Int. J. Mach. Learn. & Cyber. (2017) 8:587–595
DOI 10.1007/s13042-015-0351-8

ORIGINAL ARTICLE

Unsupervised extreme learning machine with representational features

Shifei Ding[1,2] · Nan Zhang[1,2] · Jian Zhang[1,2] · Xinzheng Xu[1,2] · Zhongzhi Shi[2]

Received: 21 November 2014 / Accepted: 13 March 2015 / Published online: 21 March 2015

Abstract Extreme learning machine (ELM) is not only an effective classifier but also a useful cluster. Unsupervised extreme learning machine (US-ELM) gives favorable performance compared to state-of-the-art clustering algorithms. Extreme learning machine as an auto encoder (ELM-AE) can obtain principal components which represent original samples. The proposed unsupervised extreme learning machine based on embedded features of ELM-AE (US-EF-ELM) algorithm applies ELM-AE to US-ELM. US-EF-ELM regards embedded features of ELM-AE as the outputs of US-ELM hidden layer, and uses US-ELM to obtain the embedded matrix of US-ELM. US-EF-ELM can handle the multi-cluster clustering. The learning capability and computational efficiency of US-EF-ELM are as same as US-ELM. By experiments on UCI data sets, we compared US-EF-ELM k-means algorithm with k-means algorithm, spectral clustering algorithm, and US-ELM k-means algorithm in accuracy and efficiency.

Keywords Data clustering · Extreme learning machine (ELM) · Extreme learning machine as an auto encoder (ELM-AE) · Unsupervised learning

✉ Shifei Ding
dingsf@cumt.edu.cn

[1] School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China

[2] Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China

1 Introduction

Extreme learning machine (ELM) proposed by Huang et al. [1, 2] is an efficient learning algorithm of training single layer feed-forward neural networks (SLFNs). Many researches regard ELM as a learning method for regression and multiclass classification [3–6]. Regularized ELM (RELM) has been developed for classification and regression [7]. Weighted ELM (WELM) has been used for the data with imbalanced class distribution [8]. Sparse ELM achieves similar generalization performance in binary classification applications [9]. Chen et al. [10] and Lin et al. [11] present the theoretical analysis of ELM. Recently, ELM has been applied to clustering, such as ELM k-means algorithm [12], and unsupervised extreme learning machine (US-ELM) k-means algorithm [13]. Clustering is an important part of machine learning. At present, many clustering methods have been proposed, such as partitioning methods [14], hierarchical methods [15, 16], density-based methods [17, 18], and graph theory methods [19, 20].

As we know, the original data will be linear separable through a nonlinear transformation. ELM k-means algorithm uses k-means clustering in ELM feature space. US-ELM obtains embedded matrix by constructing a new cost function, and then uses k-means to cluster in embedded matrix of US-ELM. However, the input weights and biases of US-ELM are randomly generated. Therefore, the outputs of US-ELM hidden layer are not reasonable features of the original data. Extreme learning machine as an auto encoder (ELM-AE) [21] can obtain main features of the original data. At the same time, ELM-AE is not an iterative algorithm. The proposed unsupervised extreme learning machine based on embedded features of ELM-AE (US-EF-ELM) can use the features that learned from ELM-AE

instead of the US-ELM, in combination with the US-ELM cost function to obtain output weights.

In order to assess the performance of the proposed algorithm, we compare the proposed algorithm with other algorithms on eight UCI datasets. US-EF-ELM k-means algorithm has achieved satisfying results on most of the data sets. This paper is organized as follows. In Sect. 2, we review the ELM method, and introduce the model of ELM-AE, then describe the principle of US-ELM. In Sect. 3, we make a detailed description of US-EF-ELM. In Sect. 4, experimental results on UCI data sets and the performance analysis of US-EF-ELM k-means algorithm are presented. Finally, some conclusions and the intending work are given in the last section.

2 Related works

The proposed US-EF-ELM is based on ELM-AE and US-ELM. This section provides brief reviews of ELM, ELM-AE, and US-ELM.

2.1 Extreme learning machine

ELM proposed by Huang et al. is an efficient learning algorithm of SLFNs. For N distinct samples (x_i, y_i), $i = 1,\ldots, N$, $x_i \in R^j$ and $y_i \in R^m$, the ELM model structure has j input layer nodes, n hidden layer nodes, m output layer nodes and a hidden layer activation function $g(x)$. The outputs of hidden layer can be expressed as Eq. 1, and the relationship between the outputs of hidden layer and the outputs of output layer can be expressed as Eq. 2.

$$h = g(ax + b) \tag{1}$$

$$h(x_i)\beta = y_i, \quad \text{where } i = 1, 2, \ldots, N \tag{2}$$

The above equation can be rewritten as Eq. 3:

$$H\beta = Y \tag{3}$$

where

$$\mathrm{H} = \begin{bmatrix} g(\vec{a}_1, b_1, \vec{x}_1) & g(\vec{a}_1, b_1, \vec{x}_2) & \cdots & g(\vec{a}_n, b_n, \vec{x}_N) \\ g(\vec{a}_2, b_2, \vec{x}_1) & g(\vec{a}_2, b_2, \vec{x}_2) & \cdots & g(\vec{a}_n, b_n, \vec{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ g(\vec{a}_n, b_n, \vec{x}_1) & g(\vec{a}_n, b_n, \vec{x}_2) & \cdots & g(\vec{a}_n, b_n, \vec{x}_N) \end{bmatrix}^T,$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \beta_2^T \\ \vdots \\ \beta_n^T \end{bmatrix}_{n\times m}, \quad Y = \begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_N^T \end{bmatrix}_{N\times m}.$$

Using ELM to obtain the output weights β can be divided into three steps:

Step 1: Randomly select numerical values to set input weights and the bias of the hidden layer

Step 2: Calculate the output matrix $\boldsymbol{H}$

Step 3: Calculate the output weights β:

$$\beta = H^{\dagger} Y \tag{4}$$

where $H^{\dagger}$ represents the generalized inverse matrix of the output matrix $\boldsymbol{H}$.

ELM has the advantage of high training speed and better generalization. However, the robustness of ELM is bad. To solve this question, Deng et al. [7] propose RELM which combines experiential risk and structural risk. RELM aims to obtain the output weights by minimizing the regularized least squares estimation cost function, which leads to the following formulation:

$$\min L_{RELM} = \frac{1}{2}\|\beta\|^2 + \frac{C}{2}\|Y - H\beta\|^2 \tag{5}$$

where C is a parameter to balance experiential risk and structural risk.

By setting the gradient of $\boldsymbol{L_{RELM}}$ to zero, we have

$$\beta + CH^T(Y - H\beta) = 0 \tag{6}$$

When the number of training samples is larger than the number of hidden layer nodes, the output weight matrix β in RELM can be expressed as Eq. 7:

$$\beta = \left(\frac{I}{C} + H^T H\right)^{-1} H^T Y \tag{7}$$

When the number of training samples is less than the number of hidden layer nodes, the output weight matrix β in RELM can be expressed as Eq. 8:

$$\beta = H^T \left(\frac{I}{C} + HH^T\right)^{-1} Y \tag{8}$$

2.2 Extreme learning machine as an auto encoder

The auto encoder is an unsupervised neural network model which is commonly used in deep learning. The outputs of auto encoder are equal to the inputs. ELM-AE proposed by Kasun et al. [21] is a novel method of neural network which can reproduce the input signal as well as auto encoder.

The ELM-AE model consists of an input layer, a single-hidden layer and an output layer. The model structure of ELM-AE is shown in Fig. 1, which has j input layer nodes, n hidden layer nodes, j output layer nodes and a hidden layer activation function $g(x)$. ELM-AE can be divided into three different representations.

$j > n$: compressed representation

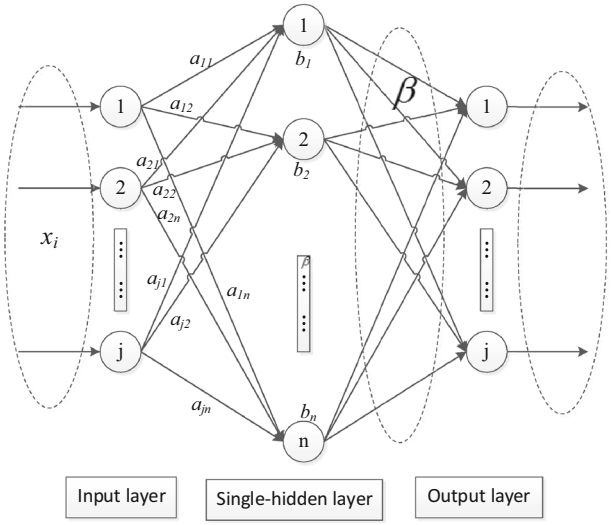

Fig. 1 The model structure of ELM-AE

Represent features from a higher dimensional input signal space to a lower dimensional feature space.

$j = n$: equal dimension representation

Represent features from an input signal space dimension equal to feature space dimension.

$j < n$: sparse representation

Represent features from a lower dimensional input signal space to a higher dimensional feature space.

There are two differences between ELM-AE and traditional ELM. Firstly, ELM is a supervised neural network and the outputs of ELM are labels, but ELM-AE is an unsupervised neural network and the outputs of ELM-AE are equal to the inputs of ELM-AE. Secondly, the input weights of ELM-AE are orthogonal and the biases of hidden layer of ELM-AE are also orthogonal, when the parameters in ELM are randomly initialized. For N distinct samples, the outputs of ELM-AE hidden layer can be expressed as Eq. 9, and the numerical relationship between the outputs of the hidden layer and the outputs of the output layer can be expressed as Eq. 10.

$$h = g(ax + b), \quad \text{where } \mathrm{a^T a = I, b^T b = 1} \tag{9}$$

$$h(x_i)\beta = x_i^T, \quad \text{where } i = 1, 2, \ldots, N \tag{10}$$

Using ELM-AE to obtain the output weights β also can be separated into three steps, but the calculation method of the output weights β in ELM-AE is different from ELM.

For sparse and compressed ELM-AE representations, output weights β are calculated by Eqs. 11 and 12:

When the number of training samples is more than the number of hidden layer nodes,

$$\beta = \left(\frac{I}{C} + H^T H\right)^{-1} H^T X \tag{11}$$

When the number of training samples is less than the number of hidden layer nodes,

$$\beta = H^T \left(\frac{I}{C} + HH^T\right)^{-1} X \tag{12}$$

For equal dimension ELM-AE representation, output weights β are calculated by Eq. 13:

$$\beta = H^{-1} X \tag{13}$$

2.3 Unsupervised extreme learning machine

US-ELM proposed by Huang et al. makes use of least square method to obtain the embedded matrix $E \in R^{N*m}$ (m is decided by us) which can be used to cluster. And the clustering validity of $\boldsymbol{E}$ is better than the original samples $X \in R^{N*j}$.

If two samples $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ are close to each other, then the outputs $\boldsymbol{y_i}$ and $\boldsymbol{y_j}$ of US-ELM should be close to each other as well. So US-ELM minimizes the following cost function,

$$\begin{aligned} L_m &= \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(y_i - y_j)^2}{2(c_1 - c_2)^2} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(y_i^2 - 2y_i y_j + y_j^2)}{2(c_1 - c_2)^2} \\ &= \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} -2w_{ij} y_i y_j + \sum_{i=1}^{n} y_i^2 \left(\sum_{j=1}^{n} w_{ij}\right)}{2(c_1 - c_2)^2} \\ &= \frac{2\hat{Y}^T(D - W)\hat{Y}}{2(c_1 - c_2)^2} = \frac{\hat{Y}^T L \hat{Y}}{(c_1 - c_2)^2} \end{aligned} \tag{14}$$

where $\boldsymbol{D}$ is a diagonal matrix, $\boldsymbol{L}$ is a Laplacian matrix, $\hat{Y} = [\vec{y}_i \quad \vec{y}_j]^T$.

US-ELM aims to obtain the output weights by minimizing the least squares regularization cost function, which leads to the following formulation:

$$\begin{aligned} &\min L_{US-ELM} = \|\beta\|^2 + C\,\mathrm{Tr}(\beta V^T H^T L H \beta) \\ &= Tr(\beta^T (I + CH^T L H)\beta) \\ &subject.to.\ (H\beta)^T H\beta = I \end{aligned} \tag{15}$$

where $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix.

If the number of the US-ELM output layer nodes for m, then choosing the output weights β whose columns are the eigenvectors corresponding to the first m smallest eigenvalues is an optimal solution to Eq. 15. However, there is always a constant eigenvector $\boldsymbol{1}$ in Laplace eigenvector space and this eigenvalue corresponding eigenvector is the smallest eigenvalue $\boldsymbol{0}$. Thus we choose eigenvectors

corresponding to first m smallest eigenvalues except for the 0 to compose the output weights β.

When the number of training samples is more than the number of hidden layer nodes,

$$(I + CH^T LH)v = \gamma H^T Hv \tag{16}$$

$$\beta = [\tilde{v}_2, \tilde{v}_3, \ldots, \tilde{v}_{n_0+1}], \quad where\ \tilde{v}_i = {}^{v_i}/_{\|Hv_i\|} \tag{17}$$

When the number of training samples is less than the number of hidden layer nodes,

$$(I + CLHH^T)\, u = \gamma HH^T u \tag{18}$$

$$\beta = H^T[\tilde{u}_2, \tilde{u}_3, \ldots, \tilde{u}_{n_0+1}], \quad where\ \tilde{u}_i = {}^{u_i}/_{\|HH^T u_i\|} \tag{19}$$

3 Unsupervised extreme learning machine based on embedded features of ELM-AE

US-ELM can obtain the output weights by constructing a different cost function instead of the ELM cost function without supervised information. However, the randomly generated input weights and biases will lead a result that the outputs of US-ELM hidden layer cannot represent features of the original samples very well. ELM-AE is an artificial neural network algorithm which is able to reconstruct the input signal. In order to achieve the reconstruction, ELM-AE obtains the main features of the original samples, as auto encoder does. ELM-AE can find principal components which represent the original samples and the learning process of ELM-AE without iteration. We can make use of features of ELM-AE instead of the outputs of US-ELM hidden layer, and then use the US-ELM cost function to obtain the output weights. We call it US-EF-ELM.

For N distinct samples (x_i, y_i), $i = 1,\ldots, N$, $x_i \in R^j$ and $y_i \in R^m$, the model structure of US-EF-ELM is shown in Fig. 2, with j input layer nodes, n hidden layer nodes, m output layer nodes and a hidden layer activation function $g(x)$. First of all, we make use of ELM-AE to calculate the output weights β_1. The transpose of β_1 is the input weights of US-EF-ELM. The outputs of US-EF-ELM hidden layer $\boldsymbol{H}$ can be expressed as

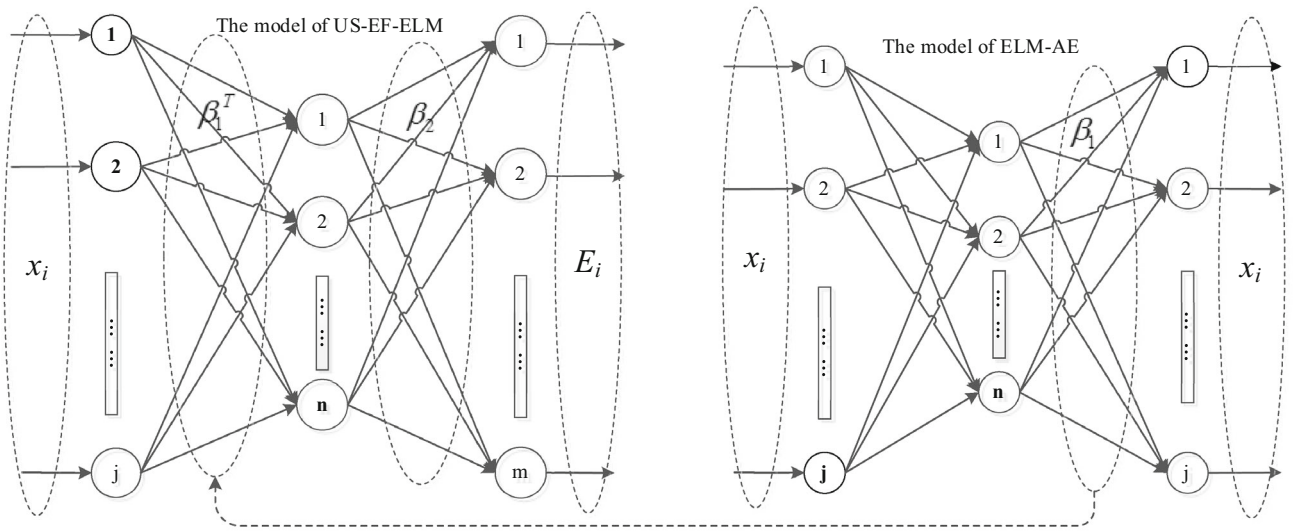

Fig. 2 The model structure of US-EF-ELM

Table 1 The details of UCI data sets

Data sets	Cluster	Dimension	N
Ionosphere	2	34	351
Breast Cancer Wisconsin (Diagnostic)	2	30	569
Musk (Version 1)	2	166	476
Musk (Version 2)	2	166	6598
Abalone	3	8	4177
ISOLET	26	617	7797
EEG Eye State	2	14	14,980
PEMS-SF	7	138,672	440

When $n \neq j$ and $N < n$,

$$\beta_1 = \left(\frac{I_n}{C_1} + H_{ELM-AE}^T H_{ELM-AE}\right)^{-1} H_{ELM-AE}^T X \quad (20)$$

When $n \neq j$ and $N < n$,

$$\beta_1 = H_{ELM-AE}^T \left(\frac{I_N}{C_1} + H_{ELM-AE} H_{ELM-AE}^T\right)^{-1} X \quad (21)$$

When $n = j$, $\beta_1 = H_{ELM-AE}^{-1} X$ (22)

$$H = g(X \cdot \beta_1^T) \quad (23)$$

US-EF-ELM aims to obtain the output weights by minimizing the US-ELM cost function, and leads to the following formulation:

$$\begin{aligned} \min_{\beta_2^T H^T H \beta_2 = I} L_{US-SL-ELM} &= \|\beta_2\|^2 + C_2 \mathrm{Tr}(\beta_2^T H^T L H \beta_2) \\ &= \mathrm{Tr}(\beta_2^T (I + C_2 H^T L H) \beta_2) \end{aligned} \quad (24)$$

So we choose the eigenvectors corresponding to the first m smallest eigenvalues except for the 0 to compose output weights β_2.

$$\begin{aligned} &\text{When } N > n, \ (I_n + C_2 H^T L H) v = \gamma H^T H v, \\ &\beta_2 = [\tilde{v}_2, \tilde{v}_3, \ldots, \tilde{v}_{n_0+1}], \quad \text{where } \tilde{v}_i = v_i / \|H v_i\| \end{aligned} \quad (25)$$

$$\begin{aligned} &\text{When } N < n, \ (I_N + C_2 L H H^T) u = \gamma H H^T u, \\ &\beta_2 = H^T [\tilde{u}_2, \tilde{u}_3, \ldots, \tilde{u}_{n_0+1}], \quad \text{where } \tilde{u}_i = u_i / \|H H^T u_i\| \end{aligned} \quad (26)$$

The outputs $\boldsymbol{E}$ of US-EF-ELM can be calculated by following formulation:

$$E = H\beta_2 \quad (27)$$

Then we can adopt the k-means algorithm to perform clustering in the embedded spaces of US-EF-ELM $\boldsymbol{E}$. The following algorithm is a summary of the proposed US-EF-ELM.

Algorithm. US-EF-ELM algorithm.

Inputs:

The samples $\boldsymbol{X} \in \boldsymbol{R}_N \times \boldsymbol{R}_j$, the parameters $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$.

Outputs:

The outputs $\boldsymbol{E} \in \boldsymbol{R}_N \times \boldsymbol{R}_m$.

Method:

Step 1: Initiate an ELM-AE network of n hidden neurons, and calculate the outputs of the hidden layer $\boldsymbol{H}_{ELM\text{-}AE} \in \boldsymbol{R}_N \times \boldsymbol{R}_n$.

Step 2: Make use of Equation 21-22 to calculate the output weights of ELM-AE β_1.

Step 3: Construct the graph Laplacian matrix $\boldsymbol{L}$ from $\boldsymbol{X}$.

Step 4: Initiate an US-EF-ELM network of n hidden neurons. The input weight matrix is the transpose of β_1. Then calculate the outputs of the hidden layer $\boldsymbol{H} \in \boldsymbol{R}_N \times \boldsymbol{R}_n$.

Step 5: Make use of Equation 25-26 to calculate the output weights of US-EF-ELM β_2.

Step 6: **Return** the embedded spaces of US-EF-ELM $\boldsymbol{E} \in \boldsymbol{R}_N \times \boldsymbol{R}_m$: $E = H\beta_2$.

4 Experiments and results

In this section, the performance of the US-EF-ELM k-means clustering algorithm is compared with classical methods (k-means algorithm and spectral clustering algorithm) and the US-ELM k-means clustering algorithm.

4.1 Data sets

The data sets used in the experiments are all from the UCI Machine Learning Repository, which include Ionosphere, Breast Cancer Wisconsin (Diagnostic), Musk (Version 1), Musk (Version 2), Abalone, ISOLET, EEG Eye State, and

Table 2 The performance comparison of the proposed US-EF-ELM

Data sets	Accuracy	k-means	SC	US-ELM k-means	US-EF-ELM k-means
Ionosphere	Max	71.23	66.38	83.48	85.76
	Mean	70.93 ± 1.99	64.42 ± 0. 22	66.51 ± 12.64	69.05 ± 10.58
Breast Cancer Wisconsin (Diagnostic)	Max	92.79	93.15	92.9701	93.50
	Mean	92.79 ± 0	93.15 ± 0	83.41 ± 15.10	93.38 ± 0.08
Musk (Version 1)	Max	56.30	56.09	67.86	70.38
	Mean	54.07 ± 0.76	56.09 ± 0	54.55 ± 3.41	54.61 ± 3.48
Musk (Version 2)	Max	74.48	54.15	85.77	86.078
	Mean	54.92 ± 4.04	54.15 ± 0	81.21 ± 7.27	81.698 ± 6.34
Abalone	Max	52.74	53.27	53.91	58.20
	Mean	52.54 ± 0.19	52.41 ± 2.58	53.61 ± 0.95	51.45 ± 2.95
ISOLET	Max	60.18	38.99	64.7557	66.12
	Mean	53.10 ± 3.45	34.91 ± 2.81	57.03 ± 3.38	57.79 ± 3.40
EEG Eye State	Max	55.11	55.11	59.80	58.68
	Mean	55.11 ± 0	55.11 ± 0	54.14 ± 2.20	58.64 ± 0.01
PEMS-SF	Max	35.68	24.09	37.73	49.77
	Mean	31.46 ± 2.05	22.12 ± 1.64	32.24 ± 2.30	37.48 ± 3.34

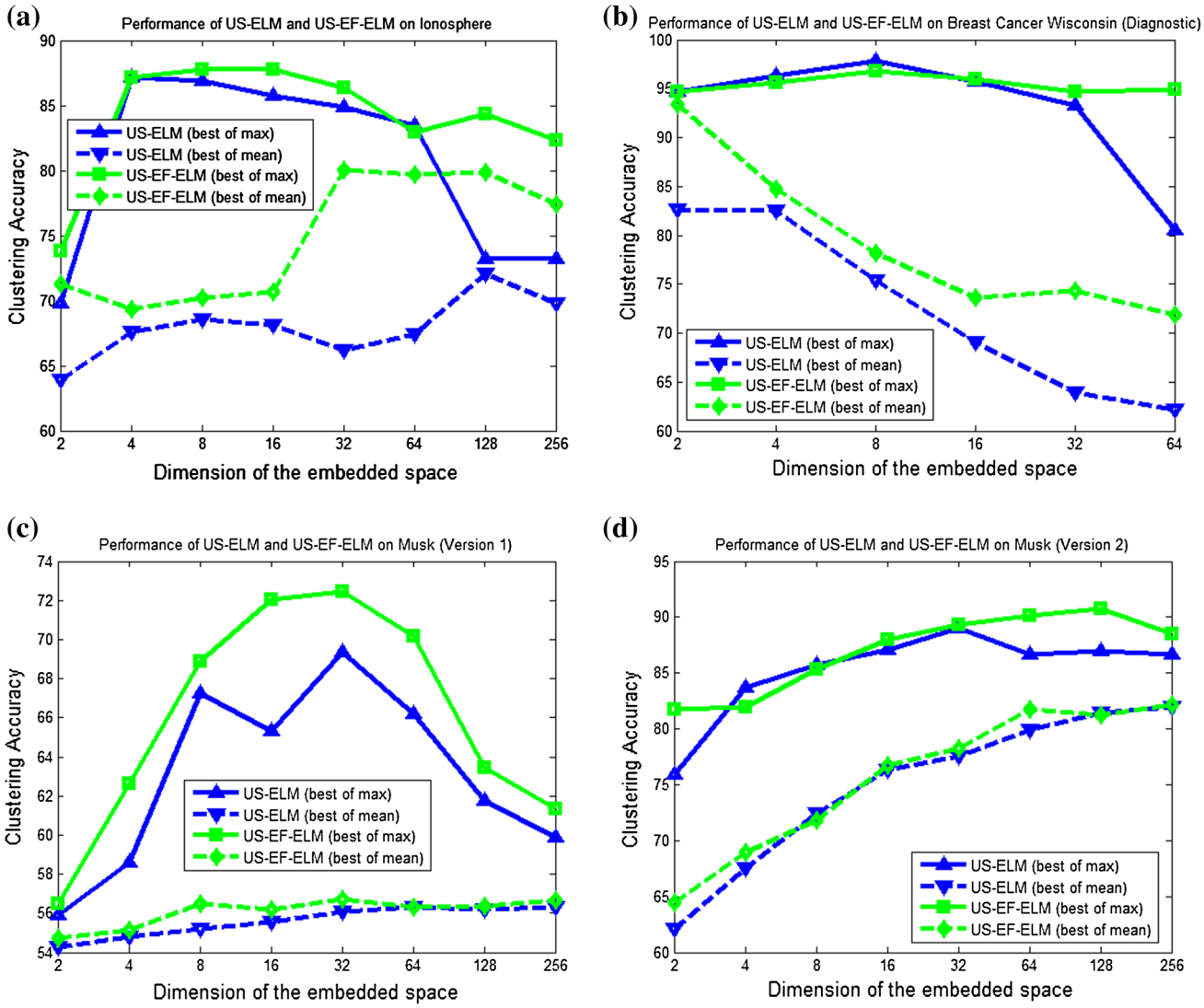

Fig. 3 Clustering performance on UCI data sets as a function of the dimension of the embedded space. **a** Clustering performance on Ionosphere. **b** Clustering performance on Breast Cancer Wisconsin (Diagnostic). **c** Clustering performance on Musk (Version 1). **d** Clustering performance on Musk (Version 2). **e** Clustering performance on Abalone. **f** Clustering performance on ISOLET. **g** Clustering performance on EEG Eye State. **h** Clustering performance on PEMS-SF

PEMS-SF. The details of those data sets are given in Table 1.

4.2 Experimental setup

We do experiments in a work station with a core i7 DMI2-Intel 3.6 GHz processor and 18 GB RAM running MATLAB 2012B. We run k-means algorithm, spectral clustering (SC) algorithm, US-ELM k-means algorithm, and US-EF-ELM k-means algorithm 100 times, and then run SC algorithm 10 times on last five datasets.

The number of US-ELM hidden nodes and US-EF-ELM hidden nodes is set to 2000 for all data sets, and activation functions of US-ELM and US-EF-ELM are sigmoid function. The kernel of SC algorithm is the Gaussian kernel. The parameter $\boldsymbol{C}$ in US-ELM and the parameters $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$ in US-ELM are chosen from [10^{-4} 10^{-3} 10^{-2} 10^{-1} 10^{0} 10^{1} 10^{2} 10^{3} 10^{4}].

4.3 Quality of the clustering results

This paper uses clustering accuracy (ACC) [22] to measure the quality of the clustering results. For N distinct samples $x_i \in R^j$, y_i and c_i are the inherent category label and the predicted cluster label of $\boldsymbol{x_i}$, the calculation formula of ACC is

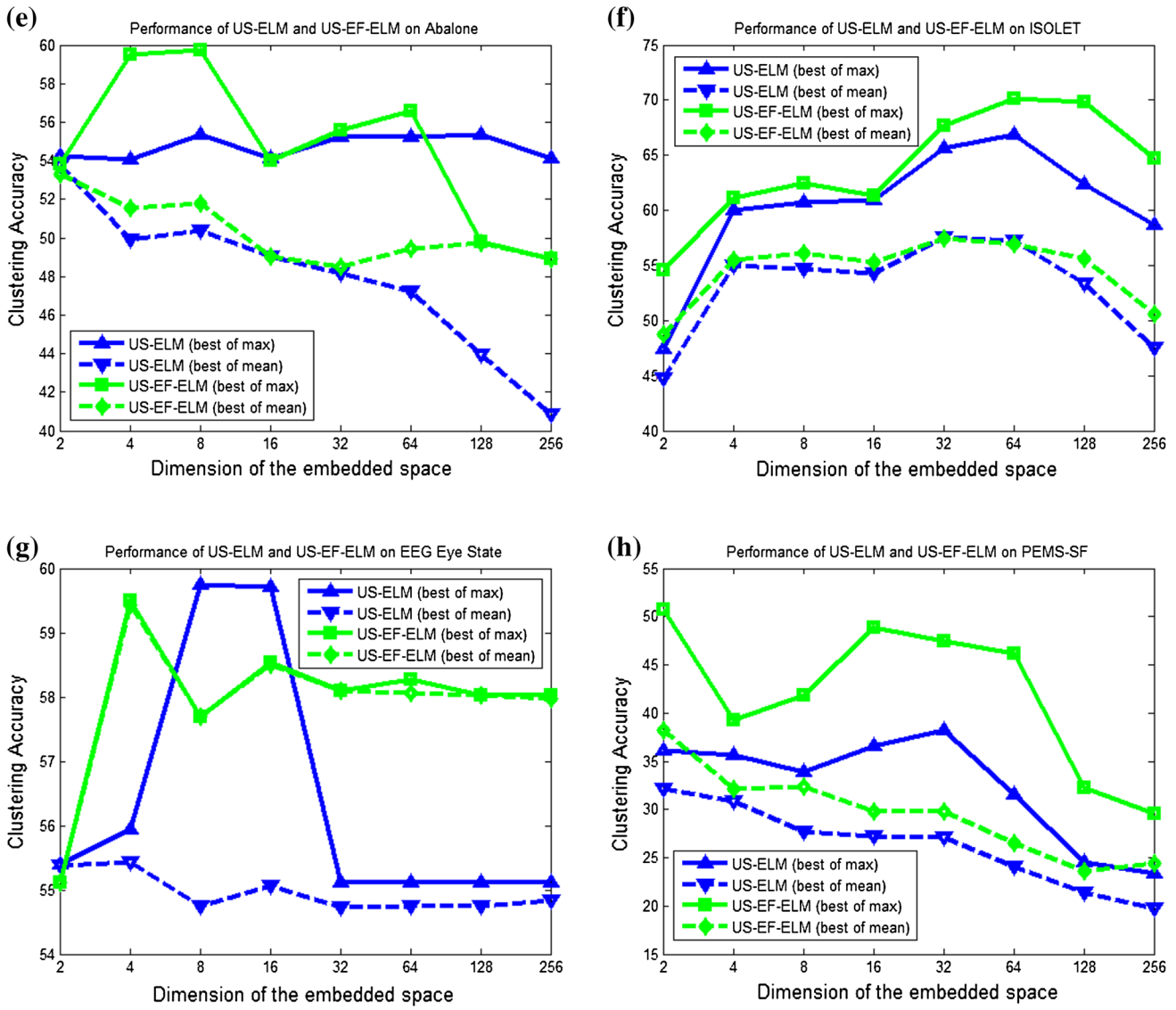

Fig. 3 continued

$$ACC = \sum_{i=1}^{N} \delta(y_i, map(c_i)) \Big/ N \tag{28}$$

where $map(\cdot)$ maps each cluster label to a category label by the Hungarian algorithm [23] and this mapping is optimal, let $\delta(y_i, c_i)$ equal to 1 if $y_i = c_i$ or equals to 0 otherwise. The higher values of the ACC, the better the clustering performance.

The comparison of these algorithms is shown in Table 2, and the dimensions of the embedded space of US-ELM and US-EF-ELM are selected based on Fig. 3. It is obvious that the US-EF-ELM k-means algorithm has achieved gratifying results on most of the data sets. The first three data sets are small, next five data sets are larger. SC algorithm performs better on small data sets. US-ELM k-means algorithm and US-EF-ELM k-means algorithm perform better on larger data sets.

The embedded space of US-ELM and US-EF-ELM makes an obvious effect on clustering performance. Figure 3 shows the best clustering performance of US-ELM and US-EF-ELM in different dimensions of the embedded space. We can get the following conclusion from Fig. 3: (1) the best cluster performance of US-EF-ELM is better than US-ELM; (2) the US-EF-ELM is better than US-ELM in a wide range of embedding dimensions; (3) Huang et al. [13] shows that the US-ELM attains its best performance with a very low dimensional embedding, but the US-ELM and US-EF-ELM attain their best performance with a high dimensional embedding on ISOLET and Musk (Version 2) data sets. Performing k-means in a

Table 3 The training time comparison of the proposed US-EF-ELM

Data sets	k-means (s)	SC (s)	US-ELM k-means (s)	US-EF-ELM k-means (s)
Ionosphere	0.0084	0.5413	0.2689	1.3288
Breast Cancer Wisconsin (Diagnostic)	0.0105	1.2418	0.5699	2.4575
Musk (Version 1)	0.0115	1.1357	0.9218	2.0904
Musk (Version 2)	0.0111	979.6176	3.9395	2.4397
Abalone	0.0165	231.3975	8.4078	12.1316
ISOLET	1.8812	2135.4	28.0250	38.9578
EEG Eye State	0.0126	11445	50.9086	64.3058
PEMS-SF	1.8939	1300.8	27.6309	362.7193

relatively lower dimensional space is just for our reference.

4.4 Computational efficiency

The training time comparison of these algorithms is shown in Table 3. It is obvious that the k-means algorithm has the best computational efficiency. In terms of the process of US-EF-ELM k-means algorithm, the training time of US-EF-ELM is longer than US-ELM. The embedded space dimension of US-EF-ELM is different from US-ELM, so the training time of US-ELM k-means algorithm is longer on some datasets. From experimental results on PEMS-SF dataset, we can conclude that the computational efficiency of US-EF-ELM is significantly lower than US-ELM when samples have high dimension. The process of ELM-AE has taken a long time when samples have high dimension. The training time of SC algorithm is longest on large datasets. Though the training time of US-EF-ELM k-means algorithm is longer than US-ELM on most of the datasets, US-EF-ELM is still efficient.

5 Conclusions

In this paper, US-EF-ELM is presented, and US-EF-ELM k-means algorithm gets better clustering performances compared to US-ELM k-means algorithm on UCI datasets. Besides the good performance of US-EF-ELM k-means algorithm, the proposed algorithm is also computational efficient. US-EF-ELM is a learning algorithm of SLFNs, and can be a learning algorithm of multi layer feed-forward neural networks. The new algorithm makes use of ELM-AE to train the weights in each layer and then uses the US-ELM cost function to obtain the output weights. However, it is difficult to determine the model structure. Future research is to find a suitable model structure of the algorithm.

6 Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 61379101), the National Key Basic Research Program of China (No. 2013CB329502), and the Natural Science Foundation of Jiangsu Province (No. BK20130209).

References

1. Huang GB, Zhu QY, Siew CK (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of the international joint conference on neural networks (IJCNN'2004). IEEE Press, Budapest, pp 25–29
2. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1):489–501
3. Huang GB, Wang DH, Lan Y (2011) Extreme learning machines: a survey. Int J Mach Learn Cybern 2(2):107–122
4. Cambria E, Huang GB, Kasun LLC, Zhou HM, Vong CM, Lin J, Yin JP, Cai ZP, Liu Q, Li K, Leung VCM, Feng L, Ong YS, Lim MH, Akusok A, Lendasse A, Corona F, Nian R, Miche Y, Gastaldo P, Zunino R, Decherchi S, Yang XF, Mao KZ, Oh BS, Jeon J, Toh KA, Beng A, Teoh J, Kim J, Yu HC, Chen YQ, Liu JF (2013) Extreme learning machines [trends & controversies]. IEEE Intell Syst 28(6):30–59
5. Huang GB (2014) An Insight into extreme learning machines: random neurons, random features and kernels. Cogn Comput 6(3):376–390
6. Ding SF, Xu XZ, Nie R (2014) Extreme learning machine and its applications. Neural Comput Appl 25(3):549–556
7. Deng WY, Zheng QH, Chen L, Xu XB (2010) Research on extreme learning of neural networks. Chin J Comput 33(2):279–287
8. Zong WW, Huang GB, Chen YQ (2013) Weighted extreme learning machine for imbalance learning. Neurocomputing 101:229–242
9. Bai Z, Huang GB, Wang D, Westover MB (2014) Sparse extreme learning machine for classification. IEEE Trans Cybern. doi:10.1109/TCYB.2014.2298235
10. Chen H, Peng JT, Zhou YC, Li LQ, Pan ZB (2014) Extreme learning machine for ranking: generalization analysis and applications. Neural Networks 53:119–126
11. Lin SB, Liu X, Fang J, Xu ZB (2015) Is extreme learning machine feasible? A theoretical assessment (part II). IEEE Trans Neural Networks Learn Syst 26(1):21–34

12. He Q, Jin X, Du CY, Zhuang FZ, Shi ZZ (2014) Clustering in extreme learning machine feature space. Neurocomputing 128:88–95
13. Huang G, Song SJ, Gupta JND, Wu C (2014) Semi-supervised and unsupervised extreme learning machines. IEEE Trans Cybern 44(12):2405–2417
14. Lloyd S (1982) Least squares quantization in PCM. IEEE Trans Inf Theory 28(2):129–137
15. Defays D (1977) An efficient algorithm for a complete link method. Comput J 20(4):364–366
16. Zhang L, Zhang XH, Yang JW, Wang XQ, Li FZ (2012) Cascaded cluster ensembles. Int J Mach Learn Cybern 3(4):335–343
17. Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of 2nd international conference on KDD. AAAI Press, Menlo Park, pp 226–231
18. Xu X (2013) Enhancing gene expression clustering analysis using tangent transformation. Int J Mach Learn Cybern 4(1):75–83
19. Von Luxburg U (2007) A tutorial on spectral clustering. Stat Comput 17(4):395–416
20. Liu N, Chen F, Lu MY (2013) Spectral co-clustering documents and words using fuzzy K-harmonic means. Int J Mach Learn Cybern 4(1):31–40
21. Kasun LLC, Zhou HM, Huang GB, Vong CM (2013) Representational learning with extreme learning machine for big data. IEEE Intell Syst 28(6):31–34
22. Ding SF, Jia HJ, Shi ZZ (2014) Spectral clustering algorithm based on adaptive Nyström sampling for big data analysis. J Softw 25(9):2037–2049
23. Papadimitriou CH, Steiglitz K (1998) Combinatorial optimization: algorithms and complexity. Courier Dover Publications, Mineola