Neural Computing and Applications (2019) 31:5641–5651
https://doi.org/10.1007/s00521-018-3385-5

ORIGINAL ARTICLE

Tensor extreme learning design via generalized Moore–Penrose inverse and triangular type-2 fuzzy sets

Sharina Huang[1] · Guoliang Zhao[2] · Minghao Chen[1]

Received: 9 August 2017 / Accepted: 21 February 2018 / Published online: 1 March 2018

Abstract
A tensor-based extreme learning machine is proposed, which is referred to as tensor-based type-2 extreme learning machine (TT2-ELM). In contrast to the work on ELM, regularized ELM (RELM), weighted regularized ELM (WRELM) and least squares support vector machine (LS-SVM), which are the most often used learning algorithm in *regression* problems, TT2-ELM *adopts* the tensor structure to *construct the ELM* for type-2 fuzzy sets, Moore–Penrose inverse of tensor is used to obtain the tensor regression result. No further type-reduction method is needed to obtain the coincide type-1 fuzzy sets, and type-2 fuzzy structure can be seamlessly incorporated into the ELM scheme. Experimental results are carried out on two Sinc functions, a nonlinear system identification problem and four real-world regression problems, results show that TT2-ELM performs at competitive level of generalized performance as the ELM, RELM, WRELM and LS-SVM on the small- and moderate-scale data sets.

Keywords Extreme learning machine · Type-2 fuzzy sets · Tensor regression · Fuzzy modelling · Moore–Penrose inverse · Einstein product

1 Introduction

Extreme learning machine (ELM) has been attracting much attention because of its excellent performance in training speed and predicting accuracy [1–4]. To improve the generalization ability of ELM, varieties of ELM variants have been proposed, such as RELM [2], WRELM [5], summation wavelet ELM [6], and optimally pruned extreme learning machine (OP-ELM) [7], et al. OP-ELM utilizes least angle regression and leave-one-out validation method to afford enhanced robustness. Tikhonov-regularized OP-ELM (TROP-ELM) is a newly proposed method, where ℓ_1 penalty, ℓ_2 penalty, and a variable ranking method named multiresponse sparse regression [8] are applied to the hidden layer and the regression weights, respectively [9]. A supervised algorithm with kernel method, which randomly selects a subset of the available data samples as support vectors, referred to as the reduced kernel ELM is studied by Deng [10]. ELM is also a useful tool for large-scale data analysis, especially in dimensionality reduction in big dimensional data [11, 12]. As far as we know, all the regression problems of ELM are solved by Moore–Penrose (M–P) inverse of matrices. There exists no tensor-based ELM, since the usage of the type-2 fuzzy sets in the hidden layer mapping leads to a tensor regression problem, and little work has been down yet on ELM with type-2 fuzzy hidden layer mapping.

Modern networks (such as internet traffic, telecommunication records, mobile location and large-scale social networks) generate massive amounts of data, the data present multiple aspects and high dimensionality. Such data are complex for matrix form and easy for multi-way arrays or tensor, ELM and its variants do not suit for the tensor data with high dimensionality. Consequently, tensor

✉ Sharina Huang
hsrn1982@163.com

Guoliang Zhao
guoliang.zhao.cn@ieee.org

Minghao Chen
chenmh130264@aliyun.com

[1] School of Science, Harbin Institute of Technology, Harbin 150001, People's Republic of China

[2] College of Electronic Information Engineering, Inner Mongolia University, Hohhot 010021, People's Republic of China

decomposition such as Tucker decomposition becomes important tool for tensor summarization and tensor analysis. One major challenge of tensor decomposition is how to deal with high-dimensional or sparse data with most of the entries of the data are zero. To address the memory overflow problem, Kolda and Sun [13] *propose a* memory-efficient Tucker method, which adaptively selects the right execution strategy during the decomposition. Kolda also presents an overview of higher-order tensor decomposition, higher-order tensor applications and well-known software for higher-order tensor [14]. Incomplete tensor completion and data factorization method called CANDECOMP/PARAFAC (CP) weighted optimization [15] is equivalent to solve the weighted least squares problem. The CP expresses a tensor as the sum of component rank-one tensors. However, doing CP decomposition can be difficult due to alternating least squares optimization. To avoid this problem, a kind of gradient-based optimization method is used to do the CP decomposition [16]. A new and highly parallelizable method (ParCube) for speeding up tensor decomposition that is well-suited to producing sparse approximations in data mining applications is proposed, the ParCube can scale to truly large data sets [17]. In [18], a sparse tensor decomposition algorithm that incorporates sparsity into the estimation of decomposition components is proposed, and the method provides a general way to solve the high-dimensional latent variable models. An alternating direction method of multiplier-based method [19] is developed to solve the tensor completion problem and its low-rankness approximation problem. A new algorithm [20] is proposed to find the generalized singular value decomposition (SVD) of two matrices with the same number of columns, and the sensitivity of the algorithm to the matrix entries' errors can be suppressed. In [21], a highly efficient and scalable estimation algorithm for a class of spectral regularized matrix regression is developed, and model selection algorithm along the regularization path is finished by a degree of freedom formula. In [22], an algorithm called alternating SVD is proposed for the computation of a best rank-one approximation of tensors, the convergence of the alternating SVD is guaranteed.

Tensor regression problem is relatively new comparing with matrix regression problem. There exist a few papers on this topic, to name a few, a new tensor decomposition method (based on the fact that a tensor group endowed with the Einstein product is isomorphic to the general linear group of degree n) which has been proven to be related to the well-known canonical polyadic decomposition and tensor SVD [23] is introduced, and multilinear systems are solvable when the tensor has an odd number of modes or when the tensor has distinct dimensions in each modes. The M–P inverse of tensors with the Einstein product is first proposed in [24], and some multilinear systems' general solution or the minimum-norm least squares solution are given based on the M–P inverse of tensors. In [25], a method to compute the M–P inverse of tensors is proposed. Reverse order laws for several generalized inverses of tensors are also presented. In addition, general solutions of multilinear systems of tensors using the generalized M–P inverse of tensors with the Einstein product are discussed. In [26], the fundamental theorem of linear algebra for matrix space is extended to tensor space, and the relationship between the minimum-norm least squares solution of a multilinear system and the weighted M–P inverse of its coefficient tensor is studied.

Inspired by the *abovementioned* topics, in this paper, we extended the ELM in three folds: (1) Triangular type-2 fuzzy set is used to formulate the uncertainty; (2) Tensor structure is adopted to construct the ELM; (3) Only type-2 fuzzy membership functions are needed in tensor-based type-2 ELM (TT2-ELM), extreme learning results are solved by tensor-based regression problem in which classical M–P inverse of matrix is replaced by M–P inverse of tensor, and the type-reduction in type-2 fuzzy set is avoided. As far as we know, there exists no work on the extreme learning machine scheme which uses tensor to train the model, let alone on the type-2 fuzzy set-based ELM.

This paper is organized as follows. The basic concepts of type-2 fuzzy sets are described in Sect. 2. In Sect. 3, a tensor-based type-2 ELM is proposed, and proof of the tensor regression for TT2-ELM is presented at the end of Sect. 3. Section 4 provides three types of examples to illustrate the utilization of the proposed algorithm. Finally, conclusions and *future work* are given in Sect. 5.

2 Triangular type-2 fuzzy sets

Type-2 fuzzy set is introduced as an extension of the ordinary fuzzy set, the membership grades of type-2 fuzzy sets themselves are type-1 fuzzy sets. For a type-2 fuzzy set $\tilde{A}$, the type-2 membership function $\mu_{\tilde{A}}(x,u)$ with $x \in X$ and $u \in J_x \subseteq [0,1]$ is described as $\tilde{A} = \{((x,u),\mu_{\tilde{A}}(x,u)) | \forall x \in X, \forall u \in J_x \subseteq [0,1]\}$ [27], where $0 \le \mu_{\tilde{A}}(x,u) \le 1$. Another representation form of the type-2 fuzzy set $\tilde{A}$ can be expressed as

$$\tilde{A} = \int_{x\in X}\int_{u\in J_x} \mu_{\tilde{A}}(x,u)/(x,u), \quad J_x \subseteq [0,1], \tag{1}$$

where $\int\int$ denotes union over all admissible x and u, J_x is the primary membership of x (usually a closed interval of real numbers that is contained within [0, 1]).

To the triangular type-2 fuzzy sets, the partition of the primary domain X in the discrete form is denoted as

$\{x_1, x_2, \ldots, x_n\}$, let f_k be the secondary membership function of x_k, and it can be denoted by

$$f_k(u) = \max\left\{0, \min\left\{\frac{u-\underline{\mu}_k}{\hat{\mu}_k-\underline{\mu}_k}, \frac{u-\bar{\mu}_k}{\hat{\mu}_k-\bar{\mu}_k}\right\}\right\}, \tag{2}$$

where $\bar{\mu}_k > \hat{\mu}_k > \underline{\mu}_k$ are upper, principal and lower membership grades for $k = 1, 2, \ldots, n$, respectively.

For a given testing data set denoted as $\{D_t\}_{t=1}^N$, where $D_t = (\boldsymbol{x}_t, y_t), \boldsymbol{x}_t = (x_{t1}, x_{t2}, \ldots, x_{tK}) \in \mathbb{R}^K$ and $y_t \in \mathbb{R}$. A lower membership function matrix $\underline{\Phi} \in \mathbb{R}^{N\times 2\times L\times 1}$ can be constructed to approximate the relationship between input $\boldsymbol{x}_t$ and a desired output y_t via lower membership functions,

$$\underline{\Phi}_{:,:,1,1} = \begin{bmatrix} \underline{\mu}(\boldsymbol{w}_{11}\boldsymbol{x}_1 + b_{11}) & \underline{\mu}(\boldsymbol{w}_{12}\boldsymbol{x}_1 + b_{12}) \\ \vdots & \vdots \\ \underline{\mu}(\boldsymbol{w}_{11}\boldsymbol{x}_N + b_{11}) & \underline{\mu}(\boldsymbol{w}_{12}\boldsymbol{x}_N + b_{12}) \end{bmatrix},$$
$$\vdots$$
$$\underline{\Phi}_{:,:,L,1} = \begin{bmatrix} \underline{\mu}(\boldsymbol{w}_{L1}\boldsymbol{x}_1 + b_{L1}) & \underline{\mu}(\boldsymbol{w}_{L2}\boldsymbol{x}_1 + b_{L2}) \\ \vdots & \vdots \\ \underline{\mu}(\boldsymbol{w}_{L1}\boldsymbol{x}_N + b_{L1}) & \underline{\mu}(\boldsymbol{w}_{L2}\boldsymbol{x}_N + b_{L2}) \end{bmatrix},$$

where b_{il} and $\boldsymbol{w}_{il} = [w_{i1}, w_{i2}, \ldots, w_{iK}]$ $(i = 1, 2, \ldots, L, l = 1, 2)$ are random generated bias and input weights, respectively. Similarly, the principal and upper membership function matrices $\hat{\Phi}, \bar{\Phi} \in \mathbb{R}^{N\times 2\times L\times 1}$ are

$$\hat{\Phi}_{:,:,1,2} = \begin{bmatrix} \hat{\mu}(\boldsymbol{w}_{11}\boldsymbol{x}_1 + b_{11}) & \hat{\mu}(\boldsymbol{w}_{12}\boldsymbol{x}_1 + b_{12}) \\ \vdots & \vdots \\ \hat{\mu}(\boldsymbol{w}_{11}\boldsymbol{x}_N + b_{11}) & \hat{\mu}(\boldsymbol{w}_{12}\boldsymbol{x}_N + b_{12}) \end{bmatrix},$$
$$\vdots$$
$$\hat{\Phi}_{:,:,L,2} = \begin{bmatrix} \hat{\mu}(\boldsymbol{w}_{L1}\boldsymbol{x}_1 + b_{L1}) & \hat{\mu}(\boldsymbol{w}_{L2}\boldsymbol{x}_1 + b_{L2}) \\ \vdots & \vdots \\ \hat{\mu}(\boldsymbol{w}_{L1}\boldsymbol{x}_N + b_{L1}) & \hat{\mu}(\boldsymbol{w}_{L2}\boldsymbol{x}_N + b_{L2}) \end{bmatrix},$$
$$\bar{\Phi}_{:,:,1,3} = \begin{bmatrix} \bar{\mu}(\boldsymbol{w}_{11}\boldsymbol{x}_1 + b_{11}) & \bar{\mu}(\boldsymbol{w}_{12}\boldsymbol{x}_1 + b_{12}) \\ \vdots & \vdots \\ \bar{\mu}(\boldsymbol{w}_{11}\boldsymbol{x}_N + b_{11}) & \bar{\mu}(\boldsymbol{w}_{12}\boldsymbol{x}_N + b_{12}) \end{bmatrix},$$
$$\vdots$$
$$\bar{\Phi}_{:,:,L,3} = \begin{bmatrix} \bar{\mu}(\boldsymbol{w}_{L1}\boldsymbol{x}_1 + b_{L1}) & \bar{\mu}(\boldsymbol{w}_{L2}\boldsymbol{x}_1 + b_{L2}) \\ \vdots & \vdots \\ \bar{\mu}(\boldsymbol{w}_{L1}\boldsymbol{x}_N + b_{L1}) & \bar{\mu}(\boldsymbol{w}_{L2}\boldsymbol{x}_N + b_{L2}) \end{bmatrix}.$$

A 4-tensor $\Phi \in \mathbb{R}^{N\times 2\times L\times 3}$ can be constructed by $\underline{\Phi}_{:,:,:,1}, \hat{\Phi}_{:,:,:,2}$ and $\bar{\Phi}_{:,:,:,3}$. In the following, we denote tensor Φ by $\mathcal{A}$ for the following tensor regression section.

3 Tensor-based type-2 ELM (TT2-ELM)

In this section, we establish the basic structure of TT2-ELM. First, tensor operations and inverse of tensor are introduced. In [24], a generalized M–P inverse of even-order tensor is introduced, which is formulated as follows.

Definition 1 Let $\mathcal{A} \in \mathbb{R}^{I_{1\ldots N}\times K_{1\ldots N}}$, the tensor $\mathcal{X} \in \mathbb{R}^{K_{1\ldots N}\times J_{1\ldots N}}$ is called the M–P inverse of $\mathcal{A}$, denoted by $\mathcal{A}^+$, and it satisfied the following four tensor equations:

(1) $\mathcal{A} *_N \mathcal{X} *_N \mathcal{A} = \mathcal{A}$;
(2) $\mathcal{X} *_N \mathcal{A} *_N \mathcal{X} = \mathcal{X}$;
(3) $(\mathcal{A} *_N \mathcal{X})^* = \mathcal{A} *_N \mathcal{X}$;
(4) $(\mathcal{X} *_N \mathcal{A})^* = \mathcal{X} *_N \mathcal{A}$,

where $I_{1\ldots N} = I_1 \times \cdots \times I_N, J_{1\ldots N} = J_1 \times \cdots \times J_N, K_{1\ldots N} = K_1 \times \cdots \times K_N, *_N$ is the Einstein product[1] of tensor, and $\mathcal{A} *_N \mathcal{X}$ is defined as

$$(\mathcal{A} *_N \mathcal{X})_{i_{1\ldots N}j_{1\ldots N}} = \sum_{k_{1\ldots N}} a_{i_{1\ldots N}k_{1\ldots N}} x_{k_{1\ldots N}j_{1\ldots N}}. \tag{3}$$

For an even-order tensor $\mathcal{A} \in \mathbb{R}^{I_{1\ldots N}\times I_{1\ldots N}}$, Brazell et al. [28] shown that the inverse of tensor $\mathcal{A}$ exists if there exists tensor $\mathcal{X} \in \mathbb{R}^{I_{1\ldots N}\times I_{1\ldots N}}$ such that $\mathcal{A} *_N \mathcal{X} = \mathcal{X} *_N \mathcal{A} = \mathcal{I}$, denoted by $\mathcal{A}^{-1}$. Clearly, if $\mathcal{A}$ is invertible, then $\mathcal{A}^{(i)} = \mathcal{A}^+ = \mathcal{A}^{-1}$, where $\mathcal{A}^{(i)}$ satisfies the ith condition of Definition 1. For a tensor $\mathcal{S} \in \mathbb{R}^{I_{1\ldots N}\times J_{1\ldots N}}$, it follows from Definition 1 that $\mathcal{S}^+ \in \mathbb{R}^{J_{1\ldots N}\times I_{1\ldots N}}$ and

$$(\mathcal{S}^+)_{J_{1\ldots N}\times I_{1\ldots N}} = (\mathcal{S})^+_{I_{1\ldots N}\times J_{1\ldots N}}, \tag{4}$$

where

$$s^+ = \begin{cases} s^{-1}, & \text{if } (I_1, \ldots, I_N) = (J_1, \ldots, J_N), s \neq 0, \\ 0, & \text{if } (I_1, \ldots, I_N) \neq (J_1, \ldots, J_N), \\ & \text{or } (I_1, \ldots, I_N) = (J_1, \ldots, J_N), s = 0. \end{cases}$$

and $s = (\mathcal{S})_{i_{1\ldots N}\times j_{1\ldots N}}$. It is easy to see that $\mathcal{S} *_N \mathcal{S}^+$ and $\mathcal{S}^+ *_N \mathcal{S}$ are diagonal tensors whose diagonal entries are 1 or 0.

For a tensor $\mathcal{A} \in \mathbb{R}^{J_{1\ldots N}\times I_{1\ldots N}}$, if $\mathcal{A} *_N \mathcal{A}^* = \mathcal{A}^* *_N \mathcal{A} = \mathcal{I}$, then $\mathcal{A}$ is called the orthogonal tensor. The SVD of a tensor $\mathcal{A}$ has the form as $\mathcal{A} = \mathcal{U} *_N \mathcal{B} *_N \mathcal{V}^*$, where $\mathcal{U} \in \mathbb{R}^{I_{1\ldots N}\times I_{1\ldots N}}$ and $\mathcal{V} \in \mathbb{R}^{J_{1\ldots N}\times J_{1\ldots N}}$ are orthogonal tensors, and $\mathcal{B} \in \mathbb{R}^{J_{1\ldots N}\times I_{1\ldots N}}$ satisfies Eq. (4). For some 'square' case, that is $I_k = J_k, k = 1, 2, \ldots, N$, the existence for the inverse of $\mathcal{A}$ is given by an isomorphism map $\mathscr{L} : \mathbb{T}_{I_{1\ldots N}\times I_{1\ldots N}}(\mathbb{R}) \to \mathbb{M}_{(\prod_{i=1}^N I_i)\times(\prod_{i=1}^N I_i)}(\mathbb{R})$, where the

[1] TPROD efficiently allows any type of tensor product between two multi-dimensional arrays, which can be downloaded from Mathworks's file exchange.

Einstein product is used in the tensor group $\mathbb{T}_{I_{1\ldots N}\times I_{1\ldots N}}(\mathbb{R})$, and the matrix group $\mathbb{M}_{(\prod_{i=1}^{N} I_i)\times(\prod_{i=1}^{N} I_i)}(\mathbb{R})$ used the usual matrix product [28]. The M–P inverse of $\mathcal{A}\in\mathbb{R}^{I_{1\ldots N}\times J_{1\ldots N}}$ exists and is unique [24], and $\mathcal{A}^{+}=\mathcal{V}*_N\mathcal{S}^{+}*_N\mathcal{U}^{*}$ (Pseudo code of M–P inverse of *even-order tensor* $\mathcal{A}$ is listed in Algorithm 1).

Algorithm 1 M-P inverse of even order tensor $\mathcal{A}$

```
Require: Input: Tensor $\mathcal{A}$, $I_{1\ldots N}$ and $J_{1\ldots N}$.
Ensure: $N$ is even, $I=\prod_{i=1}^{N} I_i$, $J=\prod_{i=1}^{N} J_i$.
1: Reshape tensor $\mathcal{A}\in\mathbb{R}^{I_{1\ldots N}\times J_{1\ldots N}}$ into a matrix $A\in\mathbb{R}^{I\times J}$.
2: Perform SVD on matrix $A$, $A$ is decomposed as $A=USV^*$, where $V^*$ is the conjugate transpose of $V$.
3: Reshape the matrices $U,S$ and $V^*$ into the corresponding order tensors $\mathcal{U},\mathcal{S}$ and $\mathcal{V}^*$, respectively.
4: Output: Calculate the M-P inverse of $\mathcal{A}$ via $\mathcal{A}^{+}=\mathcal{V}*_N\mathcal{S}^{+}*_N\mathcal{U}^{*}$.
```

TT2-ELM[2] is a neural network with a single-layer feed forward, structure of TT2-ELM is shown in Fig. 1. For a given testing data set denoted as $\{D_t\}_{t=1}^{N}$, where $D_t=(\boldsymbol{x}_t,y_t),\boldsymbol{x}_t=(x_{t1},x_{t2},\ldots,x_{tK})\in\mathbb{R}^{K}$ and $y_t\in\mathbb{R}$. TT2-ELM can be constructed to approximate the relationship between input $\boldsymbol{x}_t$ and desired output y_t. For N training patterns, the TT2-ELM's mathematical model can be formulated in a compact form

$$\mathcal{A}*_N\mathcal{X}=\mathcal{Y},\tag{5}$$

where $\mathcal{X}\in\mathbb{R}^{J_{1\ldots 2}},\mathcal{Y}\in\mathbb{R}^{I_{1\ldots 2}}$, and $\mathcal{A}\in\mathbb{R}^{I_{1\ldots 2}\times J_{1\ldots 2}}$ can be reshaped via Φ with dimension $N\times 2\times L\times 3$. We can find the value of output weight $\mathcal{X}$ by solving the linear tensor Eq. (5).

Remark 1 The regression tensor's dimension is $N\times 2\times L\times 3$, we can deduce that $\mathcal{Y}\in R^{N\times 2}$ based on Einstein product. A vector should be copied once to meet the dimension requirement.

In the following, the multilinear system Eq. (5) is generalized as follows

$$\mathcal{A}*_N\mathcal{X}=\mathcal{Y},\tag{6}$$

where $\mathcal{A}\in\mathbb{R}^{I_{1\ldots N}\times J_{1\ldots N}},\mathcal{X}\in\mathbb{R}^{J_{1\ldots N}}$ and $\mathcal{Y}\in\mathbb{R}^{I_{1\ldots N}}$. It is obviously that (5) is the special case of (6) when $N=2$. The following theorem holds for the multilinear system Eq. (6).

Theorem 1 *The multilinear system* (6) *is solvable if and only if* $\mathcal{X}=\mathcal{A}^{(1)}*_N\mathcal{Y}$ *is a solution of the TT2-ELM with the system Eq.* (6), *where* $\mathcal{A}^{(1)}$ *is a solution of* $\mathcal{A}*_N\mathcal{X}*_N\mathcal{A}=\mathcal{A}$. *If there doesn't exist any* $\mathcal{X}\in\mathbb{R}^{J_{1\ldots N}}$ *for Eq.* (6), *then there exists a minimum-norm solution* $\mathcal{X}=\mathcal{A}^{+}*_N\mathcal{Y}$ *to unsolvable multilinear system* (6), *and the norm is defined as the Frobenius norm*

$$\|\cdot\|_F=\sqrt{\sum_{i_{1\ldots N}j_{1\ldots N}}|a_{i_{1\ldots N}j_{1\ldots N}}|^2}.$$

[2] TT2-ELM's source MATLAB code can be download directly from https://github.com/zggl/TriT2ELM.

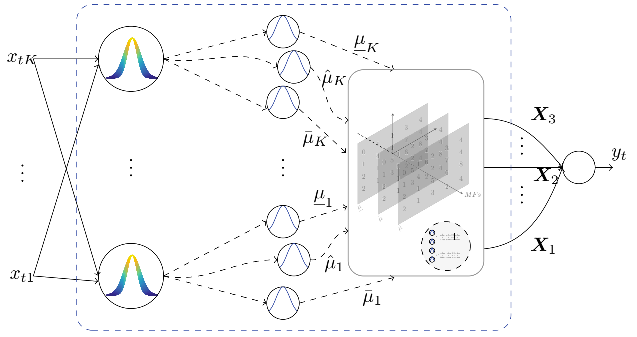

Fig. 1 Tensor-based type-2 ELM

Proof The proof is based on kernel theory and norm theory of tensor, which is slightly different from [24] (Theorem 4.1 (1)), it is more easily for us to understand the idea that how the tensor linear equation is incorporated into the extreme learning design framework. Since $\mathcal{A}*_N(\mathcal{A}^{(1)}*_N\mathcal{A})*_N\mathcal{Z}=\mathcal{A}*_N\mathcal{Z}$, where $\mathcal{Z}$ is an arbitrary tensor with suitable order, we knew that $(\mathcal{I}-\mathcal{A}^{(1)}*_N\mathcal{A})*_N\mathcal{Z}$ satisfies that $\mathcal{A}*_N(\mathcal{I}-\mathcal{A}^{(1)}*_N\mathcal{A})*_N\mathcal{Z}=0$. It means that $(\mathcal{I}-\mathcal{A}^{(1)}*_N\mathcal{A})*_N\mathcal{Z}$ belongs to the null space of $\mathcal{A}*_N\mathcal{X}=0$. Obviously, the general solution is

$$\mathcal{X}=\mathcal{A}^{(1)}*_N\mathcal{Y}+(\mathcal{I}-\mathcal{A}^{(1)}*_N\mathcal{A})*_N\mathcal{Z},\tag{7}$$

where $\mathcal{A}^{(1)}$ is the 1-inverse of $\mathcal{A}$. The general solution is constituted by two parts, one is from Eq. (6), the other is from null space of equation $\mathcal{A}*_N\mathcal{X}=0$.

For multilinear system (6), if the multilinear system (6) is unsolvable, there exists a minimum-norm solution to Eq. (6). Let $\mathcal{E}(\mathcal{X})=\|\mathcal{A}*_N\mathcal{X}-\mathcal{Y}\|_F^2,\mathcal{E}_1(\mathcal{X})=\mathrm{trace}((\mathcal{A}*_N\mathcal{X})^T*_N(\mathcal{A}*_N\mathcal{X})),\mathcal{E}_2(\mathcal{X})=-2\mathrm{trace}(\mathcal{Y}^T*_N(\mathcal{A}*_N\mathcal{X}))$, and $\mathcal{E}_3(\mathcal{X})=\mathrm{trace}(\mathcal{Y}^T*_N\mathcal{Y})$, it holds that $\mathcal{E}(\mathcal{X})=\mathcal{E}_1(\mathcal{X})+\mathcal{E}_2(\mathcal{X})+\mathcal{E}_3(\mathcal{X})$. After some tensor differential operations, we have that

$$\frac{\partial\mathcal{E}_1}{\partial\mathcal{X}}=2(\mathcal{A}^T*_N\mathcal{A})*_N\mathcal{X},\ \frac{\partial\mathcal{E}_2}{\partial\mathcal{X}}=-2\mathcal{A}^T*_N\mathcal{Y},\ \frac{\partial\mathcal{E}_3}{\partial\mathcal{X}}=0.\tag{8}$$

Noticed that a minimum-norm solution to Eq. (6) is equivalent to the following tensor optimization problem

$$\min_{\mathcal{X}}\ \mathcal{E}(\mathcal{X}).\tag{9}$$

The necessary condition for (9) is $\frac{\partial \mathcal{E}_1}{\partial \mathcal{X}} + \frac{\partial \mathcal{E}_2}{\partial \mathcal{X}} = 2((\mathcal{A}^T *_N \mathcal{A}) *_N \mathcal{X} - \mathcal{A}^T *_N \mathcal{Y}) = 0$, that is, the tensor equation

$$(\mathcal{A}^T *_N \mathcal{A}) *_N \mathcal{X} = \mathcal{A}^T *_N \mathcal{Y}. \tag{10}$$

Left hand of Eq. (10)'s gain tensor $\mathcal{A}^T *_N \mathcal{A}$ is a 'square tensor', tensor Eq. (10) is solvable. Using Eq. (7), we have

$$\begin{aligned}\mathcal{X} &= (\mathcal{A}^T *_N \mathcal{A})^{(1)} *_N \mathcal{A}^T *_N \mathcal{Y} \\ &\quad + (\mathcal{I} - (\mathcal{A}^T *_N \mathcal{A})^{(1)} *_N (\mathcal{A}^T *_N \mathcal{A})) *_N \mathcal{Z}.\end{aligned} \tag{11}$$

It holds that $\mathcal{X}$ is the minimizer when $\mathcal{Z} = 0$. We have $\mathcal{A} = \mathcal{A} *_N (\mathcal{A}^T *_N \mathcal{A})^{(1)} *_N (\mathcal{A}^T *_N \mathcal{A})$ by Corollary 2.14(1) [25], there also exists $(\mathcal{A}^T *_N \mathcal{A})^{(1)}$ such that $\mathcal{A}^+ = (\mathcal{A}^T *_N \mathcal{A})^{(1)} *_N \mathcal{A}^T$. Then the minimum-norm least squares solution can be obtained as $\mathcal{X} = \mathcal{A}^+ *_N \mathcal{Y}$. □

Remark 2 The tensor regression is necessary for the type-2 fuzzy set-based extreme learning scheme, since the tensor structure can incorporate the information of the secondary membership function directly, the type-reduction step which is the most important step in type-2 fuzzy inference is avoided, and type-2 fuzzy structure can be seamlessly incorporated into the ELM scheme.

Remark 3 The regression tensor $\mathcal{A} \in \mathbb{R}^{N\times 2\times L\times 3}$, the second dimension equals to 2. Thus, TT2-ELM needs two randomly generated weight vectors, any one of the vectors is randomly generated weight vectors which can be used by the RELM and WRELM.

4 Performance comparison

In this section, three types of examples are involved in algorithms' performance comparison. Two function approximation problems are presented in Sect. 4.1. A nonlinear system identification problem which has three input variables is provided in Sect. 4.2. In Sect. 4.3, four real-world regression problems in which the input variables range from 2 to 61 are given. Parameters sensitivity and stability analysis are carried out for different iteration number at the end of this section.

ELM, RELM and WRELM are applied to compare with TT2-ELM in the section. The RELM and WRELM use constraint on minimum norm of weight factor, while TT2-ELM's mathematical model only has constraint on minimum norm of error. Mathematical models of the four algorithms, that is, ELM, RELM, WRELM and TT2-ELM are listed in Table 1. The algorithms RELM and WRELM have two design parameters L and C, while ELM and TT2-ELM only presents one design parameter, the hidden layer neuron number L. All of the compared results are carried out by averaging with 1000 times. The triangular type-2 fuzzy sets are used in the TT2-ELM design. The principal membership grade is randomly generated, the lower and

Table 1 Mathematical models and its design parameters

Algorithm	Problem formulation	Design parameters
ELM	$\min_\beta \|\Phi\beta - Y\|_2^2$	L
RELM	$\min_\beta C\|\Phi\beta - Y\|_2^2 + \|\beta\|_2^2$	L, C
WRELM	$\min_\beta C\|W(\Phi\beta - Y)\|_2^2 + \|\beta\|_2^2$	L, C
TT2-ELM	$\min_\beta \|\Phi *_N \beta - \mathcal{Y}\|_F^2$	L

Table 2 Comparison results for the one-input Sinc function (12)

L	Algorithm	$C = 2^{-5}$		$C = 2^{-10}$		$C = 2^{-15}$		$C = 2^{-20}$	
		Training	Testing	Training	Testing	Training	Testing	Training	Testing
25	ELM	4.08e−04	4.76e−04	3.97e−04	4.72e−04	3.66e−04	4.46e−04	3.46e−04	4.08e−04
	RELM	7.45e−02	7.57e−02	4.38e−02	4.39e−02	1.40e−02	1.41e−02	7.73e−03	9.02e−03
	WRELM	7.56e−02	7.70e−02	4.62e−02	4.64e−02	1.49e−02	1.49e−02	7.89e−03	9.72e−03
	TT2-ELM	**8.02e−07**	**1.07e−06**	**5.99e−07**	**7.67e−07**	**1.05e−06**	**1.37e−06**	**8.81e−07**	**1.18e−06**
30	ELM	2.84e−04	3.41e−04	3.33e−04	3.94e−04	3.02e−04	3.65e−04	3.05e−04	3.57e−04
	RELM	7.34e−02	7.47e−02	4.01e−02	4.01e−02	1.23e−02	1.25e−02	6.99e−03	8.22e−03
	WRELM	7.34e−02	7.46e−02	3.94e−02	3.94e−02	1.26e−02	1.28e−02	7.56e−03	9.51e−03
	TT2-ELM	**2.13e−05**	**2.13e−05**	**2.19e−05**	**2.19e−05**	**2.18e−05**	**2.18e−05**	**2.52e−05**	**2.52e−05**
35	ELM	**2.48e−04**	**2.95e−04**	**2.60e−04**	**3.15e−04**	**2.47e−04**	**3.01e−04**	**2.58e−04**	**3.10e−04**
	RELM	7.25e−02	7.37e−02	3.52e−02	3.50e−02	1.14e−02	1.17e−02	6.67e−03	7.86e−03
	WRELM	7.25e−02	7.37e−02	3.55e−02	3.53e−02	1.16e−02	1.20e−02	7.10e−03	8.96e−03
	TT2-ELM	5.61e−03	5.62e−03	6.18e−03	6.19e−03	6.09e−03	6.09e−03	7.56e−03	7.57e−03

upper membership grades use two offset variables, one is 0.01, and the other is 0.05.

4.1 Modelling of Sinc functions

Consider one-input Sinc function

$$z = \frac{\sin x}{x}, \quad x \in [-10, 10]. \tag{12}$$

Table 2 shows the comparison results among ELM, RELM, WRELM and TT2-ELM. All the training data are randomly generated from interval $[-10, 10]$, while testing data are uniformly generated from interval $[-10, 10]$ with step equals to 0.01. TT2-ELM solves a regression problem $\mathcal{A}\mathcal{X} = \mathcal{Y}$ with $\mathcal{A} \in \mathbb{R}^{600\times 2\times L\times 3}$, $\mathcal{X} \in \mathbb{R}^{L\times 3}$, and $\mathcal{Y} \in \mathbb{R}^{600\times 2}$ ($L \in \{25, 30, 35\}$). It is needed to be pointed out that the data should be the repeat copies of the first column of the array if we only have a single column data, the column data should repeat once more to meet the dimension requirement of $\mathcal{Y}$ for TT2-ELM. When $L = 35$, the ELM outperforms the TT2-ELM on all the design parameters $\{2^{-5}, 2^{-10}, 2^{-15}, 2^{-20}\}$.

Consider two-input Sinc function

$$z = \frac{\sin x \cdot \sin y}{xy}, (x, y) \in [-\pi, \pi]^2. \tag{13}$$

Table 3 Comparison results for the two-input Sinc function (13)

L	Algorithm	$C = 2^{-5}$		$C = 2^{-10}$		$C = 2^{-15}$		$C = 2^{-20}$	
		Training	Testing	Training	Testing	Training	Testing	Training	Testing
20	ELM	4.54e−02	7.32e−02	4.54e−02	7.32e−02	4.54e−02	7.32e−02	4.54e−02	7.32e−02
	RELM	4.70e−02	7.06e−02	4.68e−02	7.08e−02	4.64e−02	7.11e−02	4.59e−02	7.19e−02
	WRELM	4.82e−02	7.13e−02	4.82e−02	7.12e−02	4.80e−02	7.11e−02	4.75e−02	7.08e−02
	TT2-ELM	**4.40e−02**	**6.74e−02**	**4.40e−02**	**6.74e−02**	**4.40e−02**	**6.74e−02**	**4.40e−02**	**6.74e−02**
60	ELM	4.45e−02	7.51e−02	4.45e−02	7.51e−02	4.45e−02	7.51e−02	4.45e−02	7.51e−02
	RELM	4.69e−02	7.07e−02	4.66e−02	7.09e−02	4.60e−02	7.16e−02	4.56e−02	7.25e−02
	WRELM	4.81e−02	7.12e−02	4.81e−02	7.12e−02	4.76e−02	7.08e−02	4.73e−02	7.07e−02
	TT2-ELM	**4.11e−02**	**6.70e−02**	**4.11e−02**	**6.70e−02**	**4.11e−02**	**6.70e−02**	**4.11e−02**	**6.70e−02**
75	ELM	4.45e−02	7.53e−02	4.44e−02	7.53e−02	4.44e−02	7.53e−02	4.44e−02	7.52e−02
	RELM	4.69e−02	7.07e−02	4.66e−02	7.09e−02	4.59e−02	7.17e−02	4.55e−02	7.27e−02
	WRELM	4.81e−02	7.12e−02	4.81e−02	7.12e−02	4.76e−02	7.08e−02	4.73e−02	7.06e−02
	TT2-ELM	**4.03e−02**	**6.73e−02**	**4.03e−02**	**6.73e−02**	**4.03e−02**	**6.73e−02**	**4.03e−02**	**6.73e−02**

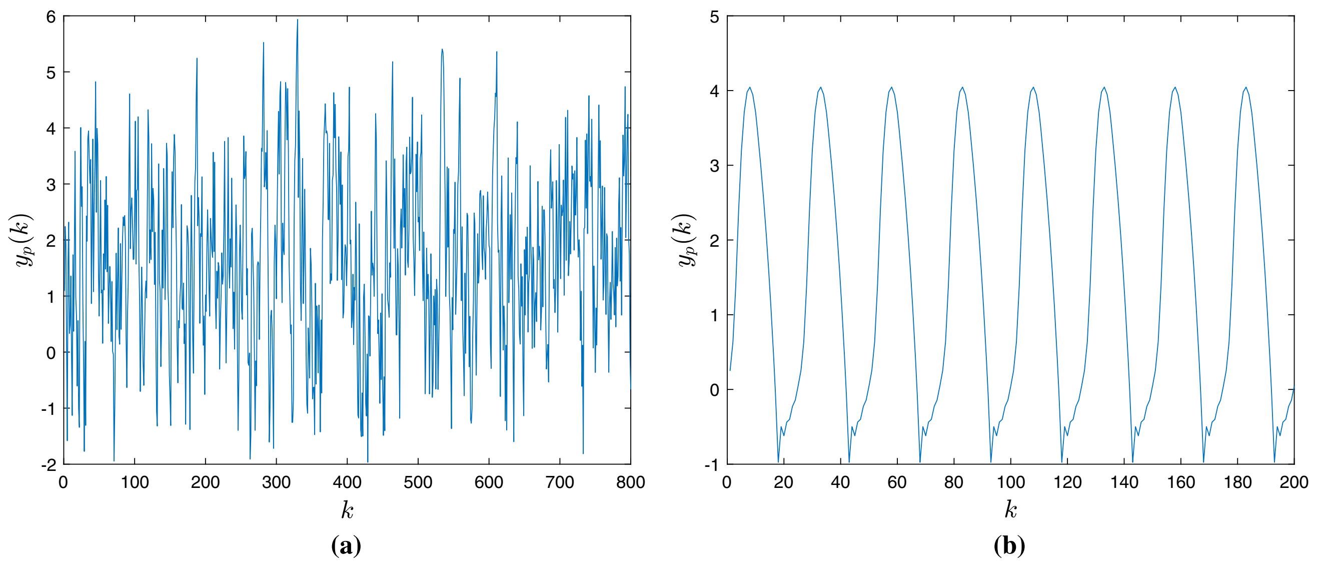

Fig. 2 Nonlinear system output. **a** Randomly generated control effort $u(k) \in \{-2, 2\}$ and **b** specified control effort $u(k) = \sin(2\pi k/25)$

Equally spaced 41×41 data pairs are used for training, and equally spaced 30×30 testing data are used to check generalization.

Table 3 lists the comparison results for the two-input Sinc function (13). Results show that TT2-ELM outperforms ELM, RELM and WRELM for all the cases. Combining the results which are listed in Tables 2 and 3, we can infer that the hidden neuron number L is important to each algorithm's training and testing precision, the hidden neuron number L must be lied in the certain range. The performance will be poor when the hidden neuron number L out of the range, and it will be tested again by the next subsection's nonlinear system identification problem.

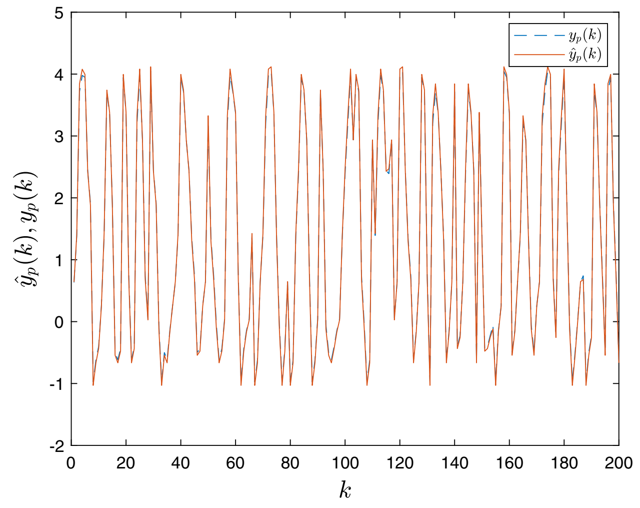

Fig. 3 Actual output and testing output using $u(k) = \sin(2\pi k/25)$ with $L = 35$

4.2 Nonlinear system identification

Consider a nonlinear system given by [29]

$$y_p(k) = \frac{y_p(k-1)y_p(k-2)(y_p(k-1)+2.5)}{1+(y_p(k-1))^2+(y_p(k-2))^2} + u(k-1). \tag{14}$$

The equilibrium state of the system is (0, 0) and input is chosen as $u(k) \in \{-2, 2\}$, stable operation is guaranteed in the range of $\{-2, 2\}$. Uniformly distributed random variable in the range $\{-2, 2\}$ is chosen as training input and testing input is given by $u(k) = \sin(2\pi k/25)$. By selecting $[y_p(k-1), y_p(k-2), u(k-1)]$ as input variables and $y_p(k)$ as output variable, the system can write in form

$$\hat{y}_p(k) = \hat{f}(y_p(k-1), y_p(k-2), u(k-1)). \tag{15}$$

Total 800 data are selected, 600 data are used for training and 200 data for testing. Figure 2a shows the nonlinear system output which is randomly generated by $u(k) \in \{-2, 2\}$. Figure 2b shows the nonlinear system output with specified control effort $u(k) = \sin(2\pi k/25)$.

To the relation between the hidden neuron number L and testing error, we did the comparisons on nonlinear system identification for TT2-ELM. Figure 3 shows the actual output $y_p(k)$ and testing output $\hat{y}_p(k)$ using specified control effort $u(k) = \sin(2\pi k/25)$ with $L = 35$, and the testing error is 0.0580. Figure 4a shows the averaged 100 results of training error, and testing error for different hidden layer number L which varies from 25 to 75 is shown in Fig. 4b. Figure results show that the testing error will not always decrease when training error decreases and the hidden layer number L increases. Thus, to achieve a satisfiable precision, trail-and-error method should be used for the

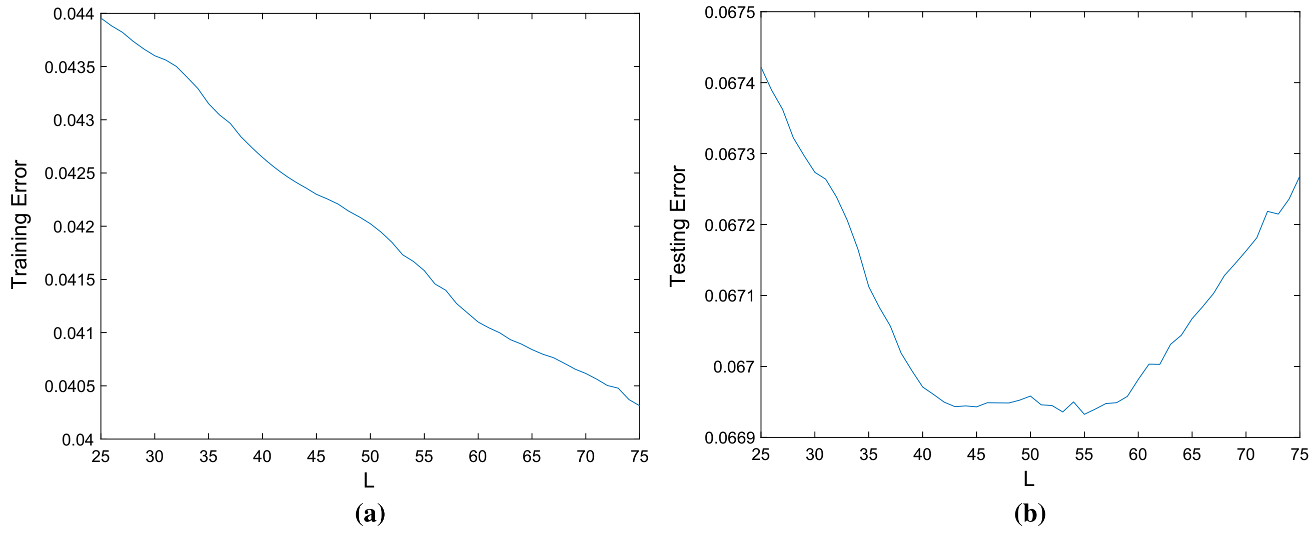

Fig. 4 The relation between the error and the hidden neuron number L

experiments based on different hidden neuron number when we used a priori type-2 fuzzy sets, although the weighting and bias parameters are randomly generated.

4.3 Real-world regression problems

In this section, four real-world regression problems are considered. Auto-Mpg is a data set which collects miles per gallon data from different automobile brand. The bank data set simulates the customers' levels of patience who choose their preferred bank depending on eight factors, such as residential areas, distances, fictitious temperature controlling bank choice, et al. Diabetes is to investigate the dependence of the level of serum C-peptide (to measure the patterns of residual insulin secretion) on the various factors. Triazines is a data set which is usually used to learn a regression equation or rules to predict the activity from the descriptive structural attributes. User-defined parameters of the four data sets are tabulated in Table 4, these data sets include three small-scale data sets and one moderate-scale data set. Table 5 shows the mean and standard deviation on auto-Mpg, bank, diabetes and triazines with 1000 experiment results. Five algorithms, ELM, RELM, WRELM, LS-SVM [30] and PLS [31], are adopted in the algorithm comparisons. TT2-ELM outperforms the other five algorithms on auto-Mpg and diabetes with respect to testing error, and outperforms the ELM, RELM, WRELM and LS-SVM on bank data set with respect to the training error. PLS achieves the best training error on three data sets, that is auto-Mpg, bank and diabetes, and achieves best testing error on bank. To the data set triazines, PLS fails to obtain the training error or testing error due to the occurrence of matrix singularity problem, reciprocal condition number is near 0. LS-SVM has the best training error on triazines, and WRELM performs relatively better than the other three variants of ELM, TT2-ELM has a comparable performance with ELM and RELM.

Table 4 User-defined parameters of the data sets

Data set	#Attributes	#Train set	#Test set
Auto-Mpg	7	300	92
Bank	9	4000	4192
Diabetes	2	576	192
Triazines	61	100	86

Table 5 Training and testing error on auto-Mpg, bank, diabetes and triazines

Data set	Algorithm	Training (RMSE)		Testing (RMSE)	
		Mean	SD	Mean	SD
Auto-Mpg	ELM	4.35e−01	2.41e+0	4.34e−01	2.55e+0
	RELM	1.35e+0	1.61e+0	1.31e+0	1.61e+0
	WRELM	1.34e+0	1.62e+0	1.35e+0	1.61e+0
	LS-SVM	1.83e+0	1.65e−02	3.89e+0	8.50e−02
	TT2-ELM	2.94e−01	7.12e−01	**2.93e−01**	6.84e−01
	PLS	**1.08e−06**	**5.93e−21**	1.50e+00	**1.00e−14**
Bank	ELM	3.24e−02	5.87e−02	3.21e−02	5.84e−02
	RELM	3.32e−02	5.83e−02	3.09e−02	5.86e−02
	WRELM	3.12e−02	5.83e−02	3.22e−02	5.86e−02
	LS-SVM	8.51e−02	2.80e−03	6.62e−02	1.05e−03
	TT2-ELM	3.10e−02	6.01e−02	3.50e−02	5.86e−02
	PLS	**1.62e−02**	**9.37e−17**	**3.48e−02**	**3.33e−16**
Diabetes	ELM	1.52e−01	1.55e−01	1.57e−01	1.37e−01
	RELM	1.50e−01	1.34e−01	1.57e−01	1.40e−01
	WRELM	1.50e−01	1.47e−01	1.57e−01	1.34e−01
	LS-SVM	1.07e+0	2.82e−02	2.22e+0	3.59e−01
	TT2-ELM	1.52e−01	1.31e−01	**1.57e−01**	1.35e−01
	PLS	**2.43e−02**	**3.33e−16**	1.62e−01	**2.30e−15**
Triazines	ELM	9.83e−02	2.41e−01	1.09e−01	1.66e−01
	RELM	9.94e−02	2.10e−01	1.07e−01	1.67e−01
	WRELM	1.05e−01	2.00e−01	**1.04e−01**	1.66e−01
	LS-SVM	**4.04e−02**	**6.55e−03**	1.50e−01	**1.39e−02**
	TT2-ELM	9.78e−02	2.00e−01	1.10e−01	1.59e−01
	PLS	–	–	–	–

4.4 Parameters sensitivity and stability analysis

In this section, parameters sensitivity and stability analysis are carried out for different iteration number. The parameter C is chosen from the parameter set $\{2^{-k} | k = 5, 10, 15, 20, 25, 30\}$ for RELM and WRELM. Hidden layer number L is varied from 10 to 80 and the best result is recorded. All the data sets use 100 iteration steps, except auto-Mpg uses 50. Figure 5a shows TT2-ELM achieves the smallest value among the *five algorithms*, the mean is 0.4529 and standard deviation is 0.0076. While RELM and WRELM stabilized around 1.7593. Figure 5b shows that the algorithms ELM, RELM and WRELM all vary along 0.0664, while TT2-ELM's curve presents the mean 0.0436 and standard deviation 9.01e−4. To the data set bank, PLS is the best algorithm, as is shown in Fig. 5b. Figure 5c shows the testing error on diabetes. Figure 5d shows the testing error on triazines. Figure 5c shows that numerical results of PLS flow flatly than other algorithms, as is shown in Table 5, while PLS meets the singularity problem on triazines. The test results show that PLS is not suitable for data sets when linear relation between input and output is not strong. TT2-ELM obtains the best performance on the diabetes, and worst on the triazines, this may partially due to RELM and WRELM utilized the regularization and weighted optimization method, TT2-ELM uses no further technique to the object function (see Table 1). Considering that TT2-ELM explores the input–output relation by tensor structure, the overall performance shows that TT2-ELM can perform better if more design parameters are considered in the mathematical model.

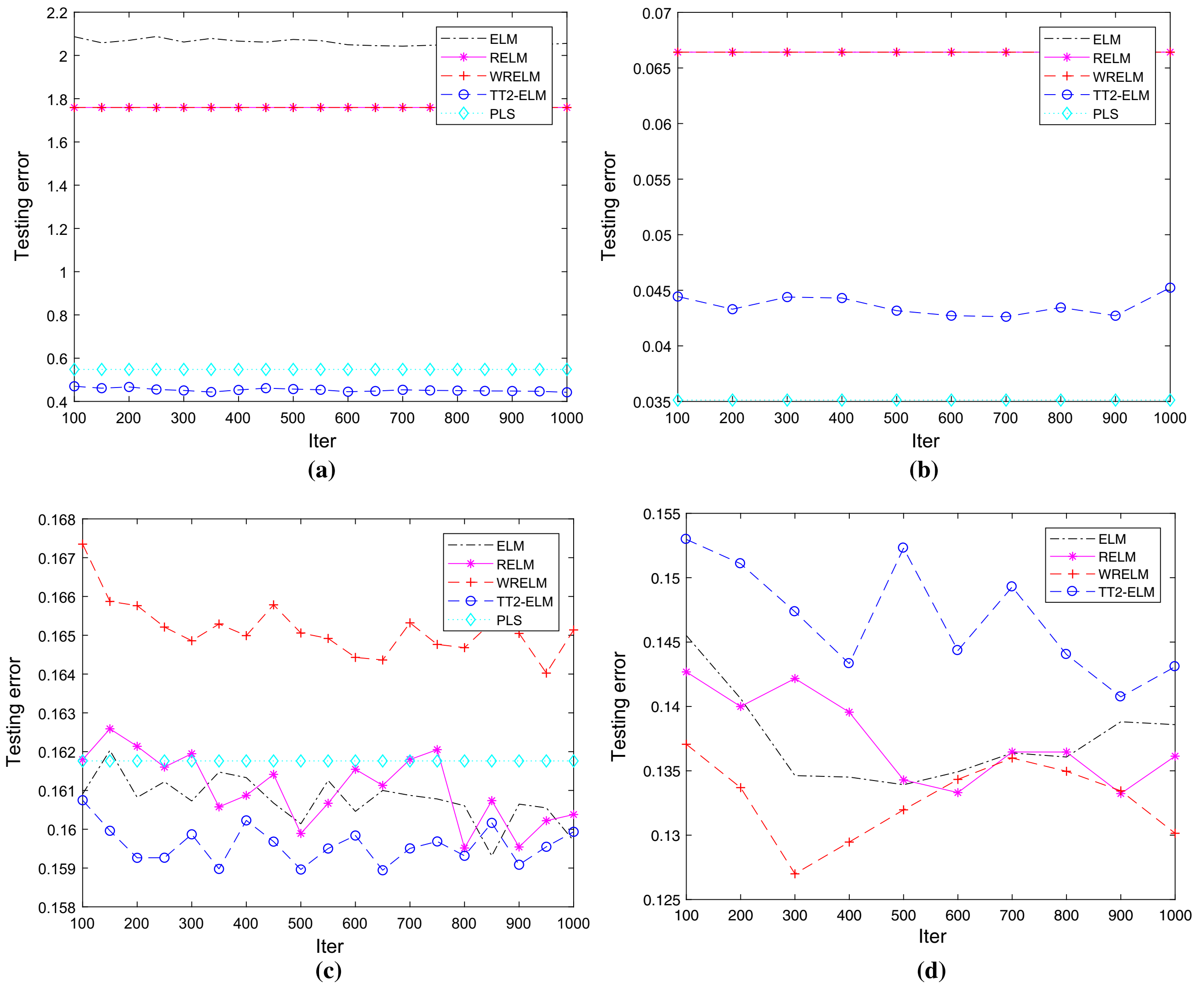

Fig. 5 Performance of ELM, RELM, WRELM, TT2-ELM and PLS by varying iteration number for regression data sets: **a** auto-Mpg, **b** bank, **c** diabetes and **d** triazines (PLS fails due to matrix singularity problem)

5 Conclusions and future work

Tensor regression is a new topic for fuzzy system modelling, it can be inferred as a high-dimensional case of the classical matrix regression problem. In this paper, we introduce the tensor regression to the type-2 fuzzy extreme learning machine scheme, the type-reduction step is avoided since the tensor structure can incorporate the information of the secondary membership function directly. Experimental results show that TT2-ELM outperforms ELM, RELM and WRELM on most of the data set. However, it should be noted that TT2-ELM has an inherit drawback which came from the tensor itself, the algorithm is not efficient for large-scale problem since it is time-consuming to solve a large-scale tensor regression problem. The batch training method which is proposed in this paper is space demanding for large-scale problem, an alternative way to avoid this problem is to design an incremental tensor decomposition algorithm to iteratively obtain the inverse of the tensor, it means that new theory on tensor operations is needed. Although we think that would be a great research, it is a different approach and might be more suitable for a different paper.

In this paper, a special tensor structure is constructed for the purpose of type-2 fuzzy set which is used in extreme learning machine scheme. There still exists much more work to be done. One direction is to design an extreme learning machine for general fuzzy sets, such as type-2 trapezoidal fuzzy sets, spiked type-2 fuzzy sets, et al. To the tensor regression, a sequential regression algorithm is more appreciated for online training or learning purpose. Generalized M–P inverse is essential for the tensor regression, the tensor operation should be redefined to endow the generalized M–P inverse more convenient to use for tensor regression.

Acknowledgements This work is supported by the National Natural Science Foundation of China (Nos. 11771111 and 61603126), and the Natural Science Foundation of Heilongjiang Province with Grant No. QC2016094.

Compliance with ethical standards

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

References

1. Huang GB, Ding XJ, Zhou HM (2010) Optimization method based extreme learning machine for classification. Neurocomputing 74(1):155–163
2. Huang GB, Zhou HM, Ding XJ, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern B Cybern 42(2):513–529
3. Zong WW, Huang GB, Chen YQ (2013) Weighted extreme learning machine for imbalance learning. Neurocomputing 101:229–242
4. Huang GB (2014) An insight into extreme learning machines: random neurons, random features and kernels. Cogn Comput 6(3):376–390
5. Deng W, Zheng Q, Chen L (2009) Regularized extreme learning machine. In: IEEE symposium on computational intelligence and data mining, pp 389–395
6. Javed K, Gouriveau R, Zerhouni N (2014) SW-ELM: a summation wavelet extreme learning machine algorithm with a priori parameter initialization. Neurocomputing 123:299–307
7. Miche Y, Sorjamaa A, Bas P, Simula O, Jutten C, Lendasse A (2010) OP-ELM: optimally pruned extreme learning machine. IEEE Trans Neural Netw 21(1):158–162
8. Efron B, Hastie T, Johnstone I, Tibshirani R (2004) Least angle regression. Ann Stat 32(2):407–451
9. Miche Y, van Heeswijk M, Bas P, Simula O, Lendasse A (2011) TROP-ELM: a double-regularized ELM using LARS and Tikhonov regularization. Neurocomputing 74(16):2413–2421
10. Deng WY, Ong YS, Zheng QH (2016) A fast reduced kernel extreme learning machine. Neural Netw 76:29–38
11. Deng WY, Bai Z, Huang GB, Zheng QH (2016) A fast SVD-hidden-nodes based extreme learning machine for large-scale data analytics. Neural Netw 77:14–28
12. Ding SF, Guo LL, Hou YL (2017) Extreme learning machine with kernel model based on deep learning. Neural Comput Appl 28(8):1975–1984
13. Kolda TG, Sun J (2008) Scalable tensor decompositions for multi-aspect data mining. In: 2008 eighth IEEE international conference on data mining. IEEE, Pisa, Italy, pp 363–372
14. Kolda T, Bader B (2009) Tensor secompositions and applications: a survey. SIAM Rev 51(3):455–500
15. Acar E, Dunlavy DM, Kolda TG, Rup MMO (2011) Scalable tensor factorizations for incomplete sata. Chemometr Intell Lab Syst 106(1):41–56
16. Acar E, Dunlavy DM, Kolda TG (2011) A scalable optimization approach for fitting canonical tensor decompositions. J Chemom 25(2):67–86
17. Papalexakis EE, Faloutsos C, Sidiropoulos ND (2012) ParCube: sparse parallelizable tensor decompositions. In: Cristianini N (eds) Proceedings of 2012 European conference on machine learning and knowledge discovery in databases. Springer, Berlin, Heidelberg, pp 521–536
18. Sun WW, Lu JW, Liu H, Cheng G (2017) Provable sparse tensor decomposition. J R Stat Soc Ser B (Stat Methodol) 79(3):899–916
19. Ji TY, Huang TZ, Zhao XL, Ma TH, Deng LJ (2017) A non-convex tensor rank approximation for tensor completion. Appl Math Model 48:410–422
20. Friedland S (2005) A new approach to generalized singular value decomposition. SIAM J Matrix Anal Appl 27(2):434–444
21. Zhou H, Li LX (2014) Regularized matrix regression. J R Stat Soc Ser B (Stat Methodol) 76(2):463–483
22. Friedland S, Mehrmann V, Pajarola R, Suter SK (2013) On best rank one approximation of tensors. Numer Linear Algebra Appl 20(6):942–955
23. De Lathauwer L, De Moor B, Vandewalle J (2000) A multilinear singular value decomposition. SIAM J Matrix Anal Appl 21(4):1253–1278
24. Sun LZ, Zheng BD, Bu CJ, Wei YM (2016) Moore–Penrose inverse of tensors via Einstein product. Linear Multilinear Algebra 64(4):686–698
25. Behera R, Mishra D (2017) Further results on generalized inverses of tensors via the Einstein product. Linear Multilinear Algebra 65(8):1662–1682

26. Ji J, Wei YM (2017) Weighted Moore–Penrose inverses and fundamental theorem of even-order tensors with Einstein product. Front Math China. https://doi.org/10.1007/s11464-017-0628-1
27. Mendel JM (2001) Uncertain rule-based fuzzy logic system: introduction and new directions. Prentice-Hall, Prentice
28. Brazell M, Li N, Navasca C, Tamon C (2013) Solving multilinear systems via tensor inversion. SIAM J Matrix Anal Appl 34(2):542–570
29. Rong HJ, Huang GB, Sundararajan N, Saratchandran P (2009) Online sequential fuzzy extreme learning machine for function approximation and classification problems. IEEE Trans Syst Man Cybern Part B (Cybern) 39(4):1067–1072
30. Suykens JAK, Vandewalle J (1999) Least squares support vector machine classifiers. Neural Process Lett 9(3):293–300
31. Li HD, Xu QS, Liang YZ (2014) libPLS: an integrated library for partial least squares regression and discriminant analysis. PeerJ PrePrints 2: e190v1, source codes available at www.libpls.net