Memetic Comp. (2016) 8:85–95
DOI 10.1007/s12293-016-0188-z

REGULAR RESEARCH PAPER

Extreme learning machine via free sparse transfer representation optimization

Xiaodong Li[1] · **Weijie Mao**[2] · **Wei Jiang**[2] · **Ye Yao**[1]

Received: 10 November 2015 / Accepted: 11 April 2016 / Published online: 30 April 2016

Abstract In this paper, we propose a general framework for Extreme Learning Machine via free sparse transfer representation, which is referred to as transfer free sparse representation based on extreme learning machine (TFSR-ELM). This framework is suitable for different assumptions related to the divergence measures of the data distributions, such as a maximum mean discrepancy and K-L divergence. We propose an effective sparse regularization for the proposed free transfer representation learning framework, which can decrease the time and space cost. Different solutions to the problems based on the different distribution distance estimation criteria and convergence analysis are given. Comprehensive experiments show that TFSR-based algorithms outperform the existing transfer learning methods and are robust to different sizes of training data.

✉ Xiaodong Li
hzxiaodong22@163.com

Weijie Mao
wjmao@iipc.zju.edu.cn

Wei Jiang
jiangwei@iipc.zju.edu.cn

[1] School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, People's Republic of China

[2] State Key Laboratory of Industrial Control Technology Institute of Cyber-Systems and Control Zhejiang University, Yuquan Campus, Hangzhou 310027, People's Republic of China

Keywords Extreme learning machine · Transfer learning (TL) · Free sparse representation

1 Introduction

Machine learning and big data mining have recently been more attentions by researchers from different research fields [1–3]. Support vector machine (SVM) is based on the statistical learning and structural risk minimization principle [4,5]. However, it is known that both BP [6] neural network, SVM, and LS-SVM have some challenging issues such as slow learning speed, trivial human intervention and poor computational scalability [5,7]. A new learning algorithm, i.e., extreme learning machine (ELM) was proposed by Huang et al. [8]. Compared with BP neural networks, SVMs, and LS-SVMs, the ELM have better generalization performance at a much faster learning speed and with least human intervention.

Although ELM has made some achievements, there is still room for improvement. Some scholars are engaged in optimizing the learning algorithm of ELM [42–44]. Han et al. [9] encoded a priori information to improve the function approximation of ELM. Kim et al. [10] introduced a variable projection method to reduce the dimension of the parameter space. Zhu et al. [11] used a differential evolutionary algorithm to select the input weights for the ELM. Some other scholars dedicated themselves to optimize the structure of ELM. Wang et al. [12] properly selected the input weights and bias of ELM in order to improve the performance of ELM. Li et al. [13] proposed a structure-adjustable online ELM learning method, which can adjust the number of hidden layer RBF nodes. Huang et al. [14,15] proposed an incremental structure ELM, which can increase the number of hidden nodes gradually. Meanwhile, another incremental approach referred to as error minimized extreme learning machine (EM-ELM) was proposed by Feng et al. [16]. All these incremental ELM's start from a small sized ELM hidden layer, and random hidden node (nodes) are added to the hidden layer. During the growth of networks, the output weights are updated incrementally. On the other hand, an alternative to optimize the structure of ELM is to train a network structure that is larger than necessary and then prune the unnecessarily nodes during the learning. A pruned ELM (PELM) was proposed by Rong et al. [17,18] as a classification problem. Yoan et al. [19] proposed an optimally pruned extreme learning machine (OP-ELM) methodology. Besides, there are still other attempts to optimize the structure of ELM such as CS-ELM [20] proposed by Lan et al., which used a subset model selection method. Zong et al. [21] presented weighted extreme learning machine for imbalance learning. Fu [22] employed kernel ELM in the field research for the detection of impact location.

In many supervised machine learning algorithms, it is usually required to assume that the training and test data follow the same distribution. However, this assumption does not hold true in many real applications and challenges the traditional learning theories. To deal with such situations, transfer learning, as a new machine learning algorithm, has attracted a lot of attention because it can be a robust classifier with little or even no labeled data from the target domain by using the mass of labeled data from other existing domains (a.k.a., source domains) [23,24]. Pan et al. [25] proposed a Q learning system for continuous spaces. This is constructed as a regression problem for an ELM. Zhang et al. [26] proposed Domain Adaptation Extreme Learning Machine (DAELM), which learns from a robust classifier in E-nose systems, without loss of the computational efficiency and learning ability of traditional ELM. Huang et al. [27] extend ELMs for both semi-supervised and unsupervised tasks based on the manifold regularization. In the past years, researchers have made substantial contribution to ELM theories and applications in varying fields. Huang et al. [28,29] studied the general architecture of locally connected ELM and kernel ELM. Tang et al. [30] proposed the multilayer perceptron, extending an ELM-based hierarchical learning framework.

In this paper, this issues will be investigated. Therefore, an algorithm called free sparse transfer learning based on the ELM algorithm (TFSR-ELM) is proposed, which uses a small amount of target tag data and a large amount of source domain old data to build a high-quality classification model. The method takes the advantages of the traditional ELM and overcomes the issue that traditional ELM cannot freely transfer knowledge. In addition, a so-called TFSR-KELM based on the kernel extreme learning machine ELM is proposed as an extension to the TFSR-ELM method for pattern classification problems. Experimental results show the effectiveness of the proposed algorithm.

2 Brief review of the ELM and kernel ELM learning algorithms

In this section, a brief review of the ELM proposed in [31] is given. The essence of ELM is that in ELM the hidden layer need not be tuned. The output function of ELM for generalized SLFNs is

$$\begin{aligned} f_L(x) &= \sum_{i=1}^{L} \beta_i h_i(x_j) = \sum_{i=1}^{L} \beta_i h(w_i \cdot x_j + b_i) \\ &= \mathrm{h}(x)\beta \quad j = 1, \ldots, N \end{aligned} \tag{1}$$

where $w_i \in R^n$ is the weight vector connecting the input nodes and the ith hidden node,$b_i \in R$ is the bias of the ith hidden node, $\beta_i \in R$ is the weight connecting the ith hidden node and the output node, and $f_L(x) \in R$ is the output of the SLFN. $w_i \cdot x_j$ denotes the inner product of w_i and $x_j \cdot w_i$ and b_i are the learning parameters of hidden nodes and they are randomly chosen before learning.

If the SLFN with N hidden nodes can approximate the N samples with zero error, then there exist β_i, w_i, and b_i such that

$$\sum_{i=1}^{L} \beta_i h(w_i \cdot x_j + b_i) = t_j, \quad j = 1, \ldots, N \tag{2}$$

Equation (2) can be written compactly as

$$\mathbf{H\beta} = \mathbf{T}. \tag{3}$$

where

$$\mathbf{H} = \begin{pmatrix} \mathrm{h}(x_1) \\ \vdots \\ \mathrm{h}(x_N) \end{pmatrix} = \begin{pmatrix} h(w_1, b_1, x_1) & \cdots & h(w_L, b_L, x_1) \\ \vdots & \ddots & \vdots \\ h(w_1, b_N, x_1) & \cdots & h(w_L, b_L, x_N) \end{pmatrix}_{N \times L},$$

$$\mathbf{T} = [t_1, \ldots, t_N]^T, \quad \text{and} \quad \boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_L]^T.$$

Numerous efficient methods can be used to calculate the output weights β including but not limited to orthogonal projection methods, iterative methods [32] and singular value decomposition (SVD) [33].

According to the ridge regression theory [34], one can add a positive value to the diagonal of $\mathbf{HH}^T$; the resultant solution is more stable and tends to have better generalization performance:

$$f(x) = \mathbf{H\beta} = \mathrm{h}(x)H^T \left(\frac{\mathbf{I}}{C} + \mathbf{HH}^T \right)^{-1} \mathbf{T}, \tag{4}$$

The feature mapping $\mathrm{h}(x)$ is usually known to the user in ELM. However, if the feature mapping $\mathrm{h}(x)$ is unknown to users, a kernel matrix for ELM can be defined as follows [34]:

$$\Omega_{ELM} = \mathbf{HH}^T : \Omega_{ELM_{i,j}} = \mathrm{h}(x_i) \cdot \mathrm{h}(x_j) = K(x_i, x_j). \tag{5}$$

Thus, the output function of kernel ELM classifier can be written compactly as:

$$f(x) = \mathrm{h}(x)\mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{HH}^T\right)^{-1}$$
$$\mathbf{T} = \begin{bmatrix} K(x, x_1) \\ \vdots \\ K(x, x_N) \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \Omega_{ELM}\right)^{-1} \mathbf{T}. \tag{6}$$

```
Algorithm 1: Given a training set $\{(x_i,t_i)\}_{i=1}^N \subset R^n \times R^n$, activation
kernel function $g(\cdot)$, and the hidden node number L:
  Step 1: Randomly assign input weight $w_i$ and bias $b_i$, i = 1,⋯,L.
  Step 2: Calculate the hidden layer output matrix $\mathbf{H}$.
  Step 3: Calculate the output weight β: $\beta = \mathbf{H}^{\dagger}\mathrm{T}$.
```

$$\min_{B,S} \|X - UV\|_F^2 + Tr(VLV^T) + \lambda \sum_{i=1}^N \|v_i\|_1$$
$$s.t. \|u_k\|_2^2 \le 1, k = 1, \ldots, N \tag{8}$$

where $\lambda \ge 0$is the regularization parameter.

3.2 Unsupervised extreme learning machine

In unsupervised algorithm, the entire training data $X = \{x_i\}_{i=1}^N$ are not labeled (N is the number of training patterns) and the target is to find the underlying structure of the original data. When there is no labeled data, the formulation is

$$\min_{\beta \in R^{n_l \times n_o}} \frac{1}{2} \|\beta\|^2 + \lambda T_r \left(\beta^T H^T L H \beta\right). \tag{9}$$

Note that the above formulation always attains its minimum at $\beta = 0$. As suggested in [35], it is necessary to introduce additional constraints to avoid a degenerated solution. Specifically, the formulation of US-ELM is given by

$$\min_{\beta \in R^{n_l \times n_o}} \frac{1}{2} \|\beta\|^2 + \lambda T_r \left(\beta^T H^T L H \beta\right).$$
$$s.t. (H\beta)^T H\beta = I_{n_o} \tag{10}$$

An optimal solution to problem (10) is given by choosing β as the matrix whose columns are eigenvectors (normalized to satisfy the constraint) corresponding to the first n_o smallest eigenvalues of the generalized eigenvalue problem:

$$(I_{n_l} + \lambda H^T L H)\nu = \gamma H^T H \nu. \tag{11}$$

In the algorithm of Laplacian eigenmaps, the first eigenvector is discarded since it is always a constant vector proportional to 1 (corresponding to the smallest eigenvalue 0) [27,35]. In the US-ELM algorithm, the first eigenvector of (11) also leads to small variations in the embedding and is not useful for data representation. Therefore, it is suggested to abandon this trivial solution as well.

3 Proposed learning algorithm

In this section, the overall architecture of the proposed TFSR-ELM is introduced in detail, and a new kernel ELM free sparse representation is presented, this is utilized to determine the basic elements of TFSR-ELM.

3.1 Graph-laplacian regularization

Given a set of N-dimensional data points $X = \{x_i\}_{i=1}^N$, we can construct the nearest neighbor graph G with N vertices, where each vertex represents a data point. Let W be the weight matrix of G. If x_i is among the k-nearest neighbors of x_i, or vice versa, $W_i = 1$, otherwise, $W_i = 0$. We define the degree of x_i as $d_i = \sum_{j=1}^m W_{ij}$, and $D = diag(d_1, \ldots, d_m)$. Consider the problem of mapping the weighted graph G to sparse representations V,

$$\hat{L}_N = \frac{1}{2} \sum_{i,j} (v_i - v_j)^2 W_{i,j} = Tr(\hat{V}^T L \hat{V}), \tag{7}$$

where $Tr(\cdot)$ denotes the trace of a matrix and $L = D - W$ is the Laplacian matrix. By incorporating the Laplacian regularizer (7) into the original sparse representation, we can obtain the following objective function of GraphSC [36]:

Let $\gamma_1, \gamma_2, \ldots, \gamma_{n_o+1} (\gamma_1 \leq \gamma_2 \leq \cdots \leq \gamma_{n_o+1})$ be the (n_o+1) smallest eigenvalues of (11) and $\nu_1, \nu_2, \ldots, \nu_{n_o+1}$ be their corresponding eigenvectors. Then, the solution to the output weights β is given by

$$\beta^* = \left[\nu_2', \nu_3', \ldots, \nu_{n_o+1}'\right], \tag{12}$$

where $\nu_i' = \nu_i / \|\mathbf{H}\nu_i\|, i = 2, \ldots, n_o+1$ are the normalized eigenvectors. If the number of labeled data is less than the number of hidden neurons, problem (12) is underdetermined. In this case, we have the following alternative formulation by using the same trick as in previous sections:

$$(I_u + \lambda L H H^T) u = \gamma H H^T u. \tag{13}$$

Again, let $u_1, u_2, \ldots, u_{n_o+1}$ be generalized eigenvectors corresponding to the (n_o+1) smallest eigenvalues (13), then the final solution is given by

$$\beta^* = H^T \left[\tilde{u}_2, \tilde{u}_3, \ldots, \tilde{u}_{n_o+1}\right], \tag{14}$$

where $\tilde{u}_i = u_i / \left\|\mathbf{H}\mathbf{H}^T u_i\right\|, i = 2, \ldots, n_o+1$ are the normalized eigenvectors.

Our task is classified, the US-ELM in Algorithm 2:

3.3 Framework of TFSR-ELM

To build an effective association between Y_s and Y_t, we propose to embed all labels into a latent Euclidean space using a graph-based representation. As a result, the relationship between labels can be represented by the distance between the corresponding prototypes of the labels in the latent space. Furthermore, we show that predictions made by each source classifier can also be mapped into the latent space, which makes the knowledge transfer from source classifiers possible. Finally, a regularization framework is applied to learning an effective classifier for the data classification task. In this manner, the transfer learning framework depends on optimizing the free sparse representation simultaneously by the problem designer, and for this reason it is called "transfer free sparse representation learning" (TFSR-ELM in short). The free sparse representation algorithm is adopted to perform classification in the embedded space. The framework of the proposed TFSR-ELM method is shown in Fig. 1.

The traditional TL-ELM builds a learning model using the training and test data with different distributions that deal with transfer knowledge. The representations learned by TFSR-ELM can be used for building a robust and accurate data classifier. For every class, a US-ELM classifier is introduced, the objective function of which is integrated into (9). US-ELM hyperplane normal vectors for every class are coupled as columns of matrix $W \in R^{D\times m}$. In addition, all the margins are grouped for the training objects with respect to all the classes into matrix $\Psi \in \mathbb{R}^{m\times N_l}$.

```
Algorithm 2: Given a training set $\mathbf{X} \in R^{N\times n_i}$, a $n_o$ dimensional
space $\mathbf{E} \in R^{N\times n_o}$, the label vector $y \in \mathrm{N}_+^{N\times 1}$

Step 1: Build the graph Laplacian $\mathbf{L}$ from $\mathbf{X}$.

Step 2: Compute the hidden layer output matrix $\mathbf{H}$.

Step3: if $n_h \leq N$, compute the generalized eigenvector
$v_2, v_3, \cdots, v_{n_o+1}$, according to equation (10), Let
$\beta = \left[v_2', v_3', \cdots, v_{n_o+1}'\right]$, where $v_i' = v_i / \|\mathbf{H}v_i\|$, $i = 2, \cdots, n_o + 1$.else
compute the generalized eigenvector $u_2, u_3, \cdots, u_{n_o+1}$, according to
equation (13), Let $\beta = \mathbf{H}^T \left[u_2', u_3', \cdots, u_{n_o+1}'\right]$, $\lim_{x\to\infty}$ where
$u_i' = u_i / \|\mathbf{H}\mathbf{H}^T u_i\|$, $i = 2, \cdots, n_o + 1$.

Step4: Compute the embedding matrix: $\mathbf{E} = \mathbf{H}\beta$.

Step5: For clustering, Treat each row of $\mathbf{E}$ as a point, and
cluster the $N$ points into $K$ clusters using the k-means al-
gorithm. Let $y$ be the label vector of cluster index for all
the points.

return $\mathbf{E}$ (for embedding task) or $y$ (for clustering task);
```

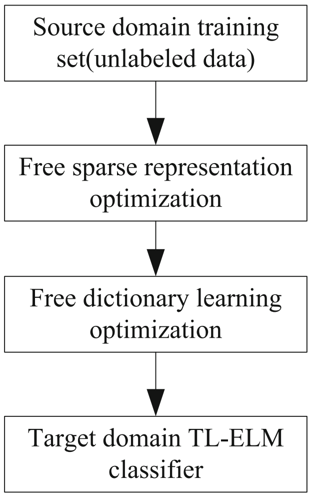

Source domain training set(unlabeled data)
Free sparse representation optimization
Free dictionary learning optimization
Target domain TL-ELM classifier

Fig. 1 The framework of proposed TFSR-ELM algorithm

$$\min_{U,V,W} \|X-UV\|_F^2+\lambda\sum_{i=1}^{N}\|v_i\|_1+Tr(V\tilde{M}V^T) + \lambda_1\left(\frac{1}{2}\|W\|_F^2+c1^T\Psi 1\right) \quad s.t. \quad \|u_k\|_2^2\le 1, k=1,\dots,K, \mathbf{1}-\Psi\preceq Y\circ(W^TV), \Psi\succeq 0, \tag{15}$$

where λ_1 is a tuning parameter, c is ELM coefficient, $\mathbf{1}$ is a matrix of ones, 1 is a vector of ones, $\circ$ denotes Hadamard product, and $\preceq$; $\succeq$ stand for element-wise inequalities.

A three-step algorithm is proposed for efficiently solving the TFSR-ELM optimization problem. The Lagrangian function for (15) is as follows

$$L(W,\Psi,U,V,\Gamma,\Theta,\nu)=\|X-UV\|_F^2+\lambda\sum_{i=1}^{N}\|v_i\|_1 + Tr(V\tilde{M}V^T)+\ \lambda_1\left(\frac{1}{2}\|W\|_F^2\right)+\sum_{k=1}^{K}(\|u_k\|_2^2-1) + \mathbf{1}^T((\lambda_1 c1^T+\ \Theta)\Psi+\Gamma\circ(\mathbf{1}-\Psi-Y\circ(W^TV)))\mathbf{1}, \tag{16}$$

where $\Gamma,\Theta\in\mathbb{R}^{m\times N_L}$, $\upsilon\in\mathbb{R}^K$ are the dual variables associated with corresponding inequality constraints. According to the duality theory, the following problem can be solved:

$$\max_{\Gamma,\Theta,\nu}\ \min_{W,\Psi,U,V}\ L(W,\Psi,U,V,\Gamma,\Theta,\nu) \quad s.t.\Gamma\succeq 0,\ \Theta\succeq 0,\ \nu\succeq 0. \tag{17}$$

The first order optimality conditions over W, Ψ will have the following form

$$W=(\Gamma\circ Y)V^T, 0=(\Gamma\circ Y)\mathbf{1}, \Gamma\preceq kc, \tag{18}$$

By substituting the values of (18) and (16) into (17), we obtain the following optimization problem

$$\max_{\Gamma,\nu}\min_{U,V} L_D(U,V,\Gamma,\nu) \quad s.t.(\Gamma\circ Y)\mathbf{1}=0, 0\preceq\Gamma\preceq kc,\ \nu\succeq 0, \tag{19}$$

where $L_D(U,V,\Gamma,\nu)$ has the following form

$$L_D(U,V,\Gamma,\nu)=\|X-UV\|_F^2+\lambda\sum_{i=1}^{N}\|v_i\|_1+1^T\Gamma 1 + \sum_{k=1}^{K}\nu_k(\|u_k\|_2^2-1) + Tr\left(V\left(\tilde{M}-\frac{1}{2}\mathrm{E}\right)V^T\right), \tag{20}$$

where $\mathrm{E}=(\Gamma\circ Y)^T(\Gamma\circ Y)$. Problem (20) can be efficiently solved via the following three steps of iterative algorithms.

Free Sparse Codes Learning is done by optimizing

$$\min_{V}\|X-UV\|_F^2+\lambda\sum_{i=1}^{N}\|v_i\|_1 + Tr\left(V\left(\tilde{M}-\frac{1}{2}E\right)V^T\right). \tag{21}$$

where the dictionary U is fixed. Because equation (21) with L_1-regularization is non-differentiable when u_k contains 0s, the standard unconstrained optimization algorithms cannot be applied. In the following, an optimization algorithm based on coordinate descent is introduced to solve this problem. It is easy to see that the problem (21) is convex, thus, the global minimum can be achieved. In order to solve the problem by optimization over each u_k, the problem (21) should be rewritten in a vector form.

The reconstruction error $\|X-UV\|_F^2$ can be rewritten as follows:

$$\sum_{k=1}^{N}\|x_k-Uv_k\|^2 \tag{22}$$

The Laplacian regularizer $Tr(VLV^T)(L=\tilde{M}-\frac{1}{2}E)$ is the Laplacian matrix can be rewritten as follows:

$$Tr(VLV^T)=Tr\left(\sum_{k,i=1}^{N}L_{ki}v_kv_i^T\right) = \sum_{k,i=1}^{K}L_{ki}v_i^Tv_k=\sum_{k,i=1}^{K}L_{ki}v_k^Tv_i \tag{23}$$

Combining (22) and (23), problem (21) can be rewritten as

$$\min_{V}\sum_{i=1}^{N}\|x_i-Uv_i\|^2+\lambda\sum_{i=1}^{N}\|v_i\|_1+\sum_{k,i=1}^{K}L_{ki}v_k^Tv_i. \tag{24}$$

When updating ν_i, the vectors $\{\nu_k\}_{k\neq j}$ are fixed. Thus, the following optimization problem is obtained:

$$\min_{v_i} f(v_i) = \|x_i - Uv_i\|^2 + \lambda \sum_{k=1}^{j} \left|v_k^{(j)}\right| + L_{ii} v_i^T v_i + v_i^T h_i, \tag{25}$$

where $h_i = 2(\sum_{k \neq i} L_{ik} s_k)$ and $s_k^{(j)}$ is the kth coefficient of s_k.

Following the feature-sign search algorithm proposed in [37], Eq. (25) can adopt a subgradient strategy to solve the non-differentiable problem, which uses subgradients of $f(v_i)$ at non-differentiable points.

Dictionary learning is performed by solving the problem:

$$\max_{v} \min_{U} \left(\|X - UV\|_F^2 + \sum_{k=1}^{K} v_k (\|u_k\|_2^2 - 1) \right)$$
$$s.t. v \succeq 0, \tag{26}$$

by the iterative optimization method in [36], while fixing the coefficient matrix V. Let $v = [v_1, \ldots, v_K]$,and v_k be the Lagrange multiplier. The problem becomes a least squares problem with quadratic constraints

$$\min_{U} f(U) = \|X - UV\|^2 + \sum_{k=1}^{K} v_k (\|u_k\|^2 - 1), \tag{27}$$

$$\begin{aligned} L(U, v) &= \|X - UV\|^2 + Tr(U^T U \Lambda) - Tr(\Lambda) \\ &= Tr(X^T X) - 2Tr(U^T X V^T) \\ &\quad + Tr(V^T U^T U V) + Tr(U^T U \Lambda) - Tr(\Lambda) \end{aligned} \tag{28}$$

The optimal solution U^* can be obtained by letting the first-order derivative of (28) to be equal to zero

$$U^* V V^T - X V^T + U^* \Lambda = 0. \tag{29}$$

Then, it resolves to

$$U^* = X V^T (V V^T + \Lambda)^{-1}. \tag{30}$$

Substituting (30) into (28), the Lagrange dual function is

$$\begin{aligned} L(v) &= Tr(X^T X) - 2Tr(X V^T (V V^T + \Lambda)^{-1} V X^T) \\ &\quad + Tr((V V^T + \Lambda)^{-1} V X^T X V^T) - Tr(\Lambda) \end{aligned} \tag{31}$$

This leads to the following Lagrange dual function:

$$\min_{\Lambda} Tr(X V^T (V V^T + \Lambda)^{-1} V X^T + Tr(\Lambda)$$
$$s.t. v \succeq 0. \tag{32}$$

This equation (32) can be solved by using Newton or conjugate gradient algorithm.

Learning Unsupervised ELM. Finally, the optimal classifier parameters are searched:

$$\min_{\Gamma} \left(\frac{1}{2} Tr(V \mathrm{E} V^T - 1^T \Gamma 1) + \sum_{k=1}^{K} v_k (\|u_k\|_2^2 - 1) \right)$$
$$s.t. (\Gamma \circ Y) 1 = 0, \quad 0 \preceq \Gamma \preceq kc, \tag{33}$$

```
Algorithm 3: Given a training set $\{(x_i, y_i)\}_{i=1}^N \subset R^n \times R^n$ ,
itr_no-number of iterations $\alpha$, $\mu$, $k$, $\lambda$, $c$ -parameters,
building the MMD matrix, Graph-Laplacian matrix L, and
one-hot encoding matrix Y of labels for the objects.
U-uniform random matrix with zero mean for each column.
$\Gamma \leftarrow 0$, $\mathrm{E} \leftarrow 0$, $t \leftarrow 1$.
Step 1: Calculate the hidden layer output matrix $\mathbf{H}$.
Step 2: Calculate the output weight $\beta$: $\beta = \mathbf{H}^{\dagger}$T.
Step 3: do
    {
        Find $V$ by solving Free Sparse Codes Learning sub-
        problem.
        Find $U$ by solving Dictionary Codes Learning subproblem.
        Find $\Gamma$ and calculate $\mathrm{E}$ by solving ELM Learning sub-
        problem.
    } while( $t \le iter\_no$ )
Step 4: Output $U$, $V$
```

3.4 Framework of TFSR-KELM

Kernel extreme learning machine is based on kernel learning. Specifically, we propose a transfer free sparse representation based on a kernel ELM algorithm which starting from a basic kernel, tries to learn chains of kernel transforms that can produce good kernel matrices for the source tasks. The same sequence of transformations can be then applied to compute the kernel matrix for new related target tasks. This method is applied to the unsupervised and transfer learning.

According to kernel ELM, it can be formulated as:

```
Algorithm 4: Given a training set $\{(x_i, y_i)\}_{i=1}^{N} \subset R^n \times R^n$ ,
itr_no-number of iterations $\alpha$, $\mu$, $k$, $\lambda$, $c$ -parameters,
building the MMD matrix, Graph-Laplacian matrix L, and
one-hot encoding matrix Y of labels for the objects.
U-uniform random matrix with zero mean for each column.
$\Gamma \leftarrow 0$, E $\leftarrow 0$, $t \leftarrow 1$.
Step 1: Calculate the hidden layer output matrix $\mathbf{H}$.
Step 2: Calculate the output weight $\beta$: $\beta = \mathbf{H}^{\dagger}$T.
Step 3:
    do
     {
      Find $V$ by solving Free Sparse Codes Learning sub-
      problem.
      Find $U$ by solving Dictionary Codes Learning subproblem.
      Find $\Gamma$ and calculate E by solving KELM Learning sub-
      problem.
     } while( $t \leq$ iter_no )
 Step 4: Output $U$, $V$
```

$$\max_{\Gamma,\nu}\min_{U,V} L_D(U,V,\Gamma,\nu)$$
$$s.t.(\gamma_{KMS}(p,q)\circ Y)\mathbf{1}=0, 0\preceq\gamma_{KMS}(p,q)\preceq kc, \nu\succeq 0, \tag{34}$$

The primal of Eq.(34) is defined as

$$\min_{\Gamma}\left(\frac{1}{2}Tr(V\mathrm{E}V^T-1^T\Gamma 1)+\sum_{k=1}^{K}v_k(\|u_k\|_2^2-1)\right)$$
$$s.t.(k_{\sigma/\gamma}(x_i,x_j)\circ Y)1=0, 0\preceq k_{\sigma/\gamma}(x_i,x_j)\preceq kc, \tag{35}$$

Standard Gaussian kernel (i.e.$k_{\sigma/\gamma}(x,y) = \exp(-\frac{1}{2(\sigma/\gamma)^2}\|x-y\|^2)$, bandwidth σ/γ) is used as the default kernel.

As for multiclass classification problems, the traditional methods, such as one against one (OAO) or one against all (OAA) classification algorithms, decompose a multiclass classification problem into several binary classification problems in. However, these methods suffer from the problem of high computational complexity. For some multiclass classification problems, the optimal solution of TFSR-KELM can be formulated as

$$\min_{\Gamma}\left(\frac{1}{2}Tr(V\mathrm{E}V^T-1^T\Gamma 1)+\sum_{k=1}^{K}v_k(\|u_k\|_2^2-1)\right)$$
$$s.t.(K_s\circ Y)1=0, \quad 0\preceq K_s\preceq kc, \tag{36}$$

Proposed TFSR-KELM algorithm can be summarized as follows.

4 Experimental results

4.1 Performance evaluation of TFSR-ELM

In this section, in order to evaluate the properties of our framework, we perform the experiments on a non-text dataset obtained from the UCI machine learning repository.

4.1.1 Dataset

UCI dataset The UCI machine learning repository contains Iris, Wine, Segment, Heart, Diabetes, Flare Solar, and Splice dataset (Table 1).

ORL and yale face dataset The ORL face database [38] contains 10 different images for each of the 40 distinctive subjects. Subjects are photographed at different times, with

Table 1 UCI data set used in the experiments

Data sets	Cluster	Dimensionality	Sample Size
Iris	3	4	160
Wine	3	3	178
Segment	7	7	230
Heart	8	13	270
Diabetes	3	8	768
Flare Solar	5	9	1066
Splice	8	60	3175

varying lighting conditions, facial expressions and facial details. All images are captured against a dark homogeneous background with the subjects in an upright, frontal position with a small tolerance for side movement, as shown in Fig. 2. The Yale face database [39] contains 165 grayscale images of 15 individuals. There are 11 images per subject, one per different facial expression or configuration: center-light, with glasses, happy, left-light, without glasses, normal, right-light, sad, sleepy, surprised, and winking, as shown in Fig. 3).

MNIST dataset MNIST dataset [41] has a training set of 60,000 examples and a test set of 10,000 examples of size 28 $\times$ 28 (Fig. 4).

USPS dataset Experiments are conducted on the benchmark USPS handwritten digits dataset. USPS is composed of 7291 training images and 2007 test images of size 16 $\times$ 16. Each image is represented by a 256-dimensional vector (Fig. 5).

4.1.2 Classification performance assessment

For the UCI categorization data, the different attributes were used to classify the data in a dataset. The number of hidden neurons was set to 1000 for the first two data sets (Iris and Wine), and 2000 for the rest data sets. The hyper-parameter λ was selected from the exponential sequence $\{10^{-5}, 10^0, \ldots, 10^5\}$ based on the clustering performance.

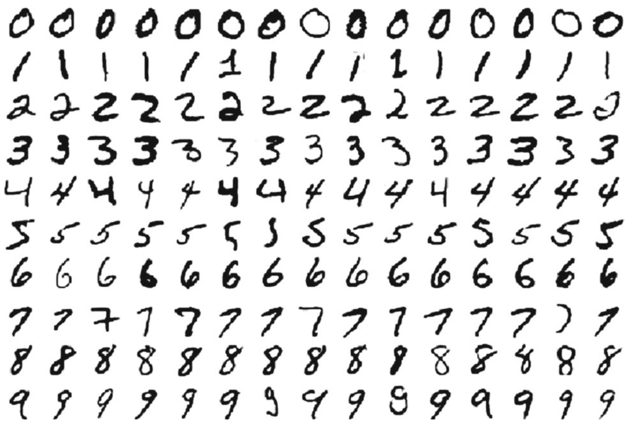


Fig. 4 Sample images in the MNIST dataset

For US-ELM, TFSR-ELM, the same affinity matrix was used, but the dimension of the embedded space was selected independently. We ran K-means algorithm was run in the original space and the embedded spaces of US-ELM, TFSR-ELM, 200 times independently. There are 200 rounds for all algorithms to get the average accuracy.

In Table 2, the TFSR-ELM method, delivered more stable results across all the datasets and is highly competitive in most of the data sets. It obtained the best classification accuracy among all methods. Hence, as discussed in the above section, TFSR-ELM algorithm possesses the traditional advantages over other methods in terms of classification accuracy.

As shown in Table 3, TFSR-ELM is obviously more superior than US-ELM in training time for almost all these datasets. Although the TFSR-ELM algorithm has lower training time, its training time is more compared with the traditional ELM algorithm.

To test the performance of US-ELM for noisy data, experiments were conducted on a series of data sets with different levels of noise. Details of the relationship between the noise level and classification accuracy are showed in Table 4. The

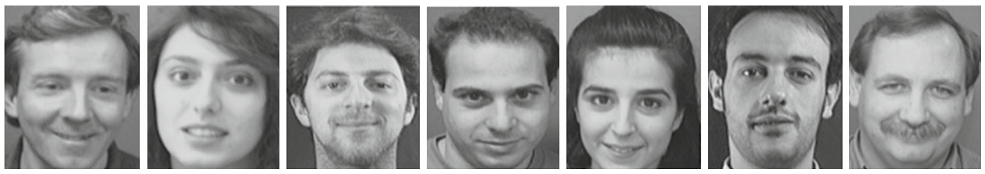


Fig. 2 Sample images in the ORL database

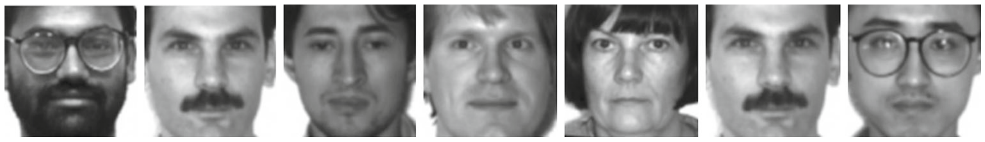


Fig. 3 Sample images in the YALE database

Fig. 5 Sample images in the USPS dataset

Table 2 Performance comparison of the proposed TFSR-ELM

Data set	k-means		ELM		USELM		TFSR-ELM	
	Avr.	Best	Avr.	Best	Avr.	Best	Avr.	Best
Iris	82.20 ± 12.8	89.30	88.22±15.23	98.71	86.02 ± 13.32	97.33	88.12 ± 17.8	97.92
Wine	92.32 ± 3.21	95.63	92.63±0.45	96.58	93.63 ± 0.21	96.63	95.77 ± 0.60	96.73
Segment	60.56 ± 5.80	67.11	64.43 ± 0.53	74.54	64.43 ± 0.53	74.50	69.22 ± 0.70	76.43
Heart	55.62 ± 6.27	61.23	60.9 ± 0.23	65.33	58.92 ± 0.23	62.22	70.22 ± 0.30	71.32
Diabetis	89.01 ± 2.64	90.21	93.58 ± 0.51	95.91	92.67 ± 1.20	93.37	94.23 ± 0.31	95.75
Flare solar	90.30 ± 6.67	91.20	92.15 ± 0.30	92.26	92.24 ± 0.21	93.32	93.27 ± 0.62	96.21
Splice	50.64 ± 5.29	58.27	55.35 ± 5.11	62.85	53.60 ± 5.80	61.54	61.60 ± 6.21	67.33
USPS	63.72 ± 3.43	71.44	75.68 ± 4.82	87.90	76.30 ± 5.90	88.41	82.50 ± 6.15	90.20
MNIST	65.88 ± 4.10	73.89	80.24 ± 5.56	93.12	79.30 ± 6.10	91.30	84.70 ± 7.21	93.23

Table 3 Training time (seconds) comparison between K-means, ELM, US-ELM, TFSR-ELM

Data set	K-means	ELM	US-ELM	TFSR-ELM
Iris	0.006	0.002	0.050	0.030
Wine	0.007	0.005	0.055	0.050
Segment	0.029	0.009	1.763	1.023
Heart	0.035	0.010	2.198	1.987
Diabetis	0.022	0.008	1.532	0.967
Flare Solar	0.033	0.013	2.045	1.923
Splice	0.062	0.022	3.133	2.996
USPS	0.101	0.068	3.242	3.356
MNIST	0.094	0.071	2.293	2.137

Table 4 The relationship between noise level and classification accuracy

Noise level	0.01	0.02	0.03	0.04	0.05	0.06
US-ELM	0.928	0.912	0.908	0.892	0.871	0.863
TFSR-ELM	0.956	0.954	0.953	0.953	0.953	0.953

noise experiment was performed on the Wine data set. The zero mean and different standard deviations of Gaussian noise were added to the training samples. As shown in Table 4, TFSR-ELM is obviously superior than US-ELM in terms of classification accuracy. However, with the increase of noise, the TFSR-ELM is relatively stable. Therefore, it shows very good robustness in terms of noise.

4.2 Performance evaluation of TFSR-KELM

In this section, in order to assess the effectiveness of the proposed methods in multi class classification problems, a study in conducted on the performance of the proposed methods TFSR-KELM for face recognition on two benchmarking face databases, namely, Yale and ORL. The target datasets are generated by rotating the original dataset clockwise 3 times by 10^0, 30^0, and 50^0, as shown in Fig. 6. Particularly, the greater the rotation angle is, the more complex will be the resulting problem becomes. Thus three faces learning problems are built for each face dataset.

Because choosing the algorithm parameters for the kernel methods still is a hot field in research, in the algorithm, parameters are generally preset. In order to evaluate the performance of the algorithm, a set of the prior parameters is first given and then the best cross-validation mean rate among the set is used to estimate the generalized accuracy in this work. Five-fold cross validation is used on the training data for parameter selection. Then, the mean of experimental obtained for the testing data is used to evaluate the performance. The overall accuracy (i.e., the percentage of the correctly labeled samples over the total number of samples) is chosen as the reference classification accuracy measure. The performance of TFSR-ELM is compared with K-means, traditional ELM, US-ELM, and TFSR-ELM. For each evaluation, five rounds of experiments are repeated with randomly selected training data, and the average result is recorded as the final classification accuracy in Table 5.

The overall accuracy of LS-SVM is lower than any other classifier for all tasks. With the increase in rotation angle, the

Table 5 Means (%) of classification accuracy(ACC) of all algorithm on Yale and ORL with different rotation angles

Face data		Algorithms			
		LS-SVM	US-ELM	ELM	TFSR-KELM
Yale	10^0	72.67	79.90	78.90	90.31
	30^0	71.82	76.66	76.29	85.89
	50^0	69.38	73.32	75.32	81.56
ORL	10^0	80.32	85.92	82.67	92.71
	30^0	78.35	83.93	81.21	88.23
	50^0	71.39	78.85	79.36	86.67

classification performance of all classifiers degrades gradually. However, for TFSR-KELM, the performance seems to degrade more slowly than the other methods. Exceptionally, traditional ELM exhibits competitive performance to some extent compared to the other methods, particularly on more complex datasets. As shown in Table 5, the TFSR-KELM method delivers more stable results across all the datasets and is highly competitive for most of the datasets. It obtains the best classification accuracy more times than any other method. Hence, as discussed in the above section, TFSR-KELM possesses overall advantages over other methods in the sense of classification accuracy.

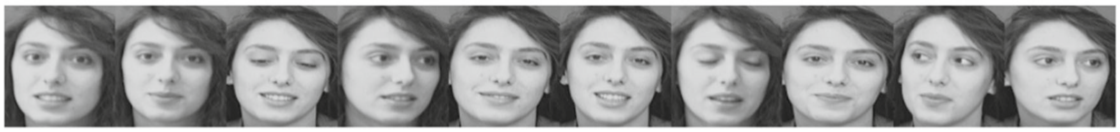
5 Conclusions and future research

The issue of free sparse learning based on transfer ELM was addressed. The basic idea of TFSR-ELM is to use a lot of source data to build a high-quality classification model. Starting from the solution of independent ELMs, it was evident that the addition of a new term in the cost function (which penalizes the diversity between consecutive classifiers) leads to transfer of knowledge. The results showed that the proposed method that uses TFSR-ELM can effectively improve the classification by learning free sparse knowledge and is robust to different sizes of training data.

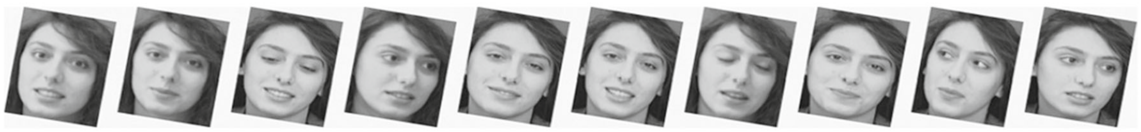
In addition, a novel transfer free sparse representation kernel extreme learning machine (TFSR-KELM) based on the kernel extreme learning machine was proposed with respect to the TFSR-ELM. Experimental results showed the effectiveness of the proposed algorithm. In the future, a study will be conducted on verify whether this method can be extended to transfer knowledge across different domains. It would be interesting to define a norm between the transformations obtained in such a setting. This norm can be used to decide what type of knowledge could be transferred based on domain similarity.

(a) Yale faces of object

(b) Yale faces of object with rotation of 10^0



(c) ORL faces of object



(d) ORL faces of object with rotation of 10^0

Fig. 6 Face image samples of the Yale and ORL datasets. **a** Yale faces of object. **b** Yale faces of object with rotation of 10^0. **c** ORL faces of object. **d** ORL faces of object with rotation of 10^0

Acknowledgments We appreciate the anonymous reviewers for the valuable comments. This work was supported by the National Natural Science Foundation of China (No. 61473252) and the National Natural Science Foundation of China (No. 61375049).

References

1. Chen CP, Zhang CY (2014) Data-intensive applications, challenges, techniques and technologies: A survey on big data. Inform Sci 275:314–347
2. Zhou ZH, Chawla N, Jin Y, Williams G (2014) Big data opportunities and challenges: Discussions from data analytics perspectives. IEEE Comput Intell Mag 9(4):62–74
3. Kasun LLC, Zhou H, Huang GB, Vong CM (2013) Representational learning with extreme learning machine for big data. IEEE Intell Syst 28(6):31–34
4. Platt J (1999) Fast training of support vector machines using sequential minimal optimization, Advances in kernel methods—support vector learning 3
5. Suykens JAK, Vandewalle J (1999) Least squares support vector machine classifiers. Neural Process Lett 9(3):293–300
6. Hetch-Neilsen R (1989) Theory of the backpropagation neural network, In International Joint Conference on Neural Networks 593–605
7. Cortes C, Vapnik VN (1995) Support vector networks. Mach Learn 20(3):273–297
8. Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Networks 17(4):879–892
9. Han F, Huang DS (2006) Improved extreme learning machine for function approximation by encoding a priori information. Neurocomputing 69:2369–2373
10. Kim CT, Lee JJ (2008) Training two-layered feedforward networks with variable projection method. IEEE Trans Neural Networks 19:371–375
11. Zhu QY, Qin AK, Suganthan PN, Huang GB (2005) Evolutionary extreme learning machine. Patt Recogn 38:1759–1763
12. Wang Y, Cao F, Yuan Y (2011) A study on effectiveness of extreme learning machine. Neurocomputing 74:2483–2490
13. Li GH, Liu M, Dong MY (2010) A new online learning algorithm for structure-adjustable extreme learning machine. Comp Math Appl 60:377–389
14. Huang GB, Chen L (2007) Convex incremental extreme learning machine. Neurocomputing 70:3056–3062
15. Huang GB, Li MB, Chen L, Siew CK (2008) Incremental extreme learning machine with fully complex hidden nodes. Neurocomputing 71:576–583
16. Feng G, Huang GB, Lin Q, Gay R (2009) Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Trans Neural Networks 20:1352–1357
17. Rong HJ, Ong YS, Tan AH, Zhu Z (2008) A fast pruned-extreme learning machine for classification problem. Neurocomputing 72:359–366
18. 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 Cyber Part B Cyber 39:1067–1072
19. Yoan M, Sorjamaa A, Bas P, Simula O, Jutten C, Lendasse A (2010) OP-ELM: optimally pruned extreme learning machine. IEEE Trans Neural Networks 21:158–162
20. Lan Y, Soh YC, Huang GB (2010) Constructive hidden nodes selection of extreme learning machine for regression. Neurocomputing 73:3191–3199
21. Zong WW, Huang GB, Chen Y (2013) Weighted extreme learning machine for imbalance learning. Neurocomputing 101:229–242
22. Fu H, Vong CM, Wong PK, et al (2014) Fast detection of impact location using kernel extreme learning machine. Neural Comp Appl 1–10
23. Dai W, Yang Q, Xue G, Yu Y (2007) Boosting for transfer learning. Proc. 24th Int Confe Mach Learn 193–200
24. Pan SJ, Yang Q (2010) A survey on transfer learning. IEEE Trans Know Data Eng 22:1345–1359
25. Pan J, Wang X, Cheng Y et al (2014) Multi-source transfer ELM-based Q learning. Neurocomputing 137:57–64
26. Zhang L, Zhang D (2015) Domain adaptation extreme learning machines for drift compensation in E-nose systems. IEEE Trans Instru Measur 64:1790–1801. doi:10.1109/TIM.2014.2367775
27. Huang G, Song S, Gupta JND, Wu C (2014) Semi-supervised and unsupervised extreme learning machines. IEEE Trans Cyber 44(12):2405–2417
28. Huang GB (2014) An insight into extreme learning machines: Random neurons, random features and kernels. Cogn Comp 6(3):376–390
29. Huang GB, Bai Z, Kasun LLC et al (2015) Local receptive fields based extreme learning machine. IEEE Comp Intell Magaz 10(2):18–29
30. Tang J, Deng C, Huang GB (2015) Extreme learning machine for multilayer perceptron. IEEE Trans Neural Networks Learn Syst. doi:10.1109/TNNLS.2015.2424995
31. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: Theory and applications. Neurocomputing 70:489–501
32. Widrow B, Greenblatt A, Kim Y, Park D (2013) The no-prop algorithm: A new learning algorithm for multilayer neural networks. Neural Networks 37:182–188
33. Rao CR, Mitra SK (1971) Generalized inverse of matrices and its applications. Wiley, New York
34. Huang GB, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cyber Part B Cyber 42:513–529
35. Belkin M, Niyogi P (2003) Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput 15(6):1373–1396
36. Zheng M, Bu J, Chen C, Wang C et al (2011) Graph regularized sparse coding for image representation. IEEE Trans Image Process 20(5):1327–1336
37. Lee H, Battle A, Raina R, Ng A (2006) Efficient sparse coding algorithms. Advances in Neural Information Processing Systems 801–808
38. ORL Face Database (2005) AT&T Laboratories Cambridge, http://www.camorl.co.uk/facedatabase.html
39. Yale Face Database (2005) Columbia Univ., http://www.cs.columbia.edu/belhumeur/pub/images/yalefaces/
40. The USPS dataset, http://www-i6.informatik.rwth-aachen.de/~keysers/usps.html
41. The MNIST dataset, http://yann.lecun.com/exdb/mnist/
42. Feng L, Ong YS, Lim MH (2013) ELM-guided memetic computation for vehicle routing. IEEE Intell Syst 8(6):38–41
43. Cambria E, Huang GB, Kasun LLC et al (2013) Extreme learning machines [trends & controversies]. IEEE Intell Syst 28(6):30–59
44. Kan EM, Lim MH, Ong YS et al (2013) Extreme learning machine terrain-based navigation for unmanned aerial vehicles. Neural Comp Appl 22(3–4):469–477