to different random initializations and less over-fitting.

The rest of this paper is organized as follows: In Section 2, we briefly summarize kernel-based ELMs. Section 3 introduces our proposed StaELM. Experimental simulations are presented in Section 4. Finally, we conclude this paper in Section 5.

2 Kernel-based ELMs

In this section, we review three existing ELMs models, i.e., the original ELM, BELM, and 1HNBKM. Because the first two use linear kernels to handle the hidden layer outputs whereas the last one uses RBF kernel for that purpose. We call them kernel-based ELMs.

2.1 ELM

ELM [6] is a single-hidden layer feed-forward neural network (SLFN) and does not require any iterative optimization to input/output weights. Given the training dataset $(\mathrm{X}_{N\times D}, \mathrm{Y}_{N\times C})$ and testing dataset $(\mathrm{X'}_{M\times D}, \mathrm{Y'}_{M\times C})$:

$$\mathrm{X}_{N\times D}=\begin{bmatrix}\mathrm{x}_1\\ \mathrm{x}_2\\ \vdots\\ \mathrm{x}_N\end{bmatrix}=\begin{bmatrix}x_{11} & x_{12} & \cdots & x_{1D}\\ x_{21} & x_{22} & \cdots & x_{2D}\\ \vdots & \vdots & \ddots & \vdots\\ x_{N1} & x_{N2} & \cdots & x_{ND}\end{bmatrix}, \mathrm{Y}_{N\times C}=\begin{bmatrix}\mathrm{y}_1\\ \mathrm{y}_2\\ \vdots\\ y_N\end{bmatrix}=\begin{bmatrix}y_{11} & y_{12} & \cdots & y_{1C}\\ y_{21} & y_{22} & \cdots & y_{2C}\\ \vdots & \vdots & \ddots & \vdots\\ y_{N1} & y_{N2} & \cdots & y_{NC}\end{bmatrix} \tag{1}$$

and

$$\mathrm{X'}_{M\times D}=\begin{bmatrix}\mathrm{x'}_1\\ \mathrm{x'}_2\\ \vdots\\ \mathrm{x'}_M\end{bmatrix}=\begin{bmatrix}x'_{11} & x'_{12} & \cdots & x'_{1D}\\ x'_{21} & x'_{22} & \cdots & x'_{2D}\\ \vdots & \vdots & \ddots & \vdots\\ x'_{M1} & x'_{M2} & \cdots & x'_{MD}\end{bmatrix}, \mathrm{Y'}_{M\times C}=\begin{bmatrix}\mathrm{y'}_1\\ \mathrm{y'}_2\\ \vdots\\ \mathrm{y'}_M\end{bmatrix}=\begin{bmatrix}y'_{11} & y'_{12} & \cdots & y'_{1C}\\ y'_{21} & y'_{22} & \cdots & y'_{2C}\\ \vdots & \vdots & \ddots & \vdots\\ y'_{M1} & y'_{M2} & \cdots & y'_{MC}\end{bmatrix}, \tag{2}$$

where N is the number of training instances, D is the number of input variables, M is the number of testing instances, and C is the number of output variables. Usually, $\mathrm{Y'}_{M\times C}$ is unknown and needs to be predicted. ELM determines $\mathrm{Y'}_{M\times C}$ as follows:

$$\mathrm{Y'}_{M\times C}=\mathrm{H'}_{M\times L}\beta_{L\times C}=\begin{cases}\mathrm{H'}\left(\mathrm{H^T H}\right)^{-1}\mathrm{H^T Y}, \text{if } N\geq L\\ \mathrm{H'H^T}\left(\mathrm{HH^T}\right)^{-1}\mathrm{Y}, \text{if } N<L\end{cases}, \tag{3}$$

where $\beta_{L\times C}$ is the output weights, L is the number of hidden layer nodes,

$$\mathrm{H}_{N\times L}=\begin{bmatrix}\mathrm{h}\left(\mathrm{x}_1\right)\\ \mathrm{h}\left(\mathrm{x}_2\right)\\ \vdots\\ \mathrm{h}\left(\mathrm{x}_N\right)\end{bmatrix}=\begin{bmatrix}g\left(\mathrm{w}_1\mathrm{x}_1+b_1\right) & g\left(\mathrm{w}_2\mathrm{x}_1+b_2\right) & \cdots & g\left(\mathrm{w}_L\mathrm{x}_1+b_L\right)\\ g\left(\mathrm{w}_1\mathrm{x}_2+b_1\right) & g\left(\mathrm{w}_2\mathrm{x}_2+b_2\right) & \cdots & g\left(\mathrm{w}_L\mathrm{x}_2+b_L\right)\\ \vdots & \vdots & \ddots & \vdots\\ g\left(\mathrm{w}_L\mathrm{x}_N+b_L\right) & g\left(\mathrm{w}_2\mathrm{x}_N+b_L\right) & \cdots & g\left(\mathrm{w}_L\mathrm{x}_N+b_L\right)\end{bmatrix} \tag{4}$$

is the hidden layer output matrix for training instances,

$$\mathrm{H'}_{M\times L}=\begin{bmatrix}\mathrm{h}\left(\mathrm{x'}_1\right)\\ \mathrm{h}\left(\mathrm{x'}_2\right)\\ \vdots\\ \mathrm{h}\left(\mathrm{x'}_M\right)\end{bmatrix}=\begin{bmatrix}g\left(\mathrm{w}_1\mathrm{x'}_1+b_1\right) & g\left(\mathrm{w}_2\mathrm{x'}_1+b_2\right) & \cdots & g\left(\mathrm{w}_L\mathrm{x'}_1+b_L\right)\\ g\left(\mathrm{w}_1\mathrm{x'}_2+b_1\right) & g\left(\mathrm{w}_2\mathrm{x'}_2+b_2\right) & \cdots & g\left(\mathrm{w}_L\mathrm{x'}_2+b_L\right)\\ \vdots & \vdots & \ddots & \vdots\\ g\left(\mathrm{w}_L\mathrm{x'}_M+b_L\right) & g\left(\mathrm{w}_2\mathrm{x'}_M+b_L\right) & \cdots & g\left(\mathrm{w}_L\mathrm{x'}_M+b_L\right)\end{bmatrix} \tag{5}$$

is the hidden layer output matrix for testing instances, $g\left(z\right)=\frac{1}{1+e^{-z}}$ is sigmoid activation function,

$$\mathrm{W}_{D\times L}=\left[\mathrm{w}_1\ \mathrm{w}_2\cdots\mathrm{w}_L\right]=\begin{bmatrix}w_{11} & w_{21} & \cdots & w_{L1}\\ w_{12} & w_{22} & \cdots & w_{L2}\\ \vdots & \vdots & \ddots & \vdots\\ w_{1D} & w_{2D} & \cdots & w_{LD}\end{bmatrix}\ \text{and}\ \mathrm{b}=\begin{bmatrix}b_1\\ b_2\\ \vdots\\ b_L\end{bmatrix} \tag{6}$$

are input weight and hidden layer biases which are randomly determined. ELM is sensitive to random initialization and has obvious over-fitting. In order to tackle these problems, two improvements, i.e., BELM and 1HNBKM, are discussed below.

2.2 BELM

BELM [12] optimizes the output weights β by using Bayesian linear regression as follows:

$$y=\mathrm{h}\left(\mathrm{x}\right)\beta+\varepsilon, \tag{7}$$

where $\varepsilon\sim\mathrm{N}\left(0,\sigma_N^2\right)$ and $\beta\sim\mathrm{N}\left(0,\alpha^{-1}\mathrm{I}_{L\times L}\right)$. The posterior distribution over output weights β is expressed as

$$\mathrm{P}\left(\beta\left|\mathrm{H},\mathrm{Y}\right.\right)=\mathrm{N}\left(\beta,S\right), \tag{8}$$

where $\beta=\sigma_N^{-2}S\mathrm{H}^{\mathrm{T}}\mathrm{Y}$ and $S=\left(\alpha\mathrm{I}+\sigma_N^{-2}\mathrm{H}^{\mathrm{T}}\mathrm{H}\right)^{-1}$ are the mean and covariance matrix respectively. For a new instance $\mathrm{x'}=\left(x'_1,x'_2,\cdots,x'_D\right)$, the output y' predicted with BELM obeys the Gaussian distribution $\mathrm{N}\left(\mu,\sigma^2\right)$, where

$$\mu=\mathrm{h}\left(\mathrm{x'}\right)\beta, \tag{9}$$

$$\sigma^2=\sigma_N^2+\mathrm{h}\left(\mathrm{x'}\right)S\mathrm{h}^{\mathrm{T}}\left(\mathrm{x'}\right). \tag{10}$$

In BELM, μ is deemed as the prediction of new instance $\mathrm{x'}$, i.e., $y'=\mu$, and σ^2 is the variance which is used to determine the confidence interval of prediction y'. There are two parameters that need to be determined in BELM: σ_N^2 and $\alpha>0$. BELM effectively controls the over-fitting but still sensitive to randomly initial weights is still not solved.

2.3 1HNBKM

Given a training dataset $(\mathrm{X}_{N\times D}, \mathrm{Y}_{N\times C})$, 1HNBKM [2] predicts the output y' for new instance x' via the following joint probability distribution

$$\begin{bmatrix} \mathrm{Y} \\ y' \end{bmatrix} \sim \mathrm{N}\left(0, \begin{bmatrix} \mathrm{K}\left(\mathrm{H},\mathrm{H}\right)+\sigma_N^2\mathrm{I} & \mathrm{k}^{\mathrm{T}}\left(\mathrm{h}\left(\mathrm{x}'\right),\mathrm{H}\right) \\ \mathrm{k}\left(\mathrm{h}\left(\mathrm{x}'\right),\mathrm{H}\right) & k\left(\mathrm{h}\left(\mathrm{x}'\right),\mathrm{h}\left(\mathrm{x}'\right)\right) \end{bmatrix}\right), \tag{11}$$

where the meaning of σ_N^2 is same as in BELM,

$$\mathrm{K}\left(\mathrm{H},\mathrm{H}\right)_{N\times N} = \begin{bmatrix} k\left(\mathrm{h}\left(\mathrm{x}_1\right),\mathrm{h}\left(\mathrm{x}_1\right)\right) & k\left(\mathrm{h}\left(\mathrm{x}_1\right),\mathrm{h}\left(\mathrm{x}_2\right)\right) & \cdots & k\left(\mathrm{h}\left(\mathrm{x}_1\right),\mathrm{h}\left(\mathrm{x}_N\right)\right) \\ k\left(\mathrm{h}\left(\mathrm{x}_2\right),\mathrm{h}\left(\mathrm{x}_1\right)\right) & k\left(\mathrm{h}\left(\mathrm{x}_2\right),\mathrm{h}\left(\mathrm{x}_2\right)\right) & \cdots & k\left(\mathrm{h}\left(\mathrm{x}_2\right),\mathrm{h}\left(\mathrm{x}_N\right)\right) \\ \vdots & \vdots & \ddots & \vdots \\ k\left(\mathrm{h}\left(\mathrm{x}_N\right),\mathrm{h}\left(\mathrm{x}_1\right)\right) & k\left(\mathrm{h}\left(\mathrm{x}_N\right),\mathrm{h}\left(\mathrm{x}_2\right)\right) & \cdots & k\left(\mathrm{h}\left(\mathrm{x}_N\right),\mathrm{h}\left(\mathrm{x}_N\right)\right) \end{bmatrix} \tag{12}$$

is a kernel matrix,

$$\mathrm{k}\left(\mathrm{h}\left(\mathrm{x}'\right),\mathrm{H}\right) = \begin{bmatrix} k\left(\mathrm{h}\left(\mathrm{x}'\right),\mathrm{h}\left(\mathrm{x}_1\right)\right) & k\left(\mathrm{h}\left(\mathrm{x}'\right),\mathrm{h}\left(\mathrm{x}_2\right)\right) & \cdots & k\left(\mathrm{h}\left(\mathrm{x}'\right),\mathrm{h}\left(\mathrm{x}_N\right)\right) \end{bmatrix} \tag{13}$$

is a kernel vector, and

$$k\left(\mathrm{u},\mathrm{v}\right) = \exp\left(-\frac{\left\|\mathrm{u}-\mathrm{v}\right\|^2}{2\lambda^2}\right). \tag{14}$$

is the RBF kernel function. We can find $k\left(\mathrm{u},\mathrm{v}\right)=1$, when u=v.

Then, the posterior distribution of predicted output y' is

$$\mathrm{P}\left(y'\left|\mathrm{h}\left(\mathrm{x}'\right),\mathrm{H},\mathrm{Y}\right.\right) = \mathrm{N}\left(\mu,\sigma^2\right), \tag{15}$$

where

$$\mu = \mathrm{k}\left(\mathrm{h}\left(\mathrm{x}'\right),\mathrm{H}\right)\left(\mathrm{K}\left(\mathrm{H},\mathrm{H}\right)+\sigma_N^2\mathrm{I}\right)^{-1}\mathrm{Y}, \tag{16}$$

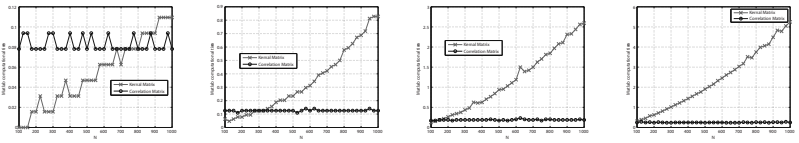
$$\sigma^2 = k\left(\mathrm{h}\left(\mathrm{x}'\right),\mathrm{h}\left(\mathrm{x}'\right)\right) - \mathrm{k}\left(\mathrm{h}\left(\mathrm{x}'\right),\mathrm{H}\right)\left(\mathrm{K}\left(\mathrm{H},\mathrm{H}\right)+\sigma_N^2\mathrm{I}\right)^{-1}\mathrm{k}^{\mathrm{T}}\left(\mathrm{h}\left(\mathrm{x}'\right),\mathrm{H}\right). \tag{17}$$

Similarly, μ is the prediction of x' and σ^2 is the variance of prediction. Parameters σ_N^2 and λ^2 are unknown and need to be determined. 1HNBKM suffers severe over-fitting due to the usage of the RBF kernel and is also sensitive to random initialization.

3 Gaussian Process-based Stable ELM

In this section, we describe our proposed StaELM which conducts the inference based on Gaussian process and uses the correlation coefficient to construct the covariance matrix. StaELM also predicts the output y' for new instance x' based on Eq. (15), where

$$\mu = \mathrm{q}\left(\mathrm{h}\left(\mathrm{x}'\right),\mathrm{H}\right)\left(\mathrm{Q}\left(\mathrm{H},\mathrm{H}\right)+\sigma_N^2\mathrm{I}\right)^{-1}\mathrm{Y}, \tag{18}$$



(a) $L = 30$ (b) $L = 100$ (c) $L = 200$ (d) $L = 300$

Fig. 1. Comparison on Matlab computational time between kernel and correlation matrices in HNBKM and StaELM respectively

$$\sigma^2 = \sigma_N^2 + q\left(\mathrm{h}\left(\mathrm{x}'\right), \mathrm{h}\left(\mathrm{x}'\right)\right) - \mathrm{q}\left(\mathrm{h}\left(\mathrm{x}'\right), \mathrm{H}\right)\left(\mathrm{Q}\left(\mathrm{H}, \mathrm{H}\right) + \sigma_N^2 \mathrm{I}\right)^{-1} \mathrm{q}^{\mathrm{T}}\left(\mathrm{h}\left(\mathrm{x}'\right), \mathrm{H}\right), \quad (19)$$

$$\mathrm{Q}\left(\mathrm{H}, \mathrm{H}\right)_{N\times N} = \begin{bmatrix} q\left(\mathrm{h}\left(\mathrm{x}_1\right), \mathrm{h}\left(\mathrm{x}_1\right)\right) & q\left(\mathrm{h}\left(\mathrm{x}_1\right), \mathrm{h}\left(\mathrm{x}_2\right)\right) & \cdots & q\left(\mathrm{h}\left(\mathrm{x}_1\right), \mathrm{h}\left(\mathrm{x}_N\right)\right) \\ q\left(\mathrm{h}\left(\mathrm{x}_2\right), \mathrm{h}\left(\mathrm{x}_1\right)\right) & q\left(\mathrm{h}\left(\mathrm{x}_2\right), \mathrm{h}\left(\mathrm{x}_2\right)\right) & \cdots & q\left(\mathrm{h}\left(\mathrm{x}_2\right), \mathrm{h}\left(\mathrm{x}_N\right)\right) \\ \vdots & \vdots & \ddots & \vdots \\ q\left(\mathrm{h}\left(\mathrm{x}_N\right), \mathrm{h}\left(\mathrm{x}_1\right)\right) & q\left(\mathrm{h}\left(\mathrm{x}_N\right), \mathrm{h}\left(\mathrm{x}_2\right)\right) & \cdots & q\left(\mathrm{h}\left(\mathrm{x}_N\right), \mathrm{h}\left(\mathrm{x}_N\right)\right) \end{bmatrix} \quad (20)$$

is a correlation matrix,

$$\mathrm{q}\left(\mathrm{h}\left(\mathrm{x}'\right), \mathrm{H}\right) = \left[q\left(\mathrm{h}\left(\mathrm{x}'\right), \mathrm{h}\left(\mathrm{x}_1\right)\right) q\left(\mathrm{h}\left(\mathrm{x}'\right), \mathrm{h}\left(\mathrm{x}_2\right)\right) \cdots q\left(\mathrm{h}\left(\mathrm{x}'\right), \mathrm{h}\left(\mathrm{x}_N\right)\right)\right] \quad (21)$$

is a correlation vector,

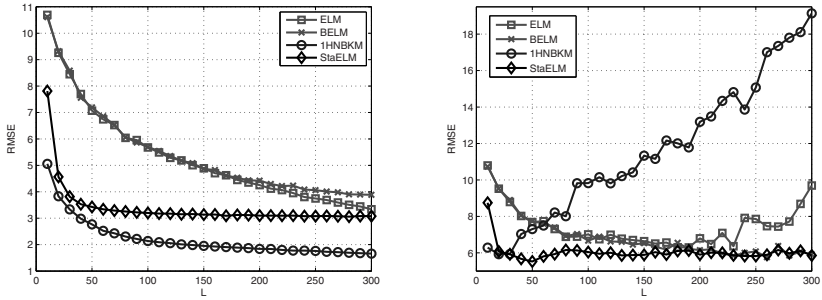
$$q\left(\mathrm{u}, \mathrm{v}\right) = \left(\frac{\rho_{\mathrm{uv}} + 1}{2}\right)^2 \quad (22)$$

is correlation function, and

$$\rho_{\mathrm{uv}} = \frac{Cov\left(\mathrm{u}, \mathrm{v}\right)}{\sqrt{D\left(\mathrm{u}\right)}\sqrt{D\left(\mathrm{v}\right)}} \quad (23)$$

is the correlation coefficient which measures the strength and direction of the linear relationship between two variables u and v. $Cov\left(\mathrm{u}, \mathrm{v}\right)$ is the covariance of variables u and v, and $D\left(\mathrm{u}\right)$ and $D\left(\mathrm{v}\right)$ are the standard deviations of u and v respectively. Note that Eq. (22) is to normalize the correlation coefficient into interval $(0, 1]$. Other normalization is also allowable. We can find that the inference process of StaELM is similar to 1HNBKM. The main difference between StaELM and 1HNBKM is that StaELM measures the similarity between two hidden layer outputs with correlation function in Eq. (22) instead of RBF kernel in 1HNBKM. The advantages of using correlation function are summarized as follows. (1) The correlation coefficient evaluates the relationship between two different hidden layer output $\mathrm{h}\left(\mathrm{u}\right)$ and $\mathrm{h}\left(\mathrm{v}\right)$ with probabilistic approach. This makes StaELM consider the inherent prior knowledge of training dataset more directly and comprehensively than kernel function based 1HNBKM. (2) The correlation coefficient reduce the chance of over-fitting of StaELM. For 1HNBKM with L hidden layer nodes, the prediction $\hat{\mathrm{Y}}$ for training dataset is

$$\hat{\mathrm{Y}} = \mathrm{K}\left(\mathrm{H}, \mathrm{H}\right)\left(\mathrm{K}\left(\mathrm{H}, \mathrm{H}\right) + \sigma_N^2 \mathrm{I}\right)^{-1} \mathrm{Y}. \quad (24)$$



(a) Training accuracy (b) Testing accuracy

Fig. 2. Curves for training and testing accuracies of different ELMs changing with L on *Concrete Compressive Strength* Dataset. StaELM is without over-fitting and obtains higher testing accuracies with less hidden nodes.

With the increase of L, $k(\mathrm{u},\mathrm{v}) \to 0$ when $\mathrm{u} \neq \mathrm{v}$. This leads to $\mathrm{K}(\mathrm{H},\mathrm{H}) \to \mathrm{I}$. Then, we can get $\hat{\mathrm{Y}} \to \mathrm{Y}$. This indicates that the RBF kernel easily results in the over-fitting of 1HNBKM. This is also confirmed by the following experimental validation. (3) Calculating the correlation matrix in Eq. (20) is more time-saving than kernel matrix in Eq. (12). We validate this fact via the following simulation on Matlab. For different L, we compare computational time of correlation matrix and kernel matrix. From Fig. 1, we can see that the computational time of kernel matrix grows exponentially with the increase of N.

StaELM is derived from Gaussian process regression (GPR) $y = \mathrm{h}(\mathrm{x})\beta + \varepsilon$ with prior $\beta \sim \mathrm{N}(0,\Sigma)$ and $\varepsilon \sim \mathrm{N}\left(0,\sigma_N^2\mathrm{I}\right)$ [11]. The mean and covariance are

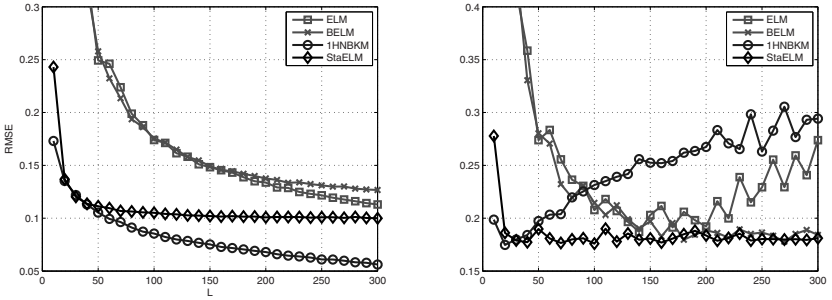
$$\mathrm{E}[y] = \mathrm{h}(\mathrm{x})\,\mathrm{E}[\beta] + \mathrm{E}[\varepsilon] = 0, \tag{25}$$

$$\mathrm{E}[yy'] = \mathrm{h}(\mathrm{x})\,\mathrm{E}\left[\beta\beta^{\mathrm{T}}\right]\mathrm{h}^{\mathrm{T}}(\mathrm{x}') + \mathrm{E}\left[\varepsilon\varepsilon^{\mathrm{T}}\right] = \mathrm{h}(\mathrm{x})\,\Sigma\mathrm{h}^{\mathrm{T}}(\mathrm{x}') + \sigma_N^2. \tag{26}$$

The key of GPR is how to determine the term $\mathrm{h}(\mathrm{x})\,\Sigma\mathrm{h}^{\mathrm{T}}(\mathrm{x}')$ in Eq. (26). Because Σ is a symmetric positive definite matrix, Σ can be decomposed into AA^{T}, where A is a lower triangular matrix. Then, we can get

$$\begin{aligned}\mathrm{h}(\mathrm{x})\,\Sigma\mathrm{h}^{\mathrm{T}}(\mathrm{x}') &= \mathrm{h}(\mathrm{x})\,\mathrm{AA}^{\mathrm{T}}\mathrm{h}^{\mathrm{T}}(\mathrm{x}') = [\mathrm{h}(\mathrm{x})\,\mathrm{A}]\,[\mathrm{h}(\mathrm{x}')\,\mathrm{A}]^{\mathrm{T}} \\ &= \phi(\mathrm{h}(\mathrm{x}))\,\phi^{\mathrm{T}}(\mathrm{h}(\mathrm{x}')) = k(\mathrm{h}(\mathrm{x}),\mathrm{h}(\mathrm{x}'))\end{aligned}, \tag{27}$$

where $k(\mathrm{u},\mathrm{v})$ is a kernel function which is used to measure the similarity between u and v. In StaELM, we replace $k(\mathrm{u},\mathrm{v})$ with $q(\mathrm{u},\mathrm{v})$ and use the correlation rather than distance to measure this similarity. In fact, we can find that the linear kernel $k(\mathrm{u},\mathrm{v}) = \mathrm{uv}^{\mathrm{T}}$ is used in ELM and BELM. Then, ELM, BELM, and 1HNBKM are all the specials cases of StaELM which conducts the prediction based on GPR.



(a) Training accuracy (b) Testing accuracy

Fig. 3. Curves for training and testing accuracies of different ELMs changing with L on *Treasury* Dataset

4 Experiments

In this section, we use 12 UCI and KEEL datasets to compare the performances of ELM, BELM, 1HNBKM, and StaELM, where 6 datasets are for regression problems and the other 6 datasets for classification problems. The basic descriptions to these datasets are listed in Tables 1 and 2 respectively. For the experimental procedure and parameter setting, we give the following descriptions.

- The input variables for regression datasets are normalized in $[-1, 1]$ and for classification datasets normalized in $[0, 1]$.
- We compare the training/testing accuracies and time for different learning algorithms. The accuracies for regression and classification problems are respectively measured with root mean square error (RMSE) and correct classification rate (CCR). The experimental results are the averages of 10 runs of 10-fold cross-validation.
- In our comparison, the parameters for different ELMs are set as $\left(\sigma_N^2, \alpha\right) = (0.001, 1)$ in BELM, $\left(\sigma_N^2, \lambda^2\right) = (0.001, 1)$ in 1HNBKM and $\sigma_N^2 = 0.001$ in StaELM respectively. The input weights and hidden biases are the random numbers in $[0, 1]$.

Tables 1 and 2 respectively give the comparative results on regression and classification datasets. According to the statistical analysis with Wilcoxon signed-ranks test at 95% significance level [7], we know that StaELM obtains the significantly better testing accuracies than other algorithms. Meanwhile, StaELM also has the better training accuracies than ELM and BELM. In addition, StaELM is also faster than 1HNBKM. On *Concrete Compressive Strength* and *Treasury* datasets, we give the curves for training and testing accuracies changing with the number of hidden layer nodes in Figs. 2 and 3. From these figures, we can clearly see that ELM and 1HNBKM have serious over-fitting problems. With the increase of L, the training RMSEs of ELM and 1HNBKM gradually decrease.

However, their testing RMSEs initially decrease with the increase of L, pass through a minimum, and then increase. Although the learning curves of BELM and StaELM all gradually decrease with the increase of L, we can find that StaELM has a faster convergence speed than BELM. This indicates that StaELM can obtain the lower RMSE with less hidden layer nodes.

The main reason that 1HNBKM has an obvious over-fitting is that the value of RBF kernel in Eq. (14) gradually approaches 0 with the increase of L. This leads to kernel matrix in Eq. (12) approximates an identity matrix. Assume the hidden layer outputs of instances u and v are

$$\mathrm{h}_L(\mathrm{u}) = \left[g(\hat{u}_1)\ g(\hat{u}_2) \cdots g(\hat{u}_L) \right], \tag{28}$$

$$\mathrm{h}_L(\mathrm{v}) = \left[g(\hat{v}_1)\ g(\hat{v}_2) \cdots g(\hat{v}_L) \right], \tag{29}$$

where $\hat{u}_l = \mathrm{w}_l \mathrm{u} + b_l$ and $\hat{v}_l = \mathrm{w}_l \mathrm{v} + b_l$, $l = 1, 2, \cdots, L$. With the increase of hidden layer nodes from L to L_1 ($L_1 > L$), the hidden layer outputs of instances u and v are changed into

$$\mathrm{h}_{L_1}(\mathrm{u}) = \left[g(\hat{u}_1) \cdots g(\hat{u}_L)\ g(\hat{u}_{L+1}) \cdots g(\hat{u}_{L_1}) \right], \tag{30}$$

$$\mathrm{h}_{L_1}(\mathrm{v}) = \left[g(\hat{v}_1) \cdots g(\hat{v}_L)\ g(\hat{v}_{L+1}) \cdots g(\hat{v}_{L_1}) \right]. \tag{31}$$

Then, we can calaulate

$$k\left(\mathrm{h}_L(\mathrm{u}), \mathrm{h}_L(\mathrm{v})\right) = \exp\left[-\frac{\sum_{l=1}^{L} \left[g(\hat{u}_l) - g(\hat{v}_l) \right]^2}{2\lambda^2} \right], \tag{32}$$

$$k\left(\mathrm{h}_{L_1}(\mathrm{u}), \mathrm{h}_{L_1}(\mathrm{v})\right) = \exp\left[-\frac{\sum_{l=1}^{L} \left[g(\hat{u}_l) - g(\hat{v}_l) \right]^2 + \sum_{l_1=L}^{L_1} \left[g(\hat{u}_{l_1}) - g(\hat{v}_{l_1}) \right]^2}{2\lambda^2} \right]. \tag{33}$$

.

Because of $\sum_{l=1}^{L} \left[g(\hat{u}_l) - g(\hat{v}_l) \right]^2 < \sum_{l=1}^{L} \left[g(\hat{u}_l) - g(\hat{v}_l) \right]^2 + \sum_{l_1=L}^{L_1} \left[g(\hat{u}_{l_1}) - g(\hat{v}_{l_1}) \right]^2$,

$$k\left(\mathrm{h}_L(\mathrm{u}), \mathrm{h}_L(\mathrm{v})\right) > k\left(\mathrm{h}_{L_1}(\mathrm{u}), \mathrm{h}_{L_1}(\mathrm{v})\right) \tag{34}$$

can be derived. This indicates that the value of $k(\mathrm{h}(\mathrm{u}), \mathrm{h}(\mathrm{v}))$ gradually decreases with the increase of hidden layer nodes. $k(\mathrm{h}(\mathrm{u}), \mathrm{h}(\mathrm{v}))$ is non-negative, so $k(\mathrm{h}(\mathrm{u}), \mathrm{h}(\mathrm{v})) \to 0$ with the increase of L. This leads to $\mathrm{K} \to \mathrm{I}$ and $\hat{\mathrm{Y}} \to \mathrm{Y}$.

For the correlation function in Eq. (22), there is not an ordering relationship between $q(\mathrm{h}_L(\mathrm{u}), \mathrm{h}_L(\mathrm{v}))$ and $q(\mathrm{h}_{L_1}(\mathrm{u}), \mathrm{h}_{L_1}(\mathrm{v}))$, because the correlation coefficient in Eq. (23) measures the similarly between $\mathrm{h}(\mathrm{u})$ and $\mathrm{h}(\mathrm{v})$ with vectorial angle cosine rather than distance between them. The increase of vector dimension will not cause the vectorial angle cosine approaches 0. Then, we can know that $\mathrm{Q} \nrightarrow \mathrm{I}$ with the increase of L. This reduces the chance of over-fitting.

Table 1. Comparison of different ELMs on 6 REGRESSION datasets

Dataset	L	ELM				BELM				1HNBKM				StaELM			
		TrainAcc	TrainTime	TestAcc	TestTime	TrainAcc	TrainTime	TestAcc	TestTime	TrainAcc	TrainTime	TestAcc	TestTime	TrainAcc	TrainTime	TestAcc	TestTime
Airfoil Self-Noise	20	4.507±0.349	0.00156	4.590±0.271	0.00000	4.507±0.349	0.01094	4.591±0.270	0.00000	2.549±0.055	15.10781	2.915±0.262	1.26875	2.832±0.066	1.53438	3.077±0.311	0.23125
(1503 instances,	30	4.052±0.130	0.00781	4.113±0.342	0.00000	4.052±0.130	0.02187	4.113±0.342	0.00000	2.363±0.063	19.97969	2.763±0.252	1.56719	2.696±0.073	1.94687	2.967±0.238	0.26875
5 inputs) UCI	100	3.055±0.099	0.04531	3.358±0.204	0.00313	3.149±0.078	0.04063	3.363±0.178	0.00000	1.905±0.042	23.82500	3.193±0.866	2.08125	2.531±0.045	1.98125	2.840±0.289•	0.28906
	200	2.433±0.051	0.14219	3.141±0.438	0.00469	2.834±0.034	0.16875	3.074±0.241	0.00625	1.588±0.022	34.43125	5.282±1.859	3.20000	2.468±0.028	2.02031	2.767±0.261•	0.37031
	300	2.337±0.045	0.22969	2.976±0.372	0.00937	2.738±0.034	0.29063	2.973±0.264	0.00000	1.384±0.021	36.05781	9.188±4.563	3.60312	2.481±0.028	1.82344	2.807±0.251•	0.35625
Concrete	20	9.477±0.531	0.00781	10.030±0.760	0.00000	9.477±0.531	0.00156	10.030±0.760	0.00000	3.365±0.120	9.82656	6.180±1.597	0.39844	4.686±0.210	1.85625	6.045±0.869•	0.18281
Compressive	30	8.458±0.312	0.00469	8.679±0.896	0.00156	8.458±0.312	0.00781	8.679±0.896	0.00000	2.829±0.145	9.66719	7.184±3.123	0.39531	3.770±0.168	1.83281	5.411±0.642•	0.19062
Strength	100	5.743±0.138	0.04531	7.064±0.647	0.00469	5.745±0.137	0.05312	7.021±0.642	0.00156	1.965±0.071	10.56563	10.910±4.743	0.51719	3.224±0.091	1.88906	5.992±1.931•	0.22812
(1030 instances,	200	4.234±0.085	0.21875	6.158±0.834	0.00469	4.374±0.068	0.22344	6.041±0.652	0.00156	1.633±0.097	13.00156	20.997±8.170	0.75469	3.067±0.096	1.94531	5.722±1.300•	0.30156
8 inputs) KEEL	300	3.338±0.108	0.56406	7.649±2.496	0.00156	3.838±0.093	0.63906	5.971±1.068	0.00156	1.420±0.108	15.64375	29.498±9.073	1.02969	3.075±0.054	1.99531	5.766±1.359•	0.35938
Energy Efficiency	20	2.865±0.093	0.00625	2.957±0.160	0.00000	2.865±0.093	0.00469	2.957±0.160	0.00000	0.502±0.050	4.11250	0.826±0.119	0.18906	0.823±0.068	0.75313	1.144±0.144	0.09062
(768 instances,	30	2.645±0.079	0.00156	2.825±0.164	0.00000	2.645±0.079	0.01406	2.825±0.164	0.00000	0.320±0.030	4.15937	0.654±0.071	0.18750	0.526±0.058	0.76719	0.876±0.139	0.09062
8 inputs) UCI	100	1.552±0.132	0.03750	1.929±0.196	0.00313	1.553±0.132	0.03906	1.916±0.195	0.00000	0.142±0.009	4.74062	0.965±0.179	0.24375	0.336±0.014	0.77969	0.639±0.083•	0.11719
	200	0.652±0.043	0.17500	1.101±0.230	0.00156	0.729±0.040	0.18594	1.103±0.139	0.00156	0.044±0.005	5.89375	2.101±0.573	0.37812	0.306±0.012	0.83906	0.620±0.074•	0.15781
	300	0.357±0.017	0.49844	0.795±0.089	0.00156	0.574±0.018	0.55312	0.912±0.121	0.00156	0.012±0.005	7.46094	2.087±0.473	0.52187	0.296±0.008	0.88594	0.622±0.074•	0.19687
Mortgage data	20	0.576±0.137	0.00469	0.575±0.116	0.00000	0.576±0.137	0.00781	0.575±0.116	0.00000	0.041±0.002	9.56406	0.062±0.008	0.39531	0.051±0.001	1.82969	0.075±0.020	0.18281
(1049 instances,	30	0.360±0.104	0.00937	0.376±0.105	0.00000	0.360±0.104	0.01250	0.376±0.105	0.00000	0.033±0.002	9.58594	0.063±0.008	0.40000	0.040±0.001	1.82656	0.067±0.013	0.17969
15 inputs) KEEL	100	0.082±0.012	0.04375	0.106±0.022	0.00156	0.082±0.012	0.05625	0.105±0.022	0.00156	0.019±0.001	10.60625	0.080±0.019	0.51875	0.031±0.001	1.89219	0.056±0.013•	0.22969
	200	0.045±0.002	0.20000	0.088±0.032	0.00625	0.049±0.002	0.22187	0.078±0.022	0.00625	0.013±0.001	12.93750	0.100±0.028	0.77031	0.030±0.001	1.95781	0.057±0.013•	0.29688
	300	0.033±0.001	0.57031	0.092±0.019	0.00781	0.041±0.001	0.64219	0.065±0.009	0.00313	0.011±0.000	16.06719	0.124±0.049	1.05469	0.030±0.001	2.08750	0.059±0.012•	0.36406
Stock Prices data	20	1.998±0.119	0.00625	2.014±0.192	0.00000	1.998±0.119	0.00937	2.014±0.192	0.00000	0.520±0.022	7.33594	0.757±0.048	0.31562	0.631±0.016	1.39063	0.835±0.060	0.13750
(950 instances,	30	1.425±0.071	0.00469	1.475±0.121	0.00000	1.425±0.071	0.01406	1.475±0.121	0.00000	0.432±0.015	7.36406	0.730±0.098	0.31719	0.505±0.010	1.41094	0.744±0.085	0.14531
9 inputs) KEEL	100	0.855±0.020	0.04688	0.971±0.056	0.00156	0.855±0.020	0.05312	0.969±0.057	0.00156	0.224±0.007	8.76719	0.895±0.136	0.42500	0.393±0.009	1.47656	0.685±0.078•	0.19844
	200	0.611±0.016	0.20781	0.827±0.041	0.00156	0.636±0.014	0.22344	0.808±0.041	0.00313	0.124±0.007	10.91094	1.110±0.075	0.64531	0.379±0.007	1.56094	0.681±0.065•	0.25625
	300	0.481±0.012	0.55781	0.860±0.085	0.00156	0.554±0.008	0.62187	0.778±0.053	0.00469	0.075±0.004	12.94531	1.251±0.291	0.86562	0.372±0.005	1.61094	0.678±0.059•	0.31094
Treasury data	20	0.569±0.084	0.00781	0.608±0.098	0.00000	0.569±0.084	0.00625	0.608±0.098	0.00000	0.135±0.003	10.43281	0.175±0.034	0.42500	0.137±0.004	1.94844	0.192±0.049	0.19375
(1049 instances,	30	0.408±0.086	0.00625	0.430±0.098	0.00000	0.408±0.086	0.01250	0.430±0.098	0.00000	0.121±0.004	10.65469	0.178±0.034	0.42656	0.120±0.005	1.94219	0.179±0.028	0.20313
15 inputs) KEEL	100	0.175±0.007	0.04688	0.217±0.028	0.00000	0.175±0.007	0.05156	0.216±0.029	0.00469	0.086±0.005	11.43906	0.225±0.058	0.54375	0.104±0.003	1.93906	0.180±0.033•	0.23750
	200	0.133±0.005	0.21094	0.204±0.031	0.00000	0.138±0.004	0.22812	0.187±0.030	0.00469	0.067±0.004	13.59844	0.272±0.080	0.80625	0.101±0.003	1.99844	0.178±0.028•	0.30938
	300	0.113±0.004	0.58750	0.291±0.189	0.00469	0.126±0.004	0.63594	0.194±0.041	0.00313	0.056±0.003	16.32500	0.293±0.060	1.06563	0.101±0.003	2.08906	0.177±0.034•	0.38594

Note: • indicates that the testing accuracy of StaELM is significantly better than the corresponding algorithm based on Wilcoxon signed-ranks test at 95% significance level.

Table 2. Comparison of different ELMs on 6 CLASSIFICATION datasets

Dataset	L	ELM				BELM				1HNBKM				StaELM			
		TrainAcc	TrainTime	TestAcc	TestTime	TrainAcc	TrainTime	TestAcc	TestTime	TrainAcc	TrainTime	TestAcc	TestTime	TrainAcc	TrainTime	TestAcc	TestTime
Automobile	20	0.701±0.023	0.00313	0.534±0.113	0.00000	0.699±0.021	0.00000	0.534±0.113	0.00000	0.843±0.018	0.07500	0.648±0.077	0.00625	0.995±0.006	0.01563	0.667±0.066•	0.00156
(159 instances,	30	0.767±0.022	0.00156	0.559±0.092	0.00313	0.762±0.026	0.00937	0.572±0.104	0.00000	0.878±0.018	0.06875	0.679±0.114	0.00469	1.000±0.000	0.02031	0.717±0.118•	0.00156
15 inputs,	100	0.993±0.007	0.02187	0.597±0.135	0.00000	0.936±0.016	0.02500	0.679±0.095	0.00000	0.976±0.007	0.08438	0.711±0.052	0.00781	1.000±0.000	0.01719	0.723±0.129•	0.00156
6 classes) KEEL	200	1.000±0.000	0.06563	0.573±0.152	0.00156	0.978±0.009	0.11875	0.666±0.139	0.00000	0.994±0.004	0.12188	0.710±0.123	0.01094	1.000±0.000	0.01719	0.735±0.164•	0.00313
	300	1.000±0.000	0.06250	0.565±0.110	0.00000	0.987±0.005	0.37812	0.660±0.129	0.00156	0.998±0.003	0.15937	0.691±0.112	0.01563	1.000±0.000	0.02656	0.716±0.110•	0.00625
Ecoli	20	0.881±0.010	0.00469	0.872±0.051	0.00000	0.881±0.009	0.00313	0.869±0.050	0.00000	0.897±0.007	0.40937	0.866±0.048	0.03125	0.899±0.009	0.07500	0.869±0.049	0.01094
(336 instances,	30	0.893±0.009	0.00781	0.851±0.071	0.00000	0.886±0.009	0.00313	0.857±0.069	0.00000	0.899±0.007	0.42969	0.857±0.069	0.03281	0.894±0.007	0.07031	0.860±0.065•	0.01406
5 inputs,	100	0.918±0.005	0.02813	0.831±0.052	0.00000	0.895±0.005	0.02656	0.848±0.057	0.00000	0.909±0.004	0.51719	0.846±0.047	0.04531	0.896±0.004	0.07969	0.860±0.037•	0.02031
8 classes) UCI	200	0.917±0.004	0.11563	0.839±0.049	0.00000	0.899±0.007	0.14688	0.860±0.056	0.00000	0.922±0.007	0.67500	0.851±0.054	0.06406	0.897±0.009	0.09219	0.863±0.056•	0.02969
	300	0.917±0.007	0.34531	0.842±0.046	0.00156	0.898±0.005	0.45156	0.854±0.037	0.00000	0.928±0.009	0.88594	0.830±0.028	0.07969	0.895±0.005	0.10469	0.863±0.041•	0.03750
Glass Identification	20	0.719±0.022	0.00313	0.651±0.086	0.00000	0.721±0.021	0.00313	0.655±0.081	0.00000	0.807±0.015	0.12188	0.673±0.078	0.01094	0.818±0.016	0.02187	0.678±0.062•	0.00781
(214 instances,	30	0.766±0.010	0.00313	0.660±0.094	0.00000	0.763±0.013	0.00781	0.660±0.092	0.00000	0.822±0.014	0.12812	0.692±0.077	0.01406	0.821±0.019	0.01875	0.702±0.093•	0.00469
9 inputs,	100	0.935±0.009	0.02813	0.616±0.082	0.00000	0.813±0.020	0.02344	0.644±0.107	0.00000	0.889±0.017	0.17031	0.639±0.113	0.01719	0.820±0.022	0.02969	0.650±0.139•	0.00625
7 classes) UCI	200	0.949±0.010	0.08906	0.635±0.126	0.00000	0.827±0.010	0.12188	0.673±0.073	0.00469	0.938±0.012	0.22031	0.673±0.083	0.02187	0.828±0.017	0.03438	0.678±0.062•	0.01250
	300	0.951±0.013	0.09531	0.642±0.127	0.00156	0.836±0.009	0.39531	0.682±0.066	0.00156	0.952±0.007	0.30156	0.655±0.103	0.02969	0.825±0.016	0.04219	0.682±0.054	0.01406
Phoneme (10%)	20	0.794±0.009	0.00156	0.766±0.056	0.00000	0.792±0.010	0.00313	0.768±0.060	0.00000	0.820±0.007	1.61406	0.781±0.040	0.07969	0.838±0.009	0.27969	0.815±0.032•	0.03750
(5404 instances,	30	0.811±0.011	0.00313	0.740±0.056	0.00000	0.804±0.010	0.00625	0.774±0.033	0.00000	0.839±0.011	1.65781	0.789±0.057	0.08594	0.840±0.011	0.28750	0.802±0.038•	0.03750
5 inputs,	100	0.878±0.009	0.02969	0.829±0.028	0.00000	0.817±0.012	0.03750	0.781±0.053	0.00000	0.877±0.008	1.87344	0.824±0.038	0.11250	0.874±0.005	0.29375	0.831±0.039•	0.05000
2 classes) KEEL	200	0.875±0.005	0.12812	0.829±0.058	0.00156	0.825±0.009	0.16719	0.796±0.034	0.00000	0.891±0.006	2.37031	0.826±0.049	0.17656	0.876±0.004	0.31406	0.833±0.059•	0.07656
	300	0.877±0.006	0.37188	0.824±0.044	0.00156	0.833±0.007	0.48594	0.783±0.048	0.00313	0.906±0.005	3.05000	0.822±0.036	0.24375	0.878±0.008	0.33594	0.829±0.041•	0.09375
Vehicle Silhouettes	20	0.668±0.016	0.00469	0.637±0.040	0.00156	0.668±0.016	0.00313	0.638±0.041	0.00156	0.828±0.008	5.44219	0.745±0.022	0.23281	0.875±0.009	1.02031	0.779±0.038•	0.10312
(846 instances,	30	0.728±0.015	0.00313	0.694±0.082	0.00000	0.727±0.014	0.00313	0.694±0.077	0.00000	0.855±0.009	5.59375	0.779±0.049	0.24375	0.892±0.008	1.02813	0.786±0.054•	0.11094
8 inputs,	100	0.853±0.009	0.04375	0.765±0.062	0.00000	0.822±0.009	0.04844	0.753±0.035	0.00000	0.901±0.008	6.15000	0.792±0.053	0.31875	0.906±0.007	1.05469	0.797±0.043•	0.14219
4 classes) KEEL	200	0.893±0.011	0.17813	0.783±0.043	0.00156	0.848±0.010	0.20469	0.764±0.046	0.00156	0.933±0.004	8.15781	0.786±0.030	0.49531	0.910±0.007	1.11250	0.800±0.032•	0.18906
	300	0.893±0.005	0.46250	0.787±0.036	0.00313	0.860±0.010	0.55469	0.767±0.036	0.00313	0.956±0.003	9.70937	0.770±0.036	0.65625	0.910±0.009	1.13750	0.794±0.041•	0.22969
Vowel Recognition-	20	0.657±0.023	0.00313	0.587±0.078	0.00000	0.657±0.023	0.00313	0.586±0.072	0.00000	0.858±0.020	1.48125	0.773±0.039	0.07813	0.991±0.006	0.26406	0.926±0.036•	0.03438
Deterding	30	0.747±0.021	0.00313	0.651±0.066	0.00000	0.747±0.020	0.00781	0.653±0.067	0.00000	0.906±0.010	1.53125	0.829±0.043	0.08281	0.998±0.004	0.27500	0.953±0.030•	0.03438
(528 instances,	100	0.967±0.009	0.03281	0.898±0.044	0.00000	0.933±0.009	0.03750	0.868±0.045	0.00000	0.981±0.005	1.81719	0.934±0.037	0.10781	1.000±0.000	0.29531	0.962±0.024•	0.05000
10 inputs,	200	0.999±0.001	0.14688	0.958±0.019	0.00156	0.965±0.007	0.16250	0.905±0.033	0.00156	0.995±0.003	2.21875	0.951±0.025	0.16094	1.000±0.000	0.30156	0.975±0.018•	0.06875
11 classes) UCI	300	1.000±0.000	0.42969	0.947±0.049	0.00000	0.977±0.004	0.47813	0.924±0.046	0.00156	0.999±0.001	3.00312	0.951±0.048	0.23906	1.000±0.000	0.33594	0.956±0.044•	0.09062

Note: Only continues inputs are used for every dataset. For *Phoneme* dataset, 10% of 5404 instances are randomly selected.

5 Conclusion

In this paper, we have presented a stable ELM (StaELM) by using correlation coefficient to measure the similarity between different hidden layer outputs. We have further analysed the rationality of StaELM in the framework of Gaussian process regression. Compared with the kernel-based methods, StaELM obviously reduces the chance of singular covariance matrix and make ELM more stable to random initialization of input weights and hidden biases. In addition, our improvement does not cause the significant increase of computational complexity.

Acknowledgments. This work was supported by the National Natural Science Foundations of China under Grants 61170040, 71371063, and 61473194.

References

1. Cao, J., Lin, Z., Huang, G.B.: Self-Adaptive Evolutionary Extreme Learning Machine. Neural Process. Lett. **36**(3), 285–305 (2012)
2. Chatzis, S.P., Korkinof, D., Demiris, Y.: The one-hidden layer non-parametric bayesian kernel machine. In: 23rd IEEE International Conference on Tools with Artificial Intelligence, pp. 825–831. IEEE Press, New York (2011)
3. Huang, G.B., Chen, L., Siew, C.K.: Universal Approximation Using Incremental Constructive Feedforward Networks with Random Hidden Nodes. IEEE Trans. Neural Netw. **17**(4), 879–892 (2006)
4. Huang, G.B., Li, M.B., Chen, L., Siew, C.K.: Incremental Extreme Learning Machine with Fully Complex Hidden Nodes. Neurocomputing **71**(4–6), 576–583 (2008)
5. Huang, G.B., Wang, D.H., Lan, Y.: Extreme Learning Machines: A Survey. Int. J. Mach. Learn. & Cybern. **2**(2), 107–122 (2011)
6. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme Learning Machine: Theory and Applications. Neurocomputing **70**(1), 489–501 (2006)
7. Janez, D.: Statistical Comparisons of Classifiers over Multiple Data Sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
8. Luo, J.H., Vong, C.M., Wong, P.K.: Sparse Bayesian Extreme Learning Machine for Multi-Classification. IEEE Trans. Neural Netw. Learn. Syst. **25**(4), 836–843 (2014)
9. Matias, T., Souza, F., Araújo, R., Antunes, C.H.: Learning of A Single-Hidden Layer Feedforward Neural Network Using An Optimized Extreme Learning Machine. Neurocomputing **129**, 428–436 (2014)
10. Miche, Y., Sorjamaa, A., Bas, P., Simula, O., Jutten, C., Lendasse, A.: OP-ELM: optimally pruned extreme learning machine. IEEE Trans. Neural Netw. **21**(1), 158–162 (2010)
11. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press, Cambridge (2006)
12. Soria-Olivas, E., Gomez-Sanchis, J., Jarman, I.H., Vila-Frances, J.: BELM: Bayesian Extreme Learning Machine. IEEE Trans. Neural Netw. **22**(3), 505–509 (2011)
13. Wang, X.Z., Shao, Q.Y., Miao, Q., Zhai, J.H.: Architecture Selection for Networks Trained with Extreme Learning Machine Using Localized Generalization Error Model. Neurocomputing **102**, 3–9 (2013)

14. Wong, K.I., Vong, C.M., Wong, P.K., Luo, J.H.: Sparse Bayesian extreme learning machine and its application to biofuel engine performance prediction. Neurocomputing **149**, 397–404 (2015)
15. Zhu, Q.Y., Qin, A., Suganthan, P., Huang, G.B.: Evolutionary Extreme Learning Machine. Pattern Recogn. **38**(10), 1759–1763 (2005)