Economic Modelling xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Economic Modelling

journal homepage: www.elsevier.com/locate/econmod

Calculating Value-at-Risk for high-dimensional time series using a nonlinear random mapping model

Heng-Guo Zhang[a,b], Chi-Wei Su[a], Yan Song[c], Shuqi Qiu[c], Ran Xiao[d], Fei Su[d,*]

[a] Department of Finance, Ocean University of China, Qingdao, Shandong, China
[b] Department of Finance and Economics, Shandong University of Science and Technology, Jinan, Shandong, China
[c] College of Information Science and Engineering, Ocean University of China, Qingdao, Shandong, China
[d] Finance Discipline Group, UTS Business School, University of Technology, Sydney, Australia

ARTICLE INFO

JEL:
C32
C45
C53

Keywords:
Extreme learning machine
High-dimensional space
Value-at-Risk
Random mapping
GARCH model
Time series

ABSTRACT

In this study, we propose a non-linear random mapping model called GELM. The proposed model is based on a combination of the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model and the Extreme Learning Machine (ELM), and can be used to calculate Value-at-Risk (VaR). Alternatively, the GELM model is a non-parametric GARCH-type model. Compared with conventional models, such as the GARCH models, ELM, and Support Vector Machine (SVM), the computational results confirm that the GELM model performs better in volatility forecasting and VaR calculation in terms of efficiency and accuracy. Thus, the GELM model can be an essential tool for risk management and stress testing.

1. Introduction

There has been intensive research on risk modeling to develop more sophisticated risk management techniques and models. Following the increasing uncertainty in financial markets and the global financial crises since the 1990s, it is of great interest to practitioners, regulators, and academic researchers (Angelidis et al., 2004). Value-at-Risk (VaR) has emerged as a suitable and remarkable tool to quantify risk, and became popular and prevalent during the 1990s because of its simplicity and easy implementation (Giot and Laurent, 2004). VaR proposed by Jorion (1995) is defined as the worst expected loss over a given horizon at different levels of confidence. Even though conceptually simple and usable, VaR estimations have become more complicated and sophisticated during the last decade.

*Overall, the VaR estimation methods can be categorized into three major types (*Kim and Lee, 2016*).* The first one is the parametric method, wherein a return process is generated with error terms that follow a specific distribution. A popular example is the Generalized Autoregressive Conditionally Heteroskedastic (GARCH) model. The GARCH model was first proposed by Bollerslev (1986), which uses historical variance to predict future volatility under the assumption of conditional heteroskedasticity. A number of extensions to the GARCH model have been proposed and applied to financial markets, including the integrated GARCH (IGARCH) model introduced by Engle and Bollerslev (1986), the exponential GARCH (EGARCH) model proposed by Nelson (1991), etc. In this study, we extend the literature on conditional heteroskedasticity to encompass a broader class of GARCH processes, namely, a non-parametric GARCH-type model. A detailed introduction to GARCH-type models is given in Section II.

The second type is the semi-parametric method. It applies either the quantile regression approach proposed by Koenker and Bassett (1978) or extreme value distribution-based methods, for example, Extreme Value Theory (EVT) introduced by Smith (1989). The EVT is well known for modeling extreme tails by analyzing the upper and lower quantiles of the corresponding distribution.[1] For example, Wang et al. (2012) use the composite quantile regression (CQR) method as proposed in Zou and Yuan (2008), to estimate the intermediate quantiles, and then extrapolated these estimates to calculate the extreme quantiles using the EVT.

The third type includes the non-parametric method, which is especially useful for analyzing big data, as in the case of machine learning. An artificial neural network (ANN) is one of the most

* Corresponding author.
E-mail address: fei.su@uts.edu.au (F. Su).

[1] For a thorough introduction to EVT, see Smith (1989) and McNeil (1997), as well as the references therein.

http://dx.doi.org/10.1016/j.econmod.2017.02.014
Received 21 July 2016; Received in revised form 7 January 2017; Accepted 15 February 2017

primitive machine learning frameworks, motivated by the learning capabilities of the human brain, and capable of performing parallel computation for time series forecasting. There are many ANN variations, such as back propagation (BP) networks introduced by Williams and Hinton (1986), and the radial basis function (RBF) proposed by Lowe (1989), among many others. However, bottlenecks in areas such as overfitting, local minima, and computation time, etc, can restrict the scalability of these models in conventional implementations. Cortes and Vapnik (1995) put forward the statistical learning theory and develop the Support Vector Machine (SVM) based on the structural risk minimization (SRM) principle, which seeks to minimize an upper bound on the Vapnik Chervonenkis (VC) dimension of the generalization error. Beyond these models, deep learning has become a popular topic since Hinton and Salakhutdinov (2006) proposed the deep belief networks (DBN). A DBN can handle complex, high-dimensional time series by exploiting multilayer hierarchical neural network architectures. In addition, convolutional neural networks (CNNs) introduced by LeCun et al. (1989) are a family of multilayer neural networks that automatically learn hierarchical features, and therefore extract complex translation and distortion invariant features in higher layers. A brief introduction to these models of relevance is presented in Section II as well. One of the most important advantages of these machine learning models is that they are data driven. In other words, the models can be reasonably estimated with no prior assumptions on the nature of the error distribution.

In the era of big data, how to model and calculate the VaR for high-dimensional time series becomes a challenging topic. Berman (2013) finds that the unique features of big data are high dimensionality, complexity, massiveness, heterogeneity, incompleteness, noisiness, and erroneousness. Thus, conventional approaches to measuring risks face major challenges in handling big data. For example, most conventional methods used to calculate the VaR were originally designed for relatively small data sets, and may require additional assumptions on the distribution of the error terms (e.g., a normal distribution assumption). In addition, Chen and Zhang (2014) argue that when analyzing big data, it is essential to speed up the estimation procedure, especially for conventional approaches, in order to reduce the computation time and memory requirements.

Recently, the extreme learning machine (ELM) has attracted increasing attention. It provides a greater generalization performance with a faster learning speed. The ELM proposed by Huang et al. (2006) was originally developed for single-hidden layer feedforward networks (SLFN), rather than using the classic gradient-based algorithms, which has been extended to the generalized SLFN. In general, Huang et al. (2012) deem that the essence of ELM is that when the input weights and hidden layer biases are randomly assigned, the output weights can be computed using the generalized inverse of the hidden layer output matrix. In this study, motivated by the remarkable success of ELM in terms of generalization performance and learning speed, we propose a non-linear random mapping model called GELM, which is based on a combination of the conditional heteroskedasticity processes and ELMs. More specifically, we apply an ELM algorithm to high-dimensional time series of GARCH processes, and thereby extend the multivariate GARCH process in a non-parametric framework.

The main characteristics of the GELM model are as follows. Firstly, compared with parametric and semi-parametric models, the GELM model utilizes a random mapping method, which does not employ the Gaussian likelihood to estimate the parameters. Specifically, the GELM randomly generates the hidden node parameters, for which it keeps the same virtues of the random parameters as in the ELM model. Once the input weights and the hidden layer biases are randomly assigned, the output weights of the GELM model can be calculated analytically using the simple generalized inverse operation of the hidden layer output matrix.

The second characteristic lies in the fact that the proposed GELM model is a non-linear data-driven model. More specifically, the GELM model learns and approximates the dynamics of time series in a non-linear fashion directly from the data, with no prior assumptions. That is, in the GELM framework, it is not to assume that the conditional dependencies are all contained in the conditional mean and variance, and the variables are not necessarily independent over time. For example, the model functions well, even when the stationary and ergodic conditions are not satisfied. Thus, the GELM model is well suited for complex problems with little prior knowledge, but with a large number of observations.

Thirdly, the GELM model is more noise tolerant with regard to connections between the system state variables. In other words, in the learning process with the given data, the GELM model can always correctly infer the hidden part of the time series, even if the sample contains noisy information. It can be estimated under weak or even no prior assumptions on the error distribution. In addition, the model can learn and estimate complex systems with incomplete, non-linear, and non-stationary data structures. Thus, it can handle big data sets in high-dimensional spaces.

The rest of the paper is organized as follows. In Section 2, previous works on GARCH-type models and feedforward neural networks are reviewed. The basic ELM model is introduced in Section 3. Section 4 proposes the GELM model and Section 5 reports on the experimental results, and discusses related issues. Lastly, Section 6 concludes the paper, with possible directions for future research.

2. Literature review

Traditional time series tools for the conditional means such as autoregressive moving average (ARMA) model introduced by Box and Jenkins (1976) have been extended to analogous models for higher moments of time series. Autoregressive Conditional Heteroscedasticity (ARCH) models are commonly used to estimate and forecast changes in the second moment of financial time series.[2] Since the seminal work of Engle (2002), much progress has been made in understanding GARCH models and their multivariate extensions.

1) *The univariate GARCH.* Given an univariate GARCH model on return series, we can infer the conditional distribution of the return, and thereby calculate the VaR of a long or short position. The univariate GARCH type models can be classified into four types.

The first type is the basic GARCH model proposed by Bollerslev (1986), which can be written as:

$$\sigma_t^2 = c + \sum_{i=1}^{n} \alpha_i \xi_{t-i}^2 + \sum_{i=1}^{n} \gamma_i \sigma_{t-i}^2 \tag{1}$$

where σ_t^2 denotes the conditional variance, α_i and γ_i are non-negative ARCH and GARCH coefficients respectively, and ξ_t is the random error. The integrated GARCH model proposed by Engle and Bollerslev (1986) is a special case of the GARCH model, where the sum of the ARCH and GARCH coefficients equals one:

$$\begin{cases} \sigma_t^2 = \sum_{i=1}^{n} \alpha_i \xi_{t-i}^2 + \sum_{i=1}^{n} \gamma_i \sigma_{t-i}^2 \\ \sum_{i=1}^{n} \alpha_i + \sum_{i=1}^{n} \gamma_i = 1 \end{cases} \tag{2}$$

The second type relaxes the assumption of symmetry in the conditional variance specification. For example, the most commonly adopted asymmetric GARCH model is the GJR model developed by Glosten et al. (1993):

[2] For a survey of ARCH-type models, please see Bollerslev et al. (1992), Bera and Higgins (1993), Pagan (1996), among many others.

$$\sigma_t^2 = c + \gamma\sigma_{t-1}^2 + \alpha\xi_{t-1}^2 + \mu\zeta_{t-1}^2 \tag{3}$$

where $\zeta_t = \min(0, \zeta_t)$, and μ proxies for the asymmetric effect. In addition, a bunch of analogous models have been proposed, such as the exponential GARCH (EGARCH) model developed by Nelson (1991) and the asymmetric power ARCH (APARCH) model proposed by Ding et al. (1993), etc.

In contrast to the first and second type, the third type characterizes the long-run dependence, such as the fractionally integrated GARCH (FIGARCH) model proposed by Baillie et al. (1996):

$$\sigma_t^2 = c[1-\gamma(\eta)]^{-1} + \{1 - [1-\gamma(\eta)]^{-1}\phi(\eta)(1-\eta)^g\}\xi_t^2 \tag{4}$$

where g is the fractional integration parameter, and η is the lag operator. Beyond that, FIGARCH-BBM developed by Bollerslev and Mikkelsen (1996), FIGARCH-CHUNG proposed by Chung (1999) and the hyperbolic GARCH (HYGARCH) model introduced by Davidson (2004) also belongs to this type.[3]

Finally, the fourth type of GARCH model is associated with the combining stylized features of long memory and asymmetric volatility. For instance, Tse (1998) extends the APARCH model to a process that is also fractionally integrated and propose the FIAPARCH model specification as follows:

$$\sigma_t^\nu = c[1-\gamma(\eta)]^{-1} + \{1 - [1-\gamma(\eta)]^{-1}\phi(\eta)(1-\eta)^g\}(|\xi_t| - \eta\xi_t)^\nu \tag{5}$$

As we can see, the FIGARCH model can be seen as a special case of the FIAPARCH model when $\eta = 0$ and $\nu = 2$.

2) *The multivariate GARCH*. Multivariate GARCH (MGARCH) models were initially developed in the late 1980s and the first half of 1990s. The multivariate approach was thoroughly illustrated by Giot and Laurent (2004) using a trivariate example with time-varying correlations. MGARCH models were also partly covered in Franses and Van Dijk (2000), as well as in most of the surveys on GARCH models mentioned above.[4]
3) *Feedforward neural networks*. Many types of neural networks have been proposed to overcome the defects of conventional GARCH-type models introduced above, and feedforward neural networks stands out as one of the most popular methods. A feedforward neural network consists of one input layer that receives stimuli from the external environment, one or more hidden layers, and one output layer that delivers the network output to the external environment. Three main approaches are usually used in training feedforward networks.

Firstly, the gradient-descent-based approach, e.g., the backpropagation (BP) method developed by Williams and Hinton (1986), is used for multi-layer feedforward neural networks. Additive type hidden nodes are mostly used in such networks. For an additive hidden node with activation function $g(z)$: $\Re \to \Re$, the output function of the ith node in the oth hidden layer is then given by:

$$G(a_i^o, b_i^o, z^o) = g(a_i^o \cdot z^o + b_i^o), \quad b_i^o \in R \tag{6}$$

where a_i^o is the weights vector connecting the $(o-1)$th layer to the ith node of the oth layer, b_i^o is the bias of the ith node of the oth layer. And, $a_i^o \cdot z^o$ denotes the inner product of vectors a_i^o and z^o. However, Huang et al. (2011) find that gradient-descent-based learning algorithms usually perform much slower than expected.

The second approach is based on the standard optimization method (e.g., SVM). For this method, a linear decision function is usually constructed:

$$f(z) = sign\left(\sum_{i=1}^{V}\beta_i w_i(z)\right) \tag{7}$$

where β_i is the output weight between the output node and the ith neuron in the last hidden layer of a perceptron, and $w_i(z)$ is the output of the ith neuron in the corresponding layer. In order to find an appropriate solution to $w_i(z)$, Cortes and Vapnik (1995) propose the Support Vector Machine (SVM) which projects data from the input space to some high-dimensional feature spaces through non-linear mapping.

The third approach is based on the least squares method, like radial basis function (RBF) network learning proposed by Lowe (1989). For an RBF hidden node with an activation function (e.g.,$g(z) = \exp(-z^2)$), $G(a_i, b_i, z)$ is then given by

$$G(a_i, b_i, z) = g(b_i\|z - a_i\|), \quad b_i \in R^+ \tag{8}$$

where R^+ indicates the set of all positive real values. The center factor a_i of the RBF hidden nodes can be randomly selected from the training data or from a region of training data. Furthermore, the impact factors b_i of the RBF hidden nodes are usually set to the same values. Huang et al. (2012) find that after the RBF hidden node parameters (a_i, b_i) are selected, the output weight vector linking the RBF hidden nodes to the output layer will be the only unknown parameter, which can be resolved using the least-squares method.

3. Basic extreme learning machine

Given a data set containing N samples, the training samples can be represented by (z_i, y_i), $i = 1, 2, \ldots, N$, with m classes, where $z_i = [z_{i1}, z_{i2}, \ldots, z_{in}]^T$ is the vector of input variables and $y_i = [y_{i1}, y_{i2}, \ldots, y_{im}]^T$ is the output vector. The single-hidden layer feedforward neural network (SLFN) with V hidden nodes and the activation function $f(z)$ is mathematically modeled as:

$$\sum_{i=1}^{V}\beta_i f(a_i \cdot z_j + b_i) = y_j, \; j = 1, \ldots, N \tag{9}$$

Thus, the basic ELM algorithm introduced by Huang et al. (2006) can be written concisely in the following matrix format:

$$H\beta = Y \tag{10}$$

where

$$H(a_1, \ldots, a_V, b_1, \ldots, b_V, z_1, \ldots, z_V) = \begin{bmatrix} G(a_1 \cdot z_1 + b_1) \ldots & G(a_V \cdot z_1 + b_V) \\ \vdots & \vdots \\ G(a_1 \cdot z_N + b_1) \ldots & G(a_V \cdot z_N + b_V) \end{bmatrix}_{N \times V}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_V^T \end{bmatrix}_{V \times m} \quad and \quad Y = \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix}_{N \times m} \tag{11}$$

and $a_i = [a_{i1}, a_{i2}, \ldots, a_{in}]^T$, $i = 1, 2, \ldots, V$ is the weight vector connecting the ith hidden neuron and the input neurons, and b_i, $i = 1, 2, \ldots, V$ is the threshold value of the ith hidden neuron. Then, $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$, $i = 1, 2, \ldots, V$ denote the weight vector which connects the ith hidden neuron and the output neurons. There are V hidden neurons with the activation function. Various activation functions can be used here, including the sigmoid function, the hard-limit function, the Gaussian function, and the multi-quadric function, etc. As in Huang et al. (2006), H is called the hidden layer output matrix of the neural network, and the ith column of H stands for the ith hidden node output with respect to inputs $z_1, z_2, \ldots, z_N$. A typical implementation of ELMs is to apply randomly computational nodes in the hidden layer, which may be independent of the training data. For ELM to reach the smallest training errors, the principal is that the smaller the norm of the weights is, the better the generalization performance the networks can provide. Because in the ELM the hidden layer need not be tuned

[3] For a detailed introduction to the integrated GARCH model, please refer to Christensen et al. (2010) and Arouri et al. (2012).

[4] For an introduction to the multivariate GARCH models, please refer to Bauwens et al. (2006).

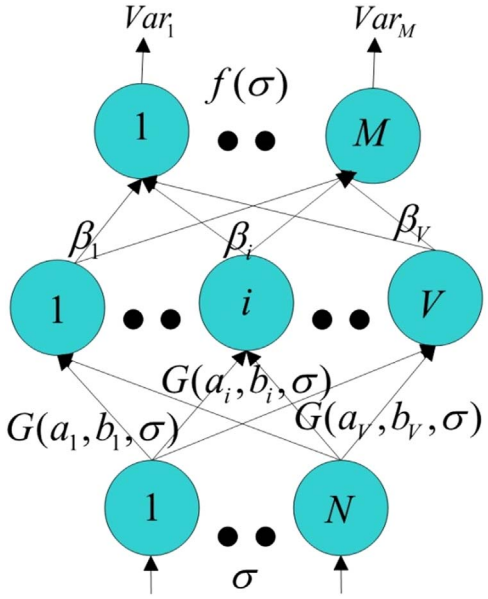

Fig. 1. Flowchart of GELM model.

and the hidden layer parameters can be fixed, one can resolve the output weights using the lease-squares method. Thus, the output weights β are calculated as:

$$\hat{\beta} = H^{\dagger}Y \tag{12}$$

where $H^{\dagger}$ is the Moore-Penrose generalized inverse of matrix H.

4. GARCH process via ELM algorithm

The basic idea of GELM model is that we calculate Value-at-Risk under the framework of GARCH process and simulate the whole process in a much faster way. Namely, in the spirit of GARCH model which makes use of the lagged variance to predict the variance of future, the GELM model utilize GARCH process and ELM algorithm to estimate the future volatility and calculate Value-at-Risk in a timely and efficient manner.

4.1. General model

The flow chart of our method is shown in Fig. 1. Suppose that the time series $z_i(t)$, $i = 1...N$, $t = 1...T$ belongs to the training data set $Z_t = \{z_1(t)... z_i(t)... z_N(t)\}$ for ELM learning. Correspondingly, we can calculate the standard deviation $\sigma_i(t)$ of the time series $z_i(t)$ by the following equation:

$$\sigma_{z_i(t)} = sqrt\{[(z_i(1) - \bar{z}_i)^2 + ...+(z_i(T) - \bar{z}_i)^2]/T\} \tag{13}$$

where $\bar{z}_i$ is the mean value of $z_i(t)$. Similarly, one can calculate the standard deviation of the training data set as $\sigma_{Z_t} = \{\sigma_{z_1(t)}... .\sigma_{z_i(t)}...\sigma_{z_N(t)}\}$. Furthermore, let the time series $y_j(t)$, $j = 1...M$, $t = 1...T$ belong to the target matrix of the training data $Y_t = \{y_1(t)... y_j(t)... y_M(t)\}$, and the standard deviation of the target matrix is calculated as $\sigma_{Y_t} = \{\sigma_{y_1(t)}....\sigma_{y_j(t)}... \sigma_{y_M(t)}\}$. Once a set of SLFNs is reasonably established via ELM leaning in the spirit of GARCH process, the standard deviations of the high-dimensional time series can then be fed into the SLFN as the training data. The output of the training process of GELM is $f(\sigma_{Z_t}) = \sigma_{O_t} = \{\sigma_{o1(t)}....\sigma_{oj(t)}... \sigma_{oM(t)}\}$, which can then be employed to calculate VaR.

The output function of SLFNs with V hidden nodes can be presented as

$$\sigma_{O_t} = f_V(\sigma_{Z_t}) = \sum_{i=1}^{V}\beta_i g_i(\sigma_{Z_t}) = \sum_{i=1}^{V}\beta_i G(a_i, b_i, \sigma_{Z_t}),\ \sigma_{Z_t} \in R^N,\ \beta_i \in R^M \tag{14}$$

where g_i denotes the output function $G(a_i, b_i, \sigma_{Z_t})$ of the ith hidden node. For additive nodes with activation function g, g_i can be defined as

$$g_i = G(a_i, b_i, \sigma_{Z_t}) = g(a_i{\cdot}\sigma_{Z_t} + b_i),\ a_i \in R^N,\ b_i \in R \tag{15}$$

and for radial basis function (RBF) nodes with activation function g, g_i becomes

$$g_i = G(a_i, b_i, \sigma_{Z_t}) = g(b_i\|\sigma_{Z_t} - a_i\|),\ a_i \in R^N,\ b_i \in R^+ \tag{16}$$

4.2. ELM learning based on GARCH process

For N arbitrarily distinct samples $(\sigma_{z_i(t)}, \sigma_{y_j(t)}) \in R^N \times R^M$, SLFNs with V hidden nodes are mathematically modeled as

$$\sum_{i=1}^{V}\beta_i g_i(\sigma_{z_j(t)}) = \sum_{i=1}^{V}\beta_i G(a_i, b_i, \sigma_{z_j(t)}) = \sigma_{o_j(t)},\ j = 1...N \tag{17}$$

which can approximate the N samples with zero error, meaning that $\sum_{j=1}^{V}\left\|\sigma_{y_j(t)} - \sigma_{o_j(t)}\right\| = 0$. The N equations above can be written compactly as:

$$H_\sigma\beta = \sigma_{Y_t} \tag{18}$$

where H_σ is called the hidden layer output matrix of the SLFN, and the ith column of H_σ is the ith hidden node output with respect to inputs $\sigma_{z_1(t)}....\sigma_{z_i(t)}... \sigma_{z_N(t)}$. $h(\sigma_{Z_t}) = G(a_1, b_1, \sigma_{Z_t}), ... ,G(a_V, b_V, \sigma_{Z_t})$ is called the hidden layer feature mapping. Then the ith row of H_σ is the hidden layer feature mapping with respect to the ith input $\sigma_{z_i(t)}$: $h(\sigma_{z_i(t)})$. Specifically, parameter vectors (a_i, b_i), H_σ and weights vector β can be chosen which satisfy Eqs. (9)–(12).

According to statistical learning theory, the principles of machine learning models involve both empirical risk minimization (ERM) and structural risk minimization (SRM). A model with good generalization performance capability should strikes a good tradeoff between these two principles. Therefore, GELM model proposed in our paper utilize the regularized ELM which is based on both structural risk minimization principle and weighted least square. Following Deng et al. (2009), the GELM algorithm can not only reach the smallest training error but also generate the smallest norm of weights.

$$Minimize:\ \|H_\sigma\beta - \sigma_{Y_t}\|^2 and\|\beta\| \tag{19}$$

Different from traditional learning algorithms, the GELM model focuses on the generalization performance of feedforward neural networks states, through which the feedforward neural networks achieve smaller training error. In fact, the smaller the norm of weights is, the better generalization performance the networks tend to have.

Theorem 4.1. Given a standard SLFN with V hidden nodes and the activation function $G: R \to R$, which is infinitely differentiable in any interval for N arbitrary distinct samples $(z_i(t), y_i(t))$, according to the continuous probability distribution, for any a_i and b_i which are randomly selected from any intervals of R^n and R respectively, the hidden layer output matrix H of the SLFN is almost surely invertible and $\|H\beta - Y_t\| = 0$.

Theorem 4.1 has been rigorously proved by Huang et al. (2006) who show that the input weights and hidden layer biases of SLFNs can be randomly assigned if the activation functions in the hidden layer are infinitely differentiable. Thus, the parameters in the ELM, e.g., the hidden layer parameters (a_i, b_i) and the output weights β_i of the hidden layer, do not need to be tuned and can be independent of the training samples. One of the advantages of ELM is that its parameters can be randomly generated, for which it has the same virtues of stochastic parameters as in GELM.

Theorem 4.2. Given a matrix H such that Hy is a minimum norm least-squares solution of a linear system $Az = y$, then it can be proved that $H = A^{\dagger}$, which is the Moore-Penrose generalized inverse of matrix A.

Theorem 4.2 as proposed in Huang et al. (2006) suggests that the minimum norm least squares solution of ELM is globally minimal and unique. Therefore, another advantage of ELM is that the Moore-Penrose generalized inverse and the smallest norm least-squares solution of the above linear system is globally minimal and unique, for which it has the same virtues of global minimum and uniqueness as in GELM.

Here, we use the Moore-Penrose generalized inverse and the minimum norm least squares method to implement the GELM algorithm:

$$\widehat{\beta} = H_{\sigma}^{\dagger}\sigma_{Y_t} \tag{20}$$

where $H_{\sigma}^{\dagger}$ is the Moore-Penrose generalized inverse of matrix H_{σ}. According to Theorems 4.1 and 4.2, the input weights and hidden layer biases of GELM can be randomly assigned if the activation functions in the hidden layer are infinitely differentiable. After the input weights and the hidden layer biases are randomly selected, GELM model can be simply seen as a linear system. The Moore-Penrose generalized inverse and the minimum norm least-squares solution of a general linear system play an important role in developing the GELM learning algorithm. In theory, the output weights which link the hidden layer to the output layer of GELM can be analytically determined through simple generalized inverse operation of the hidden layer output matrices and therefore, it will be globally minimal and unique.

4.3. Output of classification-based ELM

On the basis of ELM theory (Huang et al., 2006), for the binary classification applications, the decision function of GELM classifier is

$$f_V(\sigma_{Z_t}) = \mathrm{sign}(h(\sigma_{Z_t})\beta) \tag{21}$$

In the binary classification case, GELM only uses a single-output node, and the class label which is closer to the output value of GELM is selected as the predicted class label of the input data. There are two solutions to multi-class classification. In each case, the solution to the GELM model under the binary clarification serves as a special case of multi-class solution. For example, in the case of multi-class and multi-output nodes, the GELM model uses multi-output nodes, and the index of the output node with the highest output value is considered as the label of the input data.

4.3.1. Multi-class classifier with single output

Since GELM can approximate any target continuous functions and the output of the GELM classifier $h(\sigma_{Z_t})\beta$ can be as close to the class labels in the corresponding regions as possible, the classification problem for the proposed constrained-optimization-based ELM with a single-output node can be formulated as

$$\begin{aligned} &Minimize: L_{DGELM} = \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\varsigma_i\|^2 \\ &Subject\ \ to: h(\sigma_{Z_t})\beta = \sigma_{Y_t} - \varsigma_i,\ i = 1, \ldots, N \end{aligned} \tag{22}$$

where C is the parameter controlling for the trade-off between the training error vector ς_i and the norm of output weights β. Based on the KKT theorem, the training process in GELM is equivalent to solving the following dual optimization problem:

$$L_{GELM} = \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\varsigma_i\|^2 - \sum_{i=1}^{N}\lambda_i(h(\sigma_{z_i(t)})\beta - \sigma_{y_i(t)} + \varsigma_i) \tag{23}$$

where each Lagrange multiplier λ_i corresponds to the ith training sample. We can have the KKT optimality conditions of Eq. (23) as follows:

$$\begin{cases} \dfrac{\partial L_{GELM}}{\partial \beta} = 0 \rightarrow \beta = \sum_{i=1}^{N}\lambda_i h(\sigma_{z_i(t)})^T = H^T\lambda \\ \dfrac{\partial L_{GELM}}{\partial \varsigma_i} = 0 \rightarrow \lambda_i = C\varsigma_i,\ i = 1, \ldots, N \\ \dfrac{\partial L_{GELM}}{\partial \lambda_i} = 0 \rightarrow h(\sigma_{z_i(t)})\beta - \sigma_{y_i(t)} + \varsigma_i = 0,\ i = 1 \ldots N \end{cases} \tag{24}$$

where $\lambda_i = [\lambda_1 \ldots \lambda_N]^T$. In this case, GELM is analogous to the univariate GARCH-type model.

4.3.2. Multi-class classifier with multi-outputs

An alternative approach for multi-class applications is to assume multi-output nodes instead of a single-output node in GELM model. Thus, M-class classifiers have M output nodes. If the original class label is k, the expected output vector of the M output nodes is then $\sigma_{y_i(t)} = [0 \ldots 0, \overset{k}{1}, 0 \ldots 0]^T$. In this case, only the kth element of $\sigma_{y_i(t)} = [\sigma_{y_{i,1}(t)} \ldots \sigma_{y_{i,M}(t)}]^T$ equal one, while the rest of the elements are set to zero. The classification problem with multi-output nodes in GELM model can be formulated as

$$\begin{aligned} &Minimize: L_{DGELM} = \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\varsigma_i\|^2 \\ &Subject\ \ to: h(\sigma_{z_i(t)})\beta = \sigma_{y_i(t)}^T - \varsigma_i^T,\ i = 1 \ldots N \end{aligned} \tag{25}$$

where $\varsigma_i = [\varsigma_{i,1} \ldots \varsigma_{i,M}]^T$ is the training error vector of the M output nodes with respect to the training sample $\sigma_{z_i(t)}$. Similarly, based on the KKT theorem, the training process in GELM is equivalent to solving the following dual optimization problem:

$$L_{GELM} = \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\varsigma_i\|^2 - \sum_{i=1}^{N}\sum_{j=1}^{M}(\lambda_{ij}(h(\sigma_{z_i(t)})\beta_j - \sigma_{y_{ij}(t)} + \varsigma_{ij}) \tag{26}$$

where β_j is the vector of the weights linking hidden layer to the jth output node and $\beta = [\beta_1 \ldots \beta_M]$. Correspondingly, we can have the KKT theory-based optimality conditions as follows:

$$\begin{cases} \dfrac{\partial L_{GELM}}{\partial \beta_j} = 0 \rightarrow \beta_j = \sum_{i=1}^{N}\lambda_{ij} h(\sigma_{z_i(t)})^T \rightarrow \beta = H^T\lambda \\ \dfrac{\partial L_{GELM}}{\partial \varsigma_i} = 0 \rightarrow \lambda_i = C\varsigma_i,\ i = 1 \ldots N \\ \dfrac{\partial L_{GELM}}{\partial \lambda_i} = 0 \rightarrow h(\sigma_{z_i(t)})\beta - \sigma_{y_i(t)}^T + \varsigma_i^T = 0,\ i = 1 \ldots N \end{cases} \tag{27}$$

where . From Eqs. (23)–(27), we can see that the conditions in single-output node can be treated as a special case of the multi-output nodes with the number of output nodes setting to be one: M=1. For both cases, the hidden-layer matrix H_{σ} remains the same, and the size of H_{σ} only depends on the number of training samples N and hidden nodes V, while the number of output nodes or the number of classes is irrelevant.

4.4. Random hidden layer feature mapping

Different methods can be used to calculate the Moore-Penrose generalized inverse of a matrix, such as orthogonal projection method, orthogonalization method, iterative method, and singular value decomposition (SVD) as in Huang et al. (2012). Taking into account of the simplicity and efficiency, we use the orthogonal projection to implement the GELM algorithm in two scenarios. Namely, if $H_{\sigma}^T H_{\sigma}$ is nonsingular, then $H_{\sigma}^{\dagger} = (H_{\sigma}^T H_{\sigma})^{-1} H_{\sigma}^T$. On the other hand, if $H_{\sigma} H_{\sigma}^T$ is nonsingular, then $H_{\sigma}^{\dagger} = H_{\sigma}^T (H_{\sigma} H_{\sigma}^T)^{-1}$. Different solutions to the aforementioned KKT conditions can be obtained depending on the size of training data set.

If the number of training samples is not huge, for instance, the number of hidden nodes V can be much smaller than the number of

training samples: $V << N$, the computational cost reduces dramatically. In this case, by substituting Eq. (27), we can get:

$$\left(\frac{\mathrm{I}}{C} + H_\sigma H_\sigma^T\right)\lambda = \sigma_{Y_t} \tag{28}$$

Then from Eqs. (27) to (28), we can have

$$\beta = H_\sigma^T\left(\frac{\mathrm{I}}{C} + H_\sigma H_\sigma^T\right)^{-1}\sigma_{Y_t} \tag{29}$$

Let the time series $x_i(t)$, $i = 1..\ .N$, $t = 1..\ .T'$ belong to the test data set $X_t = \{x_1(t)...x_i(t)...x_N(t)\}$. As previously mentioned, one can get the standard deviations of the test data set as $\sigma_{X_t} = \{\sigma_{x_1(t)}....\sigma_{x_i(t)}.\ ..\sigma_{x_N(t)}\}$.

The output function of GELM is then:

$$\sigma_{test} = f(\sigma_{X_t}) = h(\sigma_{X_t})\beta = h(\sigma_{X_t})H_\sigma^T\left(\frac{\mathrm{I}}{C} + H_\sigma H_\sigma^T\right)^{-1}\sigma_{Y_t} \tag{30}$$

If the hidden layer feature mapping $h(\sigma_{X_t})$ is unknown to users, the kernel matrix of GELM classifier can be defined as $\Omega_{GELM} = H_\sigma H_\sigma^T$: $\Omega_{GELMij} = K(\sigma_{xi(t)}, \sigma_{xj(t)})$, the output function of GELM can be written compactly as:

$$\begin{aligned}\sigma_{test} &= f(\sigma_{X_t}) = h(\sigma_{X_t})\beta = h(\sigma_{X_t})H_\sigma^T\left(\frac{\mathrm{I}}{C} + H_\sigma H_\sigma^T\right)^{-1}\sigma_{Y_t}\\ &= \begin{bmatrix} K(\sigma_{X_t}, \sigma_{x_1(t)}) \\ \cdots \\ K(\sigma_{X_t}, \sigma_{x_N(t)}) \end{bmatrix}^T\left(\frac{\mathrm{I}}{C} + \Omega_{GELM}\right)^{-1}\sigma_{Y_t}\end{aligned} \tag{31}$$

In the contrast, if the number of training samples is huge, for example, it is much larger than the dimensionality of the feature space, $N >> V$, then we can have the solution from Eq. (27) as follows:

$$\begin{cases}\beta = CH_\sigma^T\varsigma \\ \varsigma = \frac{1}{C}(H_\sigma^T)^\dagger\beta \\ H_\sigma^T(H_\sigma + \frac{1}{C}(H_\sigma^T)^\dagger)\beta = H_\sigma^T\sigma_{Y_t} \\ \beta = \left(\frac{1}{C} + H_\sigma^T H_\sigma\right)^{-1}H_\sigma^T\sigma_{Y_t}\end{cases} \tag{32}$$

The output function of GELM model can be written compactly as:

$$\sigma_{test} = f(\sigma_{X_t}) = h(\sigma_{X_t})\beta = h(\sigma_{X_t})\left(\frac{\mathrm{I}}{C} + H_\sigma^T H_\sigma\right)^{-1}H_\sigma^T\sigma_{Y_t} \tag{33}$$

4.5. Value-at-Risk estimation

Since the 1990s, Value-at-Risk has become a popular method for risk measures (Giot and Laurent, 2004). $VaR(1-\rho)$ is formally defined as the boundary that is exceeded exactly $100*\rho$ times out of 100 trials. $(1-\rho)$ is called the confidence level such that:

$$P(R_t < VaR(1-\rho)) = \rho \tag{34}$$

where R_t is the realized return at time t calculated as $R_t = \ln(P_t/P_{t-1}) \times 100$, and P_t is the closing price of day t. In this paper, we extend the methodology of VaR estimation in Kim and Lee (2016) and propose a new method to calculate VaR in a high-dimensional space. Namely, after calculating the standard deviations of the test data $\sigma_{X_t} = \{\sigma_{x_1(t)}....\sigma_{x_i(t)}.\ ..\sigma_{x_N(t)}\}$, the VaR can be calculated as:

$$VaR(1-\rho) = \overline{R} - \sigma_{test}c_\rho \tag{35}$$

where $\overline{R}$ is the mean value of the sample, and c_ρ is the critical value of the assumed distribution with tail area ρ.

4.5.1. Performance of volatility forecasting

We consider two evaluation criteria commonly used in the previous literature on forecasting performance (Hou, 2013), which are the mean absolute error (MAE) and root mean square error (RMSE). Let σ_t and $\hat{\sigma}_t$ being the observed standard deviation and predicted standard deviation at time t respectively, and N_1 be the number of forecasting periods. The evaluation criteria are given as follows:

$$\begin{cases} MAE = N_1^{-1}\sum_{t=1}^{N_1}|\sigma_t - \hat{\sigma}_t| \\ RMSE = \sqrt{N_1^{-1}\sum_{t=1}^{N_1}(\sigma_t - \hat{\sigma}_t)^2}\end{cases} \tag{36}$$

4.5.2. Unconditional and conditional coverage tests

Many researchers are concerned about the adequacy of the Value-at-Risk measures, especially when they compare the various methods to calculate VaR. Abad et al. (2014) demonstrate that the standard tests on the adequacy of VaR models include the unconditional and conditional coverage tests (e.g., the LR_{uc} test, and the LR_{cc} test as follows).

Considering a stochastic process ℓ_t which follows a binomial distribution, then

$$\ell_{t+1} = \begin{cases}1 & if\ \ R_{t+1} < -VaR_{t+1|t(\rho)} \\ 0 & else\end{cases} \tag{37}$$

where the stochastic process ℓ_t is called the failure process.

The VaR forecasts are considered to be efficient if they show correct results of conditional coverage, that is, $E(\ell_{t|t-1}) = \rho$. Kupiec (1995) develops a test statistic (LR_{uc}) to test for correct unconditional coverage in the likelihood ratio framework. Namely, the likelihood ratio test statistic is given by

$$LR_{uc} = -2\log\frac{(1-\rho)^{n0}\rho^{n1}}{(1-\varpi)^{n0}\varpi^{n1}} \tag{38}$$

where ρ is the tolerance level at which VaR measures are estimated, n_1 and n_0 are the numbers of 1 and 0 in the indicator series respectively, and $\varpi = n_1/(n_0+n_1)$. The LR_{uc} tests have been widely adopted in a vast amount of literature on VAR measures, such as Angelidis et al. (2004), Cheng and Hung (2011), Orhan and Köksal (2012), Nikolaev et al. (2013),Youssef et al. (2015), Dionne et al. (2015), Kim and Lee (2016), etc.

One of the main shortcomings of the LR_{uc} test is that it does not account for the dynamics in the higher-order moments. To relax the assumption of homoscedastic errors, Christoffersen (1998) develops a more robust test, the conditional coverage test (LR_{cc}), which jointly tests for correct conditional coverage and the violations of independence assumption. Under the null hypothesis that the failure process is independent and the expected proportion of exceptions equals ρ, the corresponding likelihood ratio is given as follows:

$$LR_{cc} = -2\log\frac{(1-\rho)^{n0}\rho^{n1}}{(1-\varpi_{01})^{n00}\varpi_{01}^{n01}(1-\varpi_{11})^{n10}\varpi_{11}^{n11}} \tag{39}$$

where n_{ij} is the number of observations with value i followed by value j $(i, j = 0, 1)$,$\varpi_{01} = n_{01}/(n_{00}+n_{01})$, and $\varpi_{11} = n_{11}/(n_{10}+n_{11})$.

4.5.3. Dynamic quantile test

Engle and Manganelli (2004) propose a conditional coverage test by using a linear regression model based on the process of hit function:

$$F_t = \ell_t - \rho\begin{cases}1-\rho & if\ \ R_t < -VaR_t \\ -\rho & else\end{cases} \tag{40}$$

where F_t is a centered process in the target probability ρ. The dynamics of the hit function is modeled as:

$$F_t = \lambda_0 + \sum_{j=1}^{P}\lambda_j F_{t-j} + \sum_{k=1}^{K}\varphi_k f_k(v_t) + w_t \tag{41}$$

where w_t is an i.d.d process, and $f(\cdot)$ is a function of past exceedance of variable v_t. Under the null hypothesis that the process delivers accurate

VaR estimates and the occurrence of p consecutive exceedences is uncorrelated, the regressors should have no explanatory power. Hence the dynamic quantile (DQ) test is defined as:

$$H_0: \psi = (\lambda_0, \lambda_1, \ldots, \lambda_p, \varphi_1, \varphi_2, \ldots, \varphi_k)^T = 0 \tag{42}$$

It's straightforward to show that the dynamic quantile test statistic, which is analogous to the Wald statistic:

$$DQ = \frac{\hat{\psi}^T S^T \hat{\psi}}{\rho(1-\rho)} \rightarrow \chi^2_{1+P+k} \tag{43}$$

where S denotes the covariate matrix in Eq. (38), and $f(v_t) = \hat{VaR}_t$.

5. Experimental results and discussions

In contrast to a large body of literature on diagnostic tests devoted to univariate models, there are few tests which are specific to multivariate models. In our simulation experiment, we focus on the case of single-output node, i.e., the number of output nodes setting to one ($M = 1$), and compare the GELM model to several popular univariate GARCH-type models as well as SVMs and ELMs. The simulations have been done using the computers with 4 GB of Ram and 2+ GHZ CPU processor, and the computation environment is MATLAB 7.0.[5]

5.1. Data description

Figs. 2 and 3 plot the daily closing prices and the daily log-returns of the China Securities Index 300 (CSI300) respectively.[6] The data are obtained from the CSMAR database for the period of April 8, 2005 to May 28, 2015. Fig. 3 suggests that there is strong evidence of volatility clustering in the return series, with strong (weak) price fluctuations followed by strong (weak) fluctuations. The figure also suggests that the sample data have the properties of leptokurtosis as well, which is commonly observed in the financial time series.

In the empirical experiment, we use the daily returns of CSI index as one of our inputs. Furthermore, considering the fact that for GELM learning algorithms, the feature selection is not restricted to certain types of inputs, which is one of the advantages of GELM compared to the traditional structural models. Therefore, we implement our model by including additional 305 stocks which are randomly selected and computationally proven to be useful in predicting the volatility of CSI300 index.[7]

The empirical investigation is based on 2,462 daily observations of the CSI300. If we denote the closing price of a stock i, $i = 1 \ldots N$ at day t as P_{it}, then the daily returns R_{it} for stock i are computed on a continuous compounding basis as $R_{it} = \ln(P_{it}/P_{i(t-1)}) \times 100$.

Table 1 presents the descriptive statistics for the returns series. The return series has negative skewness, implying that the distribution is left-skewed. The kurtosis statistic is significantly greater than 3, implying that the series is fat-tailed. The Jarque-Bera test is statistically significant, suggesting that the series is not normally distributed. The statistics of the ADF test (Dickey and Fuller, 1981) and PP test (Phillips and Perron, 1988) in Table 1 checks the stationarity of the returns of CSI300 Index. The results show that the index returns of CSI 300 have no unit roots and are stationary. Therefore, conventional time-series models can be used to examine the behavior of volatility over time. The Ljung-Box Q test for serial correlation shows that the null hypothesis of no autocorrelation up to the 10th order is rejected, and serial autocorrelation is present in the index return. The Lagrange Multiplier statistic is significant at the 1% level, confirming the ARCH effects (Engle, 1982). In sum, the descriptive statistics support the idea of modelling the index returns of CSI300 by GARCH processes.

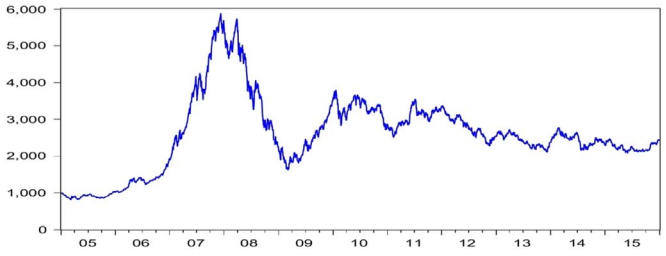

Fig. 2. Daily closing price of the CSI300.

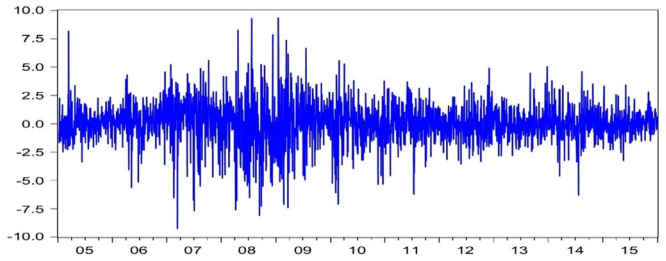

Fig. 3. Daily log-return of the CSI300.

5.2. Estimation of GARCH-type models

The estimated parameters and diagnostic tests for GARCH-type models are reported in Table 2.

Namely, we estimate 11 different parametric GARCH models, all of which use a Student's t distribution, except for the EGARCH model. The test statistics in the parentheses in top panel of Table 2 are the corresponding p values of t-statistic for the coefficients of GARCH-type models. As the p values of the coefficients are smaller than 0.05, it indicates the coefficients are significant. Turning to the results of the log likelihood values ($\hbar_6$) in Table 2, the numbers are negative and of similar magnitudes. However, with regard to the diagnostic tests, our results indicate that the null hypothesis of correct model specification can't be rejected as the Box-Pierce statistic computed with 10 lags shows neither serial correlation nor remaining ARCH effect. In sum, the stationary conditions are satisfied when using at most 2 lags for ARCH and GARCH coefficients. That is, all of the ARCH and GARCH coefficients are positive and less than one. Besides, the sum of ARCH and GARCH coefficients are smaller than one for the GARCH-type models. In fact, similar results on GARCH types model estimations have been presented in a vast amount of literature (e.g., Hou and Suardi, 2012; Arouri et al., 2012; Charfeddine, 2014; Youssef et al., 2015).

5.3. Performance of out of sample volatility forecasting

One-step-ahead forecasts for the interval $[t, t + \Delta t]$ are generated from models using an rolling window of all the observations available at time $t - 1$, thus the corresponding forecast is based on information up to time $t - 1$. To compare the out-of-sample forecasting results by employing parametric GARCH-type models, a wide range of forecasting horizons have been applied in previous studies. For example, Babikir et al. (2012) use the forecasting time horizons of 20-, 60-, and 120-day. Siburg et al. (2015) conduct 300-day ahead forecasting. Wang et al. (2015) use 5- and 10-day forecasting windows. While in Chkili et al. (2014), the forecasting periods are relatively shorter, which are 1- and 20-day ahead respectively.

[5] A summary of the GEML algorithm used in this paper is presented in Appendix A.

[6] The CSI300 is a market capitalization weighted index that maps the performance of the 300 most highly liquid A shares on the Shanghai and the Shenzhen Stock Exchanges. The index was created by the China Securities Index Company Ltd on August 4, 2005, and can be tracked back to December 2004 to a base price of 1000. For a more detailed introduction to the CSI300, please refer to the official website of China Securities Index Co Ltd.

[7] A list of the codes of 305 stocks which we use for calculation is presented in Appendix B.

Table 1
Summary statistics for the daily return of CSI300.

Mean	S.D.	Skewness	Kurtosis	Jarque-Bera	ADF	PP	Q(10)	$Q^2(10)$	ARCH test
0.080	1.815	−0.026	5.922	905.083* (0.000)	−48.229* (0.000)	−48.288* (0.000)	32.079* (0.000)	23.862* (0.000)	211.425* (0.000)

Note: Figures in (.) are p-values. Q(10) and Q2(10) are the Ljung-Box Q tests for the 10th order serial correlation of the standardized residuals and squared standardized residuals, respectively. Engles ARCH test also examines the autocorrelation of the squared returns.
* Significant at the 1% level.

Table 2
Estimation of GARCH-type models.

	Symmetric models			**Asymmetric models**			**Long memory models**			**Long memory model with asymmetry**	
	GARCH	IGARCH	RiskMetrics	GJR	EGARCH	APARCH	FIGARCH-BBM	FIGARCH-CHUNG	HYGARCH	FIAPARCH-BBM	FIAPARCH-CHUNG
$\hbar_1$	0.025** (0.028)	0.026** (0.031)	0.060 (NA)	0.042* (0.000)		0.048* (0.000)	0.234* (0.000)	0.251* (0.000)	0.250* (0.000)		0.252* (0.000)
$\hbar_2$			0.940 (NA)	0.953* (0.000)	0.507** (0.042)	0.951* (0.000)	0.715* (0.000)	0.691* (0.000)	0.694* (0.000)	0.955* (0.000)	0.701* (0.000)
$\hbar_3$	0.062* (0.000)	0.065* (0.000)									
$\hbar_4$	0.893* (0.000)	0.896* (0.000)			0.479** (0.052)						
$\hbar_5$						1.639* (0.000)	0.501* (0.000)	0.460* (0.000)	0.453* (0.004)	1.032* (0.000)	0.469* (0.000)
$\hbar_6$	5.587* (0.000)	5.296* (0.000)	6.487* (0.000)	5.591* (0.000)		5.665* (0.000)	5.551* (0.000)	5.743* (0.000)	5.461* (0.000)	5.665* (0.000)	5.608* (0.000)
Diagnostic test											
Log Value	−4524.2	−4524.74	−4539.26	−4527.68	−4581.18	−4527.36	−4528.38	−4528.45	−4528.26	−4525.7	−4528.31
$Q^2(10)$	7.894 [0.245]	7.965 [0.240]	8.125 [0.421]	8.950 [0.346]	5.383 [0.495]	9.490 [0.302]	11.877 [0.156]	11.428 [0.178]	11.455 [0.177]	7.769 [0.456]	11.593 [0.170]
ARCH(10)	0.777 [0.651]	0.783 [0.644]	0.812 [0.616]	0.901 [0.531]	0.546 [0.858]	0.955 [0.480]	1.220 [0.272]	1.175 [0.302]	1.179 [0.299]	0.800 [0.628]	1.190 [0.292]

Note: Numbers in parentheses are p-values calculated using robust standard errors. ** and * stand for significance at 5% and 1% significant level respectively. $\hbar_1$,$\hbar_2$,$\hbar_3$ and $\hbar_4$ denotes ARCH and GARCH coefficients with lag 1 and 2 respectively. The lags of the ARCH and GARCH coefficients are determined by the AIC and SIC information criteria. $\hbar_5$ denotes the estimated degree of integration for fractionally integrated GARCH-type models and $\hbar_6$ denotes the degree of freedom of Student's t-distribution respectively. Log Value denotes the value of logarithm maximum likelihood function. $Q^2(10)$ is the Ljung–Box Q-statistics for the squared standardized residuals of order 10. ARCH(10) is the non-heteroskedasticity statistics of order 10. The corresponding *p*-values of the statistics are reported in square brackets.

In fact, according to the existing literature, the forecasting performance for Value-at-Risk depends on the time horizon to some extent. Therefore, following Babikir et al. (2012), Chkili et al. (2014) and Wang et al. (2015), we compare the out-of-sample VaR forecasting performances using different types of models for 20- and 60-day ahead forecasting horizons respectively. The overall results confirm that the proposed GELM model achieves better forecasting performance at both medium- and long- time horizons.

Two criteria are employed to evaluate the 20-day and 60-day ahead forecasting performances of various models, including the mean absolute error (MAE) and the root mean square error (RMSE), which are commonly used in the previous literature, such as Arouri et al. (2012) and Charfeddine (2014). The results are also categorized into different training and testing groups respectively. MAE1 and RMSE1 refer to the mean absolute error and root mean squared error calculated for the training state respectively, while MAE2 and RMSE2 refer to the corresponding values in the test stage. The forecasting performances of all the models adopted in this paper are presented in Tables 3 and 4. For the GARCH-type models, all the coefficients are estimated via the maximum likelihood estimation (MLE). However, for the learning algorithms, the values of the criteria are the average of the estimates repeated 50 times. That is to say, 50 trials have been conducted for all the learning algorithms to get the best training accuracy which is a common practice in the domain of Machine Learning and have been recorded in a vast amount of relevant literature (e.g., Huang et al., 2006; Du et al., 2014; Alexandre et al., 2015).

In the empirical experiment, the 2462 daily observations are divided into two periods. Namely, when performing the 20-day ahead forecasting, we use the data spanning the period of April 11, 2005 to May 7, 2015 as the training period and the remaining observations are reserved as the test period (i.e., May 8, 2015, to May 28, 2015). When performing the 60-day ahead forecasting, we use the data spanning the period of April 11, 2005 to March 29, 2015 as the training period, while the remaining observations are used as the test period (i.e., March 30, 2015 to May 28, 2015).

Table 3
Performance of 20-day ahead out-of-sample volatility forecasting.

	MAE1	RMSE1	Training Time	MAE2	RMSE2	Test Time
GARCH	1.828	2.296		5.028	10.57	
IGARCH	1.803	2.27		5.007	10.51	
RISKM	1.803	2.278		5.091	10.66	
GJR	1.854	2.32		5.033	10.49	
EGARCH	1.795	2.263		5.027	10.42	
APARCH	1.836	2.306		5.047	10.54	
FIGARCH-BBM	1.798	2.295		5.14	10.75	
FIGARCH-CHUNG	1.8	2.278		5.105	10.66	
HYGARCH	1.799	2.263		5.094	10.64	
FIAPARCH-BBM	1.798	2.262		5.092	10.64	
FIAPARCH-CHUNG	1.801	2.261		5.038	10.5	
SVM	0.056	0.070	4.468	0.009	0.099	0.169
ELM	1.211	1.691	1.704	2.064	2.064	0.003
ELM-RBF	1.070	1.468	4.028	1.838	1.838	0.061
GELM	0.008	0.016	1.850	0.019	0.019	0.004
GELM-RBF	0.005	0.013	4.375	0.011	0.011	0.062

Table 4
Performance of 60-day ahead out-of-sample volatility forecasting.

	MAE1	RMSE1	Training Time	MAE2	RMSE2	Test Time
GARCH	3.822	4.215		3.279	6.355	
IGARCH	3.074	3.555		3.486	6.342	
RISKM	3.096	3.573		3.134	6.361	
GJR	3.806	4.198		3.569	6.34	
EGARCH	3.163	3.61		3.272	6.298	
APARCH	3.06	3.544		3.192	6.356	
FIGARCH-BBM	2.535	3.161		3.026	6.365	
FIGARCH-CHUNG	2.756	3.314		3.015	6.391	
HYGARCH	2.757	3.301		3.032	6.362	
FIAPARCH-BBM	2.779	3.315		3.189	6.353	
FIAPARCH-CHUNG	3.265	3.703		3.031	6.361	
SVM	0.792	1.133	3.564	0.873	1.089	0.084
ELM	1.209	1.691	1.419	1.505	1.504	0.001
ELM-RBF	1.069	1.468	2.235	1.451	1.450	0.063
GELM	0.008	0.016	1.461	0.019	0.019	0.002
GELM-RBF	0.005	0.013	2.934	0.011	0.010	0.045

According to the theory of ELM, the number of hidden nodes can be much smaller than the number of training samples in most applications. In our empirical experiment, the number of training sample in high-dimensional space is not huge (i.e., 2462 observations for each of 306 stocks). Therefore, to mitigate the potential over-parameterization problem, 300 hidden nodes are assigned for both ELM and GELM algorithms. The results are quantitatively similar to those using 1000 hidden nodes. Besides, if 1000 hidden nodes are assigned for ELM-RBF or GELM-RBF algorithms, the algorithm may not work. Thus, 250 hidden nodes are assigned for ELM-RBF and GELM-RBF algorithms in our paper.

The MAE and RMSE statistics suggest that either the SVM or ELM model is better than the GARCH-type models. GARCH-type models perform the worst with regards to all goodness-of-fit measures. As for the training results, the proposed GELM model (GELM-RBF) is better than other models. The test results clearly show that the GELM model (GELM-RBF) achieves better forecasting performance. All models are conducted under the same computation environment, so the comparability is effective. The GELM model (GELM-RBF) can not only improve the forecasting accuracy, but also substantially save the computation time. The competitive advantage lies in the fact that the GELM model is based on a combination of the GARCH model and the Regularized ELM algorithm, which randomly chooses hidden nodes and analytically determines the output weights of SLFNs. In general, ELMs tend to generate over-fitting results and fail to take heteroskedasticity into account in applications. Its performance will be seriously affected when there are outliers in the dataset. To overcome these drawbacks, when implementing GELM, we utilize the regularized ELM which is based on both structural risk minimization principle and weighted least square. Therefore, the GELM can reach the smallest training error and generate the smallest norm of weights. Besides, the output weights of GELM are determined by Moore–Penrose generalized inverse. Thus, the generalization performance of GELM improves significantly in most cases without increasing training time. In sum, the GELM model can achieve better generalization performance in terms of efficiency and accuracy for the volatility forecasting.

5.4. Performance of out of sample VaR estimation

We initially use the training data set, which contains 2312 observations, to estimate VaR_{2313} at time $t = 2313$. For each new observation, we re-estimate VaR based on the conditional levels. This means that we estimate VaR_{2314} using observations $t = 2$ to 2314, and VaR_{2315} using observations $t = 3$ to 2314, so on and so forth, until the last observation of test dataset, i.e $t = 2, 462$. Because we have 20(60) out-of-sample observations, the number of diagonal tests for VaR estimation are also 20(60). In order to save space, we only calculate the VaR for the long position. The out of sample VaRs are computed and compared with the observed returns as reported in Tables 5 and 6.

Generally speaking, the evaluation on the performance for different VaR models in terms of accuracy and reliability is a tricky issue. On the one hand, an appropriate VaR model is expected to provide reliable estimates that adequately cover significant losses. On the other hand, the estimated VaR needs to track closely the underlying risk evolution in order to minimize the idle capital and other costs. Therefore, following He et al. (2012), we adopt the Mean Squared Error (MSE) to compare the predictive accuracy of VaR models. Specifically, the MSE can be defined as $N_1^{-1}\sum_{t=1}^{N_1}(\hat{\text{VaR}}_t - \text{VaR}_t)^2$, where N_1 is the number of periods for out-of-sample forecasting. In addition, when examining the performance for VaR calculation, we also consider the LR_{uc} test, the LR_{cc} test, and the DQ test. The results of the test statistics for 10- and 20-day ahead VaR estimates are shown in Tables 5 and 6 respectively.

Here, the null hypothesis of the LR test is that the failure rate doesnot equal the theoretical value, while the null hypothesis of the DQ test is that the goodness of fit for the estimated VaR process is significantly poor. The null hypotheses for the LR test or DQ test are rejected if the p-value is larger than the 5% significance level when taking a long position with 0.05 and 0.025 confidence levels respectively. According to both the LR and DQ tests, the GELM model (GELM-RBF) is better at calculating VaR than other models. For example, for both 10-day ahead and 20-day ahead estimation horizons, the p values of LR tests and DQ tests for GELM (GELM-RBF) are larger than 0.05, which indicates the rejection of the null hypotheses and confirms that the computed VaR estimates are sufficiently accurate and the goodness of fit for the computed VaR estimates is significantly good. Meanwhile, the MSE of the GELM (GELM-RBF) is significantly smaller than those of other models, which suggests that the GELM model (GELM-RBF) is more adept at forecasting the VaR.

The empirical results in Tables 5 and 6 indicate that under different confidence levels, the estimated GELM (GELM-RBF) model has a much lower value of MSE and therefore higher predictive accuracy, suggesting that the GELM model (GELM-RBF) can substantially outperform other models. This finding is consistent with the larger p values of GELM (GELM-RBF) under different confidence levels. In

Table 5
Performance of 20-day-ahead out-of-sample VaR estimates.

Long	*Position*	*MSE*	*LR-uc*	*p-value*	*LR-cc*	*p-value*	*DQ*	*p-value*
GARCH	0.05	17.441	0.403	0.525	1.949	0.377	8.442	0.207
	0.025	20.717	1.286	0.256	0.962	0.618	12.610	0.049
IGARCH	0.05	17.341	0.129	0.718	1.169	0.557	5.911	0.433
	0.025	20.599	0.576	0.447	3.058	0.217	10.010	0.124
RISKM	0.05	17.589	7.175	0.007	31.275	0.000	8.160	0.226
	0.025	20.894	14.819	0.000	43.232	0.000	12.935	0.044
GJR	0.05	17.308	1.389	0.238	1.949	0.377	4.752	0.575
	0.025	20.560	1.581	0.208	0.962	0.618	2.763	0.837
EGARCH	0.05	17.193	0.010	0.918	1.169	0.557	4.015	0.674
	0.025	20.423	3.488	0.061	3.058	0.217	11.426	0.076
APARCH	0.05	17.209	1.185	0.276	1.949	0.377	4.523	0.606
	0.025	20.442	1.019	0.312	0.962	0.618	10.783	0.095
FIGARCH-BBM	0.05	17.737	7.646	0.005	31.275	0.000	12.380	0.054
	0.025	21.07	9.298	0.002	43.232	0.000	9.835	0.131
FIGARCH-CHUNG	0.05	17.589	5.846	0.015	31.275	0.000	13.299	0.038
	0.025	20.893	13.962	0.000	43.232	0.000	13.158	0.040
HYGARCH	0.05	17.556	0.824	0.363	1.949	0.377	6.757	0.343
	0.025	20.854	1.581	0.208	0.962	0.618	7.141	0.307
FIAPARCH-BBM	0.05	17.556	0.996	0.318	1.169	0.557	7.010	0.319
	0.025	20.854	2.641	0.104	3.058	0.217	8.036	0.235
FIAPARCH-CHUNG	0.05	17.325	0.527	0.467	1.169	0.557	6.160	0.405
	0.025	20.58	4.960	0.025	3.058	0.217	7.870	0.247
SVM	0.05	0.164	2.052	0.152	1.949	0.377	0.842	0.991
	0.025	0.194	1.013	0.314	0.962	0.618	0.410	0.999
ELM	0.05	3.405	2.810	0.094	3.772	0.152	10.915	0.091
	0.025	4.045	2.664	0.103	3.058	0.217	2.910	0.820
ELM-RBF	0.05	3.032	9.003	0.003	12.298	0.002	27.458	0.000
	0.025	3.602	15.155	0.000	18.502	0.000	60.485	0.000
GELM	0.05	0.031	2.052	0.152	1.949	0.377	0.842	0.991
	0.025	0.037	1.013	0.314	0.962	0.618	0.410	0.999
GELM-RBF	0.05	0.017	2.052	0.152	1.949	0.377	0.842	0.991
	0.025	0.021	1.013	0.314	0.962	0.618	0.410	0.999

sum, the MSE statistics as well as the LR tests (including both unconditional and conditional test) and the DQ test, suggest that the proposed GELM model (GELM-RBF) performs better in calculating VaR in terms of accuracy and consistency.

6. Conclusion

In view of the increasing uncertainty in the financial markets, we propose a non-linear random mapping model, called GELM model. The GELM model is based on a combination of GARCH and ELM models, and can be applied to risk management (i.e., VaR calculation and stress testing). We estimate the VaR by firstly employing the GELM model (GELM-RBF) to predict the volatility of the corresponding time series, and then extrapolate the predicted volatilities to calculate VaR. Compared with the GARCH-type models, ELM, and SVM models, the GELM models improve the forecasting performance substantially in terms of accuracy and efficiency, and thereby leading to better VaR calculations. Using daily data on the CSI300 index and additional 305 stock prices, our results confirm that the GELM model (GELM-RBF) is better at forecasting future volatility and can be used as an essential tool for risk management. However, in order to achieve more timely risk estimates (i.e. real-time VaR calculation), a new online algorithm is expected to develop, which points the direction for future study.

Acknowledgements

We are grateful to Xue-zhong He (the guest editor) and two anonymous referees for their insightful comments and suggestions, which lead to substantial improvements of the paper. We also thank the participants at the 2016 International Conference on Applied Financial Economics (ICAFE) for helpful discussions. Financial support from the National Social Science Foundation (Grant number: 15BJY155), and Ministry of Education's Humanities and Social Science Research Project (Grant number: 14YJA790049) are greatly acknowledged.

Table 6
Performance of 60-day-ahead out-of-sample VaR estimates.

Long	*Position*	*MSE*	*LR-uc*	*p-value*	*LR-cc*	*p-value*	*DQ*	*p-value*
GARCH	0.05	10.485	0.536	0.464	1.804	0.406	8.555	0.200
	0.025	12.455	1.306	0.253	0.177	0.916	12.922	0.044
IGARCH	0.05	10.464	0.300	0.583	0.431	0.806	5.832	0.442
	0.025	12.430	0.795	0.372	0.177	0.916	9.871	0.130
RISKM	0.05	10.495	7.286	0.006	52.901	0.000	8.804	0.184
	0.025	12.467	15.034	0.000	80.039	0.000	12.504	0.051
GJR	0.05	10.461	1.634	0.201	6.052	0.048	5.517	0.479
	0.025	12.426	1.035	0.308	2.987	0.224	3.316	0.768
EGARCH	0.05	10.391	0.010	0.917	6.052	0.048	3.268	0.774
	0.025	12.344	3.098	0.078	2.987	0.224	11.782	0.067
APARCH	0.05	10.487	1.012	0.314	6.052	0.048	4.280	0.638
	0.025	12.457	1.035	0.308	2.987	0.224	11.082	0.085
FIGARCH-BBM	0.05	10.502	0.837	0.360	1.804	0.406	5.791	0.447
	0.025	12.475	1.936	0.164	0.177	0.916	8.119	0.229
FIGARCH-CHUNG	0.05	10.545	1.411	0.234	1.803	0.405	5.813	0.444
	0.025	12.526	2.295	0.129	0.176	0.915	7.955	0.241
HYGARCH	0.05	10.497	0.678	0.410	1.804	0.406	5.847	0.440
	0.025	12.469	1.306	0.253	0.177	0.916	9.190	0.163
FIAPARCH-BBM	0.05	10.482	1.012	0.314	1.803	0.405	5.985	0.424
	0.025	12.451	2.295	0.129	0.176	0.915	13.067	0.041
FIAPARCH-CHUNG	0.05	10.495	1.204	0.272	1.804	0.406	5.909	0.433
	0.025	12.467	1.936	0.164	0.177	0.916	8.119	0.229
SVM	0.05	1.796	52.128	0.000	52.901	0.000	113.556	0.000
	0.025	2.134	91.681	0.000	92.663	0.000	299.344	0.000
ELM	0.05	2.482	1.872	0.171	324.123	0.000	2.947	0.815
	0.025	2.948	0.193	0.660	404.361	0.000	1.435	0.963
ELM-RBF	0.05	2.393	30.787	0.000	524.116	0.000	69.350	0.000
	0.025	2.842	45.465	0.000	604.511	0.000	130.875	0.000
GELM	0.05	0.031	6.155	0.013	6.053	0.049	2.947	0.815
	0.025	0.037	3.038	0.081	2.988	0.225	1.436	0.964
GELM-RBF	0.05	0.017	6.155	0.013	6.052	0.048	2.947	0.815
	0.025	0.020	3.038	0.081	2.987	0.224	1.435	0.963

Appendix A. Algorithm of GELM Model

Algorithm 1. . ELM Learning based on GARCH process.

Given a training set $(z_i(t), y_i(t)) \in R^N \times R^M$, hidden node output function G, and hidden node number V.
Steps:
For $i = 1..$ $.N$
For $t = 1...$ T
(1) Calculate the standard deviation $\sigma_{z_i(t)}$ of the time series z_i(t):

$$\sigma_{z_i(t)} = sqrt\{[(z_i(1) - \overline{z}_i)^2 + ...+(z_i(T) - \overline{z}_i)^2]/T\}$$

So one can get the standard deviation of the training data $\sigma_{Z_t} = \{\sigma_{z_1(t)}... \ .\sigma_{z_i(t)}...\sigma_{z_N(t)}\}$. Similarly, one can get the standard deviation of the target matrix of training data $\sigma_{Y_t} = \{\sigma_{y_1(t)}....\sigma_{y_M(t)}\}$.
(2) Assign parameters of hidden nodes (a_j, b_j) randomly, $j = 1...$ V.
(3) Calculate the hidden layer output matrix H_σ:

$$H_\sigma = \begin{bmatrix} h(\sigma_{z_1(t)}) \\ \vdots \\ h(\sigma_{z_N(t)}) \end{bmatrix} = \begin{bmatrix} h_1(\sigma_{z_1(t)}) \cdots h_V(\sigma_{z_1(t)}) \\ \vdots \qquad\qquad \vdots \\ h_1(\sigma_{z_N(t)}) \cdots h_V(\sigma_{z_N(t)}) \end{bmatrix}$$

(4) Calculate the output weight $\widehat{\beta} : \widehat{\beta} = H_\sigma^{\dagger}\sigma_{Y_t}$, where $H_\sigma^{\dagger}$ is the Moore-Penrose generalized inverse of matrix H_σ.

End
End

Algorithm 2. . Value-at-Risk calculation based on ELM learning

Let the time series $x_i(t)$, $i = 1..\ .N$, $t = 1..\ .T'$ belong to the test data set $X_t = \{x_1(t)...x_i(t)...x_N(t)\}$. One can get the standard deviation of the test data set $\sigma_{X_t} = \{\sigma_{x1(t)}....\sigma_{xi(t)}.\ ..\sigma_{xN(t)}\}$.
Steps:
For $i = 1..\ .N$
For ..
(1) If the number of training samples is not huge, then we can get:

$$\sigma_{test} = f(\sigma_{X_t}) = h(\sigma_{X_t})\beta = h(\sigma_{X_t})H_\sigma^T\left(\frac{\mathrm{I}}{C} + H_\sigma H_\sigma^T\right)^{-1}\sigma_{Y_t}$$
$$= \begin{bmatrix} K(\sigma_{X_t}, \sigma_{x1(t)}) \\ \cdots \\ K(\sigma_{X_t}, \sigma_{xN(t)}) \end{bmatrix}^T\left(\frac{\mathrm{I}}{C} + \Omega_{GELM}\right)^{-1}\sigma_{Y_t}$$

(2) Alternatively, if the number of training samples is huge, and we have:

$$\sigma_{test} = f(\sigma_{X_t}) = h(\sigma_{X_t})\beta = h(\sigma_{X_t})\left(\frac{\mathrm{I}}{C} + H_\sigma^T H_\sigma\right)^{-1}H_\sigma^T\sigma_{Y_t}$$

(3) Calculate Value-at-Risk:

$$VaR(1-\rho) = \overline{R} - \sigma_{test}c_\rho$$

End
End

Appendix B. Codes of 305 Stocks as Inputs

000001	000002	000009	000024	000027	000039	000046	000060	000061	000063	000069
000100	000156	000157	000333	000338	000400	000402	000413	000423	000425	000503
000538	000539	000559	000568	000581	000598	000623	000625	000629	000630	000651
000686	000689	000709	000712	000713	000715	000725	000728	000729	000738	000750
000768	000776	000778	000783	000792	000793	000800	000825	000826	000831	000839
000858	000876	000878	000883	000895	000897	000898	000899	000917	000937	000960
000963	000970	000983	000999	002001	002007	002008	002024	002038	002051	002065
002081	002129	002142	002146	002153	002202	002230	002236	002241	002252	002292
002294	002304	002310	002344	002353	002375	002385	002399	002410	002415	002422
002450	002456	002465	002470	002475	002500	002570	002594	002653	002673	300002
300003	300015	300017	300024	300027	300058	300059	300070	300104	300124	300133
300146	300251	600000	600005	600008	600009	600010	600011	600015	600016	600018
600019	600027	600028	600029	600030	600031	600036	600038	600039	600048	600050
600060	600066	600068	600085	600089	600100	600104	600108	600109	600111	600115
600118	600150	600153	600157	600166	600170	600177	600188	600196	600208	600221
600252	600256	600271	600276	600277	600309	600315	600316	600317	600332	600340
600348	600352	600362	600369	600372	600373	600383	600398	600406	600415	600485
600489	600497	600516	600518	600519	600535	600547	600549	600570	600578	600583
600585	600588	600597	600600	600633	600637	600642	600648	600649	600660	600663
600674	600688	600690	600703	600705	600717	600718	600739	600741	600783	600795
600804	600809	600827	600837	600839	600863	600867	600873	600875	600886	600887
600893	600900	600998	600999	601006	601009	601018	601088	601098	601099	601100
601101	601111	601117	601118	601158	601166	601168	601169	601179	601186	601216
601225	601231	601238	601258	601268	601288	601299	601311	601313	601318	601328
601333	601336	601339	601369	601377	601388	601390	601398	601515	601518	601519
601555	601558	601566	601567	601579	601600	601601	601607	601618	601628	601633
601668	601669	601688	601699	601727	601766	601788	601789	603806	603988	603993
603998	900901	900902	900903	900904	900905	900906	900907			

References

Abad, P., Benito, S., López, C., 2014. A comprehensive review of value at risk methodologies. Span. Rev. Financ. Econ. 12 (1), 15–32.
Alexandre, E., Cuadra, L., Salcedo-Sanz, S., Pastor-Sánchez, A., Casanova-Mateo, C., 2015. Hybridizing extreme learning machines and genetic algorithms to select acoustic features in vehicle classification applications. Neurocomputing 152, 58–68.
Angelidis, T., Benos, A., Degiannakis, S., 2004. The use of garch models in var estimation. Stat. Methodol. 1 (1), 105–128.
Arouri, M.E.H., Hammoudeh, S., Lahiani, A., Nguyen, D.K., 2012. Long memory and structural breaks in modeling the return and volatility dynamics of precious metals. Q. Rev. Econ. Financ. 52 (2), 207–218.
Babikir, A., Gupta, R., Mwabutwa, C., Owusu-Sekyere, E., 2012. Structural breaks and garch models of stock return volatility: the case of south africa. Econ. Model. 29 (6), 2435–2443.
Baillie, R.T., Bollerslev, T., Mikkelsen, H.O., 1996. Fractionally integrated generalized autoregressive conditional heteroskedasticity. J. Econ. 74 (1), 3–30.
Bauwens, L., Laurent, S., Rombouts, J.V., 2006. Multivariate garch models: a survey. J. Appl. Econ. 21 (1), 79–109.
Berman, J.J., 2013. Principles of Big Data: Preparing, Sharing, and Analyzing Complex Information. Newnes.
Bollerslev, T., 1986. Generalized autoregressive conditional heteroskedasticity. J. Econ. 31 (3), 307–327.
Bollerslev, T., Mikkelsen, H.O., 1996. Modeling and pricing long memory in stock market volatility. J. Econ. 73 (1), 151–184.
Box, G.E., Jenkins, G.M., 1976. Time Series Analysis. Forecasting and ControlHolden-Day Series in Time Series Analysis Revised ed.. Holden-Day, San Francisco.
Charfeddine, L., 2014. True or spurious long memory in volatility: further evidence on the energy futures markets. Energy policy 71, 76–93.
Chen, C.P., Zhang, C.-Y., 2014. Data-intensive applications, challenges, techniques and technologies: a survey on big data. Inf. Sci. 275, 314–347.
Cheng, W.-H., Hung, J.-C., 2011. Skewness and leptokurtosis in garch-typed var estimation of petroleum and metal asset returns. J. Empir. Financ. 18 (1), 160–173.
Chkili, W., Hammoudeh, S., Nguyen, D.K., 2014. Volatility forecasting and risk management for commodity markets in the presence of asymmetry and long memory. Energy Econ. 41, 1–18.
Christensen, B.J., Nielsen, M.Ø., Zhu, J., 2010. Long memory in stock market volatility and the volatility-in-mean effect: the fiegarch-m model. J. Empir. Financ. 17 (3), 460–470.
Christoffersen, P.F., 1998. Evaluating interval forecasts. Int. Econ. Rev. 39 (4), 841–862.
Chung, C.-F., 1999. Estimating the fractionally integrated garch model. Working paper, National Taiwan University, Taiwan.
Cortes, C., Vapnik, V., 1995. Support-vector networks. Mach. Learn. 20 (3), 273–297.
Davidson, J., 2004. Moment and memory properties of linear conditional heteroscedasticity models, and a new model. J. Bus. Econ. Stat. *22* (1), 16–29.
Deng, W., Zheng, Q., Chen, L., 2009. . Regularized extreme learning machine. In: Computational Intelligence and Data Mining, IEEE Symposium on. IEEE, 2009, pp. 389–395.
Dickey, D.A., Fuller, W.A., 1981. Likelihood ratio statistics for autoregressive time series with a unit root. Econometrica 49 (4), 1057–1072.
Ding, Z., Granger, C.W., Engle, R.F., 1993. A long memory property of stock market returns and a new model. J. Empir. Financ. 1 (1), 83–106.
Dionne, G., Pacurar, M., Zhou, X., 2015. Liquidity-adjusted intraday value at risk modeling and risk management: an application to data from deutsche börse. J. Bank. Financ. 59, 202–219.
Du, D., Li, K., Li, X., Fei, M., Wang, H., 2014. A multi-output two-stage locally regularized model construction method using the extreme learning machine. Neurocomputing 128, 104–112.
Engle, R., 1982. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. Econometrica 50 (4), 987–1007.
Engle, R., 2002. Dynamic conditional correlation: a simple class of multivariate generalized autoregressive conditional heteroskedasticity models. J. Bus. Econ. Stat. *20* (3), 339–350.
Engle, R.F., Bollerslev, T., 1986. Modelling the persistence of conditional variances. Econ. Rev. 5 (1), 1–50.
Engle, R.F., Manganelli, S., 2004. Caviar: conditional autoregressive value at risk by regression quantiles. J. Bus. Econ. Stat. *22* (4), 367–381.
Franses, P.H., Van Dijk, D., 2000. Non-Linear Time Series Models in Empirical Finance. Cambridge University Press.
Giot, P., Laurent, S., 2004. Modelling daily value-at-risk using realized volatility and arch typemodels. J. Empir. Financ. 11 (3), 379–398.
Glosten, L.R., Jagannathan, R., Runkle, D.E., 1993. On the relation between the expected value and the volatility of the nominal excess return on stocks. J. Financ. 48 (5), 1779–1801.
He, K., Lai, K.K., Yen, J., 2012. Ensemble forecasting of value at risk via multi resolution analysis based methodology in metals markets. Expert Syst. Appl. 39 (4), 4258–4267.
Hinton, G., Salakhutdinov, R., 2006. Reducing the dimensionality of data with neural networks. Science 313 (5786), 504–507.
Hou, A., Suardi, S., 2012. A nonparametric garch model of crude oil price return volatility. Energy Econ. 34 (2), 618–626.
Hou, A.J., 2013. Asymmetry effects of shocks in chinese stock markets volatility: a generalized additive nonparametric approach. J. Int. Financ. Mark., Institutions Money 23, 12–32.
Huang, G.-B., Wang, D.H., Lan, Y., 2011. Extreme learning machines: a survey. Int. J. Mach. Learn. Cybern. 2 (2), 107–122.
Huang, G.-B., Zhou, H., Ding, X., Zhang, R., 2012. Extreme learning machine for regression and multiclass classification. Syst., Man, Cybern., Part B: Cybern., IEEE Trans. 42 (2), 513–529.
Huang, G.-B., Zhu, Q.-Y., Siew, C.-K., 2006. Extreme learning machine: theory and applications. Neurocomputing 70 (1), 489–501.
Jorion, P., 1995. Predicting volatility in the foreign exchange market. J. Financ. 50 (2), 507–528.
Kim, M., Lee, S., 2016. Nonlinear expectile regression with application to value-at-risk and expected shortfall estimation. Comput. Stat. Data Anal. *94*, 1–19.
Koenker, R., Bassett, G., 1978. Regression quantiles. Econometrica 46 (1), 33–50.
Kupiec, P., 1995. Techniques for verifiying the accuracy of risk management models. J. Derviatives 3, 73–84.
LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. Neural Comput. 1 (4), 541–551.
Lowe, D., 1989. Adaptive radial basis function nonlinearities, and the problem of generalisation. Artif. Neural Netw. 313, 171–175.
Nelson, D.B., 1991. Conditional heteroskedasticity in asset returns: a new approach. Econ.: J. Econ. Soc. 1, 347–370.
Nikolaev, N.Y., Boshnakov, G.N., Zimmer, R., 2013. Heavy-tailed mixture garch volatility modeling and value-at-risk estimation. Expert Syst. Appl. 40 (6), 2233–2243.
Orhan, M., Köksal, B., 2012. A comparison of garch models for var estimation. Expert Syst. Appl. 39 (3), 3582–3592.
Phillips, P.C., Perron, P., 1988. Testing for a unit root in time series regression. Biometrika 75 (2), 335–346.
Siburg, K.F., Stoimenov, P., Weiß, G.N., 2015. Forecasting portfolio-value-at-risk with nonparametric lower tail dependence estimates. J. Bank. Financ. *54*, 129–140.
Smith, R.L., 1989. Extreme value analysis of environmental time series: an application to trend detection in ground-level ozone. Stat. Sci. 4 (4), 367–377.
Tse, Y.K., 1998. The conditional heteroscedasticity of the yen-dollar exchange rate. J. Appl. Econ. 13 (1), 49–55.
Wang, H.J., Li, D., He, X., 2012. Estimation of high conditional quantiles for heavy-tailed distributions. J. Am. Stat. Assoc. 107 (500), 1453–1464.
Wang, X., Wu, C., Xu, W., 2015. Volatility forecasting: the role of lunch-break returns, overnight returns, trading volume and leverage effects. Int. J. Forecast. 31 (3), 609–619.
Williams, D.R.G.H.R., Hinton, G., 1986. Learning representations by back-propagating errors. Nature 323, 533–536.
Youssef, M., Belkacem, L., Mokni, K., 2015. Value-at-risk estimation of energy commodities: a long-memory garch–evt approach. Energy Econ. 51, 99–110.
Zou, H., Yuan, M., 2008. Composite quantile regression and the oracle model selection theory. Ann. Stat. 36 (3), 1108–1126.

Heng-Guo Zhang is a lecturer at Department of Finance and Economics, Shandong University of Science and Technology, China and is currently pursuing his PH. D. degree in Ocean University of China. He received his B.S degree and M.S. degree from Sun Yat-sen University, China. His research interests include finance, stochastic mathematics and machine learning. He can be contracted at zszhg@126.com.

Chi-Wei Su is a professor in the School of Economics at Ocean University of China. His major is the applied time series analysis.

Yan Song received the B. S. degree and M. S. degree in electronic engineering from the Ocean University of China (OUC), where she is currently working toward Ph. D. degree. Her research interests include machine learning and autonomous navigation for mobile robots.

Shuqi Qiu is currently working toward the M.S. degree at Ocean University of China. He received his B.S. degree in Ocean University of China. His research interests include machine learning and underwater image processing.

Ran Xiao is a PhD student working at the University of Technology Sydney Business School. She received her Master's degree from Chinese Academy of Social Sciences. During her PhD program, she has been working on empirical market microstructure and asset pricing of Emerging Market Currencies.

Fei Su is a doctoral student in Finance at the University of Technology, Sydney. Before coming to UTS, he completed two postgraduate programmes: M.A. in Economics from University of Macau, Macau and M.SC. in Statistics from Lund University, Sweden. The topic of his Ph.D. dissertation is microstructure and financial econometrics. He can be contracted at fei.su@uts.edu.au.