Neurocomputing 128 (2014) 88–95

Contents lists available at ScienceDirect

Neurocomputing

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

Clustering in extreme learning machine feature space

Qing He [a,*], Xin Jin [a,b], Changying Du [a,b], Fuzhen Zhuang [a], Zhongzhi Shi [a]

[a] Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[b] Graduate University of Chinese Academy of Sciences, Beijing 100049, China

ARTICLE INFO

Article history:
Received 28 August 2012
Received in revised form
3 December 2012
Accepted 21 December 2012
Available online 24 October 2013

Keywords:
Extreme learning machine (ELM)
ELM feature space
Data clustering
Nonnegative matrix factorization (NMF)
ELM kMeans
ELM NMF clustering

ABSTRACT

Extreme learning machine (ELM), used for the "generalized" single-hidden-layer feedforward networks (SLFNs), is a unified learning platform that can use a widespread type of feature mappings. In theory, ELM can approximate any target continuous function and classify any disjoint regions; in application, many experiment results have already demonstrated the good performance of ELM. In view of the good properties of the ELM feature mapping, the clustering problem using ELM feature mapping techniques is studied in this paper. Experiments show that the proposed ELM kMeans algorithm and ELM NMF (nonnegative matrix factorization) clustering can get better clustering results than the corresponding Mercer kernel based methods and the traditional algorithms using the original data. Moreover, the proposed methods have the advantage of being more convenient to implementation and computation, as the ELM feature mapping is much simpler than the Mercer kernel function based feature mapping methods.

1. Introduction

Cluster analysis or clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar to each other than to those in other clusters. No matter in the past or at present, being an important problem in data mining, clustering has attracted many researchers' attentions and efforts, resulting in many cluster models. These models include hierarchical clustering [1,2], centroid-based clustering (such as kMeans [3]), clustering models related to statistics [4] and density based clustering [5,6]. Based on matrix theory, many methods are also used in clustering, such as a latent semantic indexing method [7], spectral clustering [8] and nonnegative matrix factorization (NMF) [9]. It is acknowledged that through a nonlinear data transformation, the data will become more linear separable in the transformed high dimensional feature space, that is, data structure becomes much simpler. To take advantage of the feature mapping techniques, kernel methods have been used for clustering [10–12], and better results are got. Being an explicit feature mapping technique, ELM feature mapping [13,14] is more convenient than the kernel function based methods. In addition, replacement of SVM kernels with random ELM kernels can get more satisfactory results for classification and regression [15–17]. So, studying the clustering problem using ELM feature mapping techniques is meaningful.

The extreme learning machine (ELM) [18,19] was originally proposed as a new learning algorithm for single-hidden layer feedforward neural networks (SLFNs). Unlike those conventional iterative implementations, ELM randomly chooses input weights and hidden biases and then analytically determines the output weights of SLFNs. Afterwards, ELM was extended to the "generalized" SLFNs where the nodes need not be neuron alike [13,14]. Different from traditional learning algorithms for a neural type of SLFNs [20], ELM aims to reach not only the smallest training error but also the smallest norm of output weights. The learning process of ELM includes two steps. First, the input vectors are mapped into a feature space, which are the hidden layer output vectors. Then, the standard optimization method is used to find the solution that minimizes the training errors.

As a unified learning method for regression and multiclass classification, ELM tends to have better scalability and achieve similar or much better generalization performance at much faster learning speed than traditional SVM and LS-SVM [21,22]. Initially, ELM [18,19] and its variants [23–26] mainly focus on the regression applications. Latest development of ELM has shown some relationships between ELM and SVM [15,17]. To the best of our knowledge, [15] is the first paper that tries to use ELM kernels in SVM, and Huang et al. [17] demonstrated that SVM's maximal separating margin property and ELM's minimal norm of output weights property are actually consistent. Further, in theory ELM can be linearly extended to SVMs instead of only replacing SVM kernels with ELM kernels. Also, ELM uses less optimization

* Corresponding author.
E-mail addresses: heq@ics.ict.ac.cn (Q. He), sdjinxin@gmail.com, jinx@ics.ict.ac.cn (X. Jin).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.12.063

constraints. Frénay et al. [16] proposed a new parameter-insensitive kernel inspired from extreme learning and used it in nonlinear SVR (Support Vector Regression). Their method significantly reduced the computational complexity. The reason why ELM kernel can be used in SVM and get better performance may lie in that ELM has *Universal Approximation Capability* [13,14,23], which means that ELM can approximate any continuous target functions, and *Classification Capability* [21], which implies that ELM can classify any disjoint regions. In view of the advantages of the ELM and their better performance in regression and classification, we conjecture that clustering in ELM feature space would also get excellent results.

As the extreme learning machine has been demonstrated to have better performance in regression and classification, and in theory ELM has *Universal Approximation Capability* and *Classification Capability*, this means after ELM feature transformation process, the data structure becomes much simpler. Also, using ELM feature mapping techniques in SVM and its variants can get improved performance [15,17], which is a manifestation of the usefulness of the ELM feature mapping techniques. Inspired by the fact that Mercer kernel based feature transformation methods can be used in clustering [10–12] and in NMF [27] to get better results, in this paper, we explore the methods directly using the ELM feature mapping techniques in clustering, resulting in the kMeans algorithm in ELM feature space, and explore the NMF in ELM feature space and then do the clustering in the low-dimensional representation got from the NMF.

The rest of the paper is organized as follows: in Section 2, a brief review of ELM is given. Section 3 introduces our clustering algorithms using ELM feature mapping techniques. Extensive experimental results on clustering are presented in Section 4. Finally, some concluding remarks are provided in Section 5.

2. Preliminary knowledge

A word about our notations. A^T denotes the transpose of a matrix A, tr(A) means the trace operator of the corresponding matrix A. The inner product is explicitly represented by the operator $\langle \alpha \cdot \beta \rangle$ or use $\alpha^T\beta$ for convenience. $\|\cdot\|$ denotes the Frobenius norm, and $A \geq 0$ represents the matrix with nonnegative values.

Originally proposed for the single hidden-layer feedforward neural networks, extreme learning machine (ELM) has been extended to the generalized SLFNs where the hidden layer need not be neuron alike [13,14]. In ELM, the hidden layer parameters, which are randomly initialized, need not be tuned. The output function of ELM for generalized SLFNs (take one output node case as an example) is

$$f(x) = \sum_{i=1}^{L} \beta_i h_i(x) = h(x)\beta \tag{1}$$

where $\beta = [\beta_1, \ldots, \beta_L]^T$ is the vector of the output weights between the L hidden-layer nodes and the output node and $h(x) = [h_1(x), \ldots, h_L(x)]$ is the output (row) vector of the hidden layer with respect to the input x. $h(x)$ actually maps the data from the d-dimensional input space to the L-dimensional hidden-layer feature space (*ELM feature space*) H, and thus, $h(x)$ is indeed a feature mapping. For the binary classification applications, the decision function of ELM is

$$f(x) = \text{sign}\,(h(x)\beta). \tag{2}$$

Different from traditional learning algorithms [20], ELM tends to reach not only the smallest training error but also the smallest norm of output weights [21]. The classification problem for the proposed constrained-optimization-based ELM with a single output node can be formulated as

$$\begin{aligned} &\text{Minimize}: \quad L_{P_{ELM}} = \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2 \\ &\text{Subject to}: \quad h(x_i)\beta = t_i - \xi_i, \quad i = 1, \ldots, N \end{aligned} \tag{3}$$

Based on the KKT conditions, one solution can be obtained as follows. The output function of the ELM classifier is

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

where $T = [t_1, \ldots, t_N]^T$, $H = [h(x_1), \ldots, h(x_N)^T$. For multiclass classifications, among all the multiclass labels, the predicted class label of a given testing sample is closest to the output of ELM classifier.

In the implementation of ELM, it is found that the generalization performance of ELM is not sensitive to the dimensionality of the feature space (L) and good performance can be reached as long as L is large enough. Setting $L = 1000$ can always get satisfactory results, no matter whatever the size of the training data sets is [21].

3. Clustering in ELM feature space

ELM maps the original data into the ELM feature space, and then, by constructing a linear decision function (Eq. (1)), find the classifier in the feature space, which can get better results. Also, kernel methods have been used to do the clustering in the kernel space [10–12], and they also obtain encouraging performance. Since classification in the ELM feature space can get better results, we introduce the methods to do the clustering in the ELM feature space.

3.1. ELM feature mapping process

Suppose each input data is a d-dimensional vector $x = [x_1, \ldots, x_i, \ldots, x_d]^T$, through a single hidden layer feed forward neural networks, ELM will map the data into the L-dimensional ELM feature space (hidden layer feature space) H, and L is the number of the hidden nodes used in the feature mapping process, as is shown in Fig. 1.

The feature mapping can be formally described as follows:

$$\begin{aligned} h(x) &= [h_1(x), \ldots, h_i(x), \ldots, h_L(x)]T \\ &= [G(a_1, b_1, x), \ldots, G(a_i, b_i, x), \ldots, G(a_L, b_L, x)]^T \end{aligned} \tag{5}$$

where $G(a_i, b_i, x)$ is the output of the i-th hidden node, a_i is a d-dimensional weight vector between the d input nodes and the ith hidden-node, b_i is the bias of the ith hidden-node. The parameters used in the mapping process, $(a_i, b_i)_{i=1}^{L}$, can be randomly generated according to any continuous probability distribution, and they need

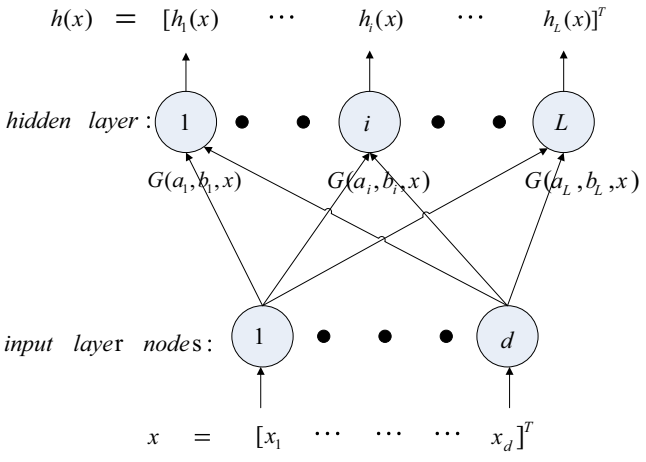


Fig. 1. ELM feature mapping process.

not be tuned, so ELM feature mapping is very efficient. Also, different from SVM, LS-SVM and PSVM, the feature mapping $h(x)=[h_1(x),\ldots,h_i(x),\ldots,h_L(x)]^T$ is usually known to users, and we can directly use it instead of using the kernel functions, which maybe simpler in some problems. According to [13,14], almost all nonlinear piecewise continuous functions can be used as the hidden-node output functions, and thus, the feature mappings used in ELM can be very diversified. The most frequently used function is the Gaussian function in Eq. (6), some other nonlinear piecewise continuous functions that satisfy the ELM universal approximation capability theorems can be found in [21]:

$$G(a,b,x)=\exp(-b\|x-a\|^2). \tag{6}$$

Using Eq. (5), it is very easy to transform the data from the original input feature space into the ELM feature space. The feature mapping process is explicit, and we can choose the functions that have the desired properties for particular problems to be solved. In SVM and its variants, the feature mapping may be unknown. Using ELM feature mapping techniques, we can get the exact data in the feature space, which is more convenient than the inner product form information in the feature space that using kernel functions.

3.2. The rationality of ELM feature mapping

On behalf of the ELM feature mapping, ELM can have many good properties, and the clustering problems can also take advantage of these feature mappings. According to ELM learning theory, a widespread type of feature mappings $h(x)$ can be used in ELM so that ELM can approximate any continuous target functions (refer to [13,23] for details). That is, given any target continuous function $f(x)$, there exists a series of β_i such that

$$\lim_{L\to+\infty}\|f_L(x)-f(x)\|=\lim_{L\to+\infty}\left\|\sum_{i=1}^{L}\beta_i h_i(x)-f(x)\right\|=0 \tag{7}$$

Seen from Eq. (7), it can be recognized that ELM can approximate any continuous functions just using a linear function in the ELM feature space, whereas a linear function in the original feature space that can approximate the continuous function may not exist. On the other hand, in SVM, LS-SVM, and PSVM, which are implemented by using kernel functions, the feature mapping $\phi(x_i)$ may be unknown, usually not every feature mapping to be used in SVM and its variants satisfy the universal approximation condition. Obviously, a learning machine with a feature mapping which does not satisfy the universal approximation condition cannot get satisfactory results.

Huang et al. also proved the classification capability of ELM [21].

Definition 3.1. A closed set is called a region regardless of whether it is bounded or not.

Lemma 3.1. *Given disjoint regions $K_1,K_2,\ldots,K_m$ in R^d and the corresponding m arbitrary real values $c_1,c_2,\ldots,c_m$, and an arbitrary region X disjointed from any K_i, there exists a continuous function f(x) such that $f(x)=c_i$ if $x\in K_i$ and $f(x)=c_0$ if $x\in X$, where c_0 is an arbitrary real value different from $c_1,c_2,\ldots,c_m$.*

Theorem 3.1. *Given a feature mapping h(x), if $h(x)\beta$ is dense in $C(R^d)$ or in C(M), where M is a compact set of R^d, then a generalized SLFN with such a hidden-layer mapping h(x) can separate arbitrary disjoint regions of any shapes in R^d or M.*

Theorem 3.1 shows that if $h(x)\beta$ can approximate any continuous functions, then ELM can separate any decision regions regardless of shapes of these regions, that is, ELM can classify any data sets if they does not overlap. All these good properties of the ELM feature mapping may help the clustering problem in the ELM feature space.

3.3. Clustering using ELM feature mapping techniques

With the notion that performing a nonlinear data transformation into some high dimensional feature space increases the probability of the linear separability of the patterns within the transformed space and therefore simplifies the associated data structure, many Mercer kernel based clustering algorithms in the transformed feature space have been proposed [10–12]. However, in these methods, since the feature mapping $\phi(x_i)$ is always implicit, they must make use of the kernel functions, which is not efficient for computation. Furthermore, in sometimes, when the explicit representation in the transformed feature space is required, the kernel function method does not work. Also, without the explicit form of the feature mapping, we cannot guarantee that the feature mapping to be used in clustering satisfies the universal approximation condition, which may affect the performance of the algorithm. Thus, the explicit feature mapping methods such as ELM feature mapping may be more appropriate.

3.3.1. kMeans algorithm in ELM feature space

Compared to the kernel based methods, clustering in the ELM feature space is more convenient. First, we transform the original data into the ELM feature space using Eq. (5). The mapping is very intuitive and straightforward (see Fig. 1), and according to the ELM universal approximation conditions, many nonlinear piecewise continuous functions can be used as the hidden-node output functions. The only parameter needs to be specified by the users is the number of the nodes in the hidden layer. According to the ELM universal approximation conditions and classification capability (Eq. (7)), a very large number of nodes can guarantee that the data will become linear separable, so we can set the parameter to a large enough number. After transforming the data into the ELM feature space, the traditional clustering method can be used directly. In this part, we use the simple kMeans algorithm and call the kMeans algorithm in ELM feature space as ELM kMeans algorithm for short. kMeans clustering problem can be described as follows: given a set of observations $(x_1,x_2,\ldots,x_m)$, where each observation is a d-dimensional real vector, kMeans clustering aims to partition the m observations into k sets $(k\le m)$ $S=\{S_1,S_2,\ldots,S_k\}$ so as to minimize the within-cluster sum of squares (WCSSs):

$$\arg\min_{S}\sum_{i=1}^{k}\sum_{x_j\in S_i}\|x_j-\mu_i\|^2 \tag{8}$$

where μ_i is the mean of points in S_i. The detailed description of the ELM kMeans algorithm is shown in Algorithm 1.

Algorithm 1. ELM kMeans algorithm.

Input:
k: the number of clusters,
L: the number of the hidden-layer nodes,
D: a data set containing m objects.

Output:
A set of k clusters.

Method:
1: Mapping the original data objects in D into the ELM feature space H using $h(x)=[h_1(x),\ldots,h_i(x),\ldots,h_L(x)]^T$;
2: Arbitrarily choose k objects from H as the initial cluster centers;
3: **repeat**
4: (Re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
5: Update the cluster means, i.e., calculate the mean value of the objects for each cluster;

6: **until** no change in the cluster centers or reached the maximal iteration number limit.
7: **return** A set of k clusters.

3.3.2. Clustering based on NMF in ELM feature space

Nonnegative matrix factorization (NMF) is a recently proposed linear method for finding low-dimensional representation of nonnegative high-dimensional data. Having the part-based representation property [28], NMF and its variations have been applied to a variety of applications, such as image classification, face and object recognition, document clustering, etc. [29]. It has been shown that the NMF based document clustering method surpasses SVD clustering methods and the spectral clustering methods according to the clustering accuracy [9]. NMF is a linear model, using nonlinear feature mapping techniques, it will be able to deal with nonlinear correlation in data, so kernel methods have been used in NMF. It must be aware that using kernel methods in NMF is not so straightforward, therefore Buciu et al. [30] require that the kernel functions must be differentiable functions and Zhang et al. [27] makes some approximations in the computation. Obviously, as an explicit feature mapping technique, ELM feature mapping can be used in NMF as well, and it could be much simpler.

As described in the previous section, ELM feature mapping has many good properties. As ELM feature mapping is explicit, NMF in the ELM feature space is more convenient than the kernel based methods. Clustering based on NMF in ELM feature space is also very straightforward. First, we transform the original data into the ELM feature space using Eq. (5). The mapping is very intuitive (see Fig. 1), and according to the ELM universal approximation conditions, many nonlinear piecewise continuous functions can be used as the hidden-node output functions. The only parameter need to be specified by the users is the number of the nodes in the hidden layer. Then, after applying traditional NMF methods in the ELM feature space, the low-dimensional representation of the data can be easily got. Using any clustering methods to the low-dimensional data can obtain the clustering result we want. For the simplicity, we choose the widely used kMeans algorithm in the final step. The detailed description of clustering based on NMF in ELM feature space is shown in Algorithm 2. It must be noted that only the activation functions that can only get nonnegative outputs can be used in Step 1 of Algorithm 2, or some postprocess be done to the result of the feature mappings to guarantee that they are nonnegative.

Algorithm 2. Clustering based on NMF in ELM feature space (ELM NMF).

Input:
k: the number of clusters,
L: the number of the hidden-layer nodes,
D: a data set containing m objects,
ε: A small threshold $\varepsilon > 0$.
Output:
A set of k clusters.
Method:
1: Mapping the original data objects in D into the ELM feature space using $h(x) = [h_1(x), \ldots, h_i(x), \ldots, h_L(x)]^T$, all the data will form a data matrix in ELM feature space as a $L \times m$ matrix $H = [h(x_1), \ldots, h(x_j), \ldots, h(x_m)]^T$.
2: Generate initial nonnegative matrices W^0 and B^0 with dimensions $L \times k$ and $k \times m$ respectively.
3: **repeat**
4: For given W, update the matrix B as $B_{au}^{t+1} = B_{au}^t (W^T H)_{au} / (W^T W B^t)_{au}$.
5: For given B, update the matrix W as $W_{ia}^{t+1} = W_{ia}^t (HB^T)_{ia} / (W^t B B^T)_{ia}$.
6: $err = \max\left\{ \frac{\| W^{t+1} - W^t \|}{\sqrt{Lk}}, \frac{\| B^{t+1} - B^t \|}{\sqrt{km}} \right\}$
7: **until** $err < \varepsilon$ or reached the maximal iteration number limit.
8: Using kMeans algorithms to cluster the data in the low-dimensional space B (details of kMeans algorithm can be found in Algorithm 1);
9: **return** A set of k clusters.

4. Experiments and results

In this section, the performance of the kMeans clustering algorithm in ELM feature space and the clustering method based on NMF in ELM feature space is compared with the classical clustering methods and the corresponding kernel based methods.

4.1. Data sets

Three data sets are used in the experiments. Two of them are from the UCI Machine Learning Repository [31] and the third one is a document corpus. The description and some preprocesses about these data sets are given below:

(1) Synthetic Control Chart Time Series Data Set (Synthetic Control for short) [31]: This data set contains 600 examples of control charts synthetically generated by the process in Alcock and Manolopoulos. There are six different classes of control charts, each class has 100 examples. The data has 60 real attributes, and we normalize each attribute into [0,1] using the min–max normalization method, mainly consider that the NMF requires that the data must be nonnegative. For the algorithms based on ELM, same as the ELM implementation reported by Zhu et al. [32], each attribute is normalized into $[-1,1]$. Min–max normalization performs a linear transformation on the original data. Suppose that $\min_A$ and $\max_A$ are the minimum and maximum values of an attribute, A, min–max normalization maps a value, v, of A to v' in the range $[new\_min, new\_max]$ by computing

$$v' = \frac{v - \min_A}{\max_A - \min_A}(new\_\max - new\_\min) + new\_\min. \quad (9)$$

(2) Libras Movement Data Set [31]: The data set contains 15 classes, each contains 24 instances, where each class references to a hand movement type in LIBRAS. As the data are gathered after a video pre-processing procedure, and a mapping operation has been done by the data provider to facilitate the analysis of these data, we use the data provided by the UCI directly without any preprocessing and normalization.
(3) NIST Topic Detection and Tracking (TDT2) corpus [33]: The TDT2 corpus consists of data collected during the first half of 1998 and taken from 6 sources, including 2 newswires (APW, NYT), 2 radio programs (VOA, PRI) and 2 television programs (CNN, ABC). It consists of 11 201 on-topic documents which are classified into 96 semantic categories. In this experiment, those documents appearing in two or more categories were removed, and only the largest 30 categories were kept. Also, for the categories that contain more than 100 documents, we randomly selected 100 documents from each category, thus leaving us with 2651 documents in total. We use the *tf-idf* weight scheme to compute the weight of each term in each document, i.e., $tf\text{-}idf = tf \times idf$, and tf is weighted as $1 + \log(tf)$ if $tf > 0$, $idf = \log(N/df)$, and each document vector is normalized to have length 1. In respect that there are too

many terms in the corpus, principal component analysis [34] is used to select the more effective features and the dimension of the data set is reduced to 200 (NMF based methods use the original data as they can find the low-dimensional representations themselves).

4.2. Evaluation of clustering

Typical objective functions in clustering formalize the goal of attaining high intra-cluster similarity and low inter-cluster similarity. This is an *internal criterion* for the quality of a clustering. But good scores on an internal criterion do not necessarily translate into good effectiveness in an application. An alternative to internal criteria is direct evaluation in the application of interest. We can then compute an *external criterion* that evaluates how well the clustering matches the gold standard classes. Xiong et al. [35] explained many evaluation measures, and two *external criteria* of clustering quality will be used in this paper.

(1) *Purity* is a simple and transparent evaluation measure: To compute *purity*, each cluster is assigned to the class which is most frequent in the cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned documents and dividing by N, N is the number of the samples to be clustered. Formally,

$$purity(S,C)=\frac{1}{N}\sum_k \max_j |s_k \cap c_j| \tag{10}$$

where $S=\{s_1,s_2,\ldots,s_K\}$ is the set of clusters and $C=\{c_1,c_2,\ldots,c_J\}$ is the set of classes. We interpret s_k as the set of data samples in s_k and c_j as the set of data samples in c_j in Eq. (10). It is obvious that high *purity* is easy to achieve when the number of clusters is large—in particular, *purity* is 1 if each sample is viewed as a cluster. But it does not matter here in that we require the number of clusters is fixed to the true number of classes.

(2) A measure that can trade off between the quality of the clustering and the number of clusters is *normalized mutual information* or *NMI*:

$$NMI(S,C)=\frac{I(S,C)}{[H(S)+H(C)]/2} \tag{11}$$

I is mutual information:

$$\begin{aligned} I(S,C) &= \sum_k\sum_j P(s_k \cap c_j)\log\frac{P(s_k \cap c_j)}{P(s_k)P(c_j)} \\ &= \sum_k\sum_j \frac{|s_k \cap c_j|}{N}\log\frac{N|s_k \cap c_j|}{|s_k||c_j|} \end{aligned} \tag{12}$$

where $P(s_k)$, $P(c_j)$, and $P(s_k \cap c_j)$ are the probabilities of a data sample being in cluster s_k, class c_j, and in the intersection of s_k and c_j, respectively. H is entropy

$$H(S)=-\sum_k P(s_k)\log\ P(s_k)=-\sum_k \frac{|s_k|}{N}\ \log\ \frac{|s_k|}{N} \tag{13}$$

$I(S,C)$ in Eq. (12) measures the amount of information by which our knowledge of the classes increases when we are told what the clusters are. *NMI* is always a number between 0 and 1.

4.3. Compared algorithms

For the kMeans algorithm in ELM feature space proposed in Section 3.3.1 (ELM kMeans Algorithm for short), to clearly display the performance of it, we compared the following three clustering algorithms:

(1) Canonical kMeans clustering method (kMeans in short).
(2) Kernel kMeans clustering algorithm [10]: Similar to the ELM kMeans algorithm that do the clustering in the transformed feature space, kernel methods have been used for clustering [10–12], and better results are got. Specifically, we use the widely used Gaussian kernel function in the kernel clustering algorithm. The parameter selection will be given later.
(3) ELM kMeans algorithm: In this paper, we use the Gaussian function as the hidden-layer node activation function, the selection of the number of the hidden-layer nodes will be discussed in the later section.

For the clustering algorithm based on NMF in ELM feature space proposed in Section 3.3.2 (ELM NMF algorithm for short), to explicitly show its performance, we compared the following three clustering algorithms:

(1) NMF clustering algorithm: First, we use NMF [36] to get a low-dimensional representation of the original data, specifically, the dimension of the low-dimension representation is set equal to k, i.e., the number of the clusters. Then, we use the canonical kMeans algorithms in the low-dimensional space to get the clustering results.
(2) Kernel NMF clustering algorithm: First, we use the NMF in the flexible kernel space which may find nonlinear correlation in the data [27]. Consequently, we can get a low-dimensional representation of the original data, and the dimension of the low-dimension representation is the number of the clusters. Then, we use the canonical kMeans algorithms in the low-dimensional space to get the final clustering results.
(3) ELM NMF algorithm proposed in Section 3.3.2: Also, the dimension of the low-dimension representation is the same as the number of the clusters. Gaussian function is chosen as the hidden-layer nodes activation function, the selection of the number of the hidden-layer nodes will be discussed in the later section.

4.4. Quality of the clustering results

We run the clustering algorithms on the three data sets described in Section 4.1. As these algorithms are stochastic, we run every algorithm 20 times on each data set and get the average result. To get the performance of these algorithms with different cluster numbers, for each data set, we select portions of these data sets with different numbers of categories, and test the algorithms on the selected data. The number of nodes used in every data set in the ELM feature mapping and the setting of parameter σ for kernel based methods are given in Table 1 (ELM based methods are not sensitive to the parameters, but the kernel based methods need particular σ value for a specific data set). The evaluation measures used in this paper are *purity* and *NMI* described in Section 4.2.

Tables 2–4 show the clustering results of kMeans, Kernel kMeans and ELM kMeans algorithms on the *Synthetic Control*, *Libras Movement* and *TDT2* data sets, respectively. It can be seen that, on all the three data sets, ELM kMeans gets better results with all the different cluster numbers than the other two algorithms.

Tables 5–7 show the clustering results of NMF clustering, kernel NMF clustering and ELM NMF clustering algorithms on the *Synthetic Control*, *Libras Movement* and *TDT2* data sets, respectively. It can be seen that, on *Synthetic Control* data set, ELM NMF

Table 1
Parameter settings for different algorithms.

Data sets	ELM kMeans (nodes)	ELM NMF (nodes)	Kernel kMeans (σ)	Kernel NMF (σ)
Synthetic control	1000	1000	0.5	1
Libras movement	1000	1000	8	0.25
TDT2	2000	2000	4	1

Table 2
kMeans based clustering performance on *Synthetic Control* data set.

Number of clusters	Purity (100%)			NMI		
	kMeans	Kernel kMeans	ELM kMeans	kMeans	Kernel kMeans	ELM kMeans
2	91.5	68.5	**100.0**	0.776	0.276	**1.000**
3	96.6	79.4	**100.0**	0.942	0.729	**1.000**
4	93.6	67.8	**100.0**	0.907	0.756	**1.000**
5	71.0	64.7	**93.3**	0.834	0.755	**0.886**
6	64.3	65.4	**88.4**	0.769	0.738	**0.833**

Table 3
kMeans based clustering performance on *Libras Movement* data set.

Number of clusters	Purity (100%)			NMI		
	kMeans	Kernel kMeans	ELM kMeans	kMeans	Kernel kMeans	ELM kMeans
3	64.5	52.7	**65.3**	0.481	0.254	**0.500**
5	65.6	57.8	**69.8**	0.583	0.497	**0.626**
7	52.9	50.8	**55.0**	0.496	0.490	**0.535**
9	52.9	49.1	**54.1**	0.523	0.510	**0.558**
11	49.1	45.0	**50.7**	0.575	0.542	**0.599**
13	45.1	44.4	**47.2**	0.589	0.565	**0.614**
15	44.5	43.6	**45.6**	0.595	0.573	**0.605**

Table 4
kMeans based clustering performance on *TDT2* data set.

Number of clusters	Purity (100%)			NMI		
	kMeans	Kernel kMeans	ELM kMeans	kMeans	Kernel kMeans	ELM kMeans
10	89.0	70.4	**93.3**	0.916	0.838	**0.935**
15	85.8	70.6	**87.7**	0.920	0.857	**0.923**
20	85.0	72.1	**85.1**	0.922	0.877	**0.923**
22	81.8	69.0	**83.1**	0.910	0.863	**0.916**
24	**83.8**	69.9	83.2	**0.915**	0.868	0.914
26	82.4	69.1	**82.9**	0.915	0.870	**0.916**
28	82.7	68.8	**83.0**	0.913	0.861	**0.914**
30	82.2	73.6	**82.4**	0.907	0.877	**0.911**

clustering algorithm is superior to the other two algorithms with all the different cluster numbers. On *Libras Movement* and *TDT2* data sets, most of the time, NMF ELM clustering algorithm can get better results than the other two algorithms.

Over all, we can see that ELM based methods, i.e., ELM kMeans and ELM NMF clustering, is better than the kernel kMeans and kernel NMF methods on all the three data sets, respectively. It must be noted that our ELM based methods, which only need to add a ELM feature mapping process, are very convenient both for implementation and computation. Kernel methods need to use kernel functions because the feature mapping is always implicit.

Table 5
NMF based clustering performance on *Synthetic Control* data set.

Cluster number	Purity (100%)			NMI		
	NMF	Kernel NMF	ELM NMF	NMF	Kernel NMF	ELM NMF
2	73.6	88.9	**100.0**	0.326	0.618	**1.000**
3	72.2	66.1	**100.0**	0.780	0.708	**1.000**
4	66.2	77.1	**100.0**	0.799	0.782	**1.000**
5	68.8	69.3	**90.5**	0.799	0.753	**0.886**
6	55.3	63.8	**83.5**	0.702	0.691	**0.835**

Table 6
NMF based clustering performance on *Libras Movement* data set.

Cluster number	Purity (100%)			NMI		
	NMF	Kernel NMF	ELM NMF	NMF	Kernel NMF	ELM NMF
3	56.7	57.3	**64.4**	0.255	0.368	**0.440**
5	63.6	**67.4**	64.6	**0.598**	0.589	0.579
7	**61.4**	60.1	59.9	**0.607**	0.588	0.596
9	56.9	**60.2**	58.4	0.576	0.597	**0.610**
11	51.8	52.5	**52.6**	0.598	0.618	**0.625**
13	48.9	48.3	**49.1**	0.596	0.612	**0.624**
15	47.4	46.1	**48.7**	0.596	0.608	**0.628**

Table 7
NMF based clustering performance on *TDT2* data set.

Cluster number	Purity (100%)			NMI		
	NMF	Kernel NMF	ELM NMF	NMF	Kernel NMF	ELM NMF
10	89.3	87.7	**94.5**	0.906	**0.918**	0.905
15	85.3	83.3	**91.0**	0.896	0.907	**0.908**
20	84.2	82.5	**88.9**	0.903	0.911	**0.918**
22	83.0	83.2	**86.9**	0.898	0.909	**0.911**
24	81.0	81.0	**87.3**	0.893	0.903	**0.904**
26	77.9	80.2	**84.0**	0.879	0.898	**0.899**
28	78.3	80.7	**84.1**	0.878	**0.900**	0.892
30	78.0	80.5	**83.0**	0.877	**0.896**	0.885

4.5. Parameter selection

For the clustering algorithms in ELM feature space, we choose the Gaussian function (Eq. (6)) as the hidden-layer nodes activation function. One of the advantages of ELM is that the parameters used in the mapping process (Eq. (5)), $(a_i, b_i)_{i=1}^{L}$, can be randomly generated according to any continuous probability distribution, and they need not be tuned. The only parameter that needs to be specified by the users is the number of the hidden-layer nodes. Seen from Eq. (7), it can be found that ELM can use a large enough number of nodes to get good performance in regression and classification. For clustering, the relationship between the result quality and the number of nodes may be the same, which we will describe later. For the Mercer kernel based methods, we use the Gaussian radial basis function as the kernel function, which is as follows:

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \tag{14}$$

The parameter σ in Gaussian radial basis function need to be specified by the users, which can be selected by experiments.

For the algorithms in ELM feature space, to test how the number of hidden-layer nodes affects the performance, we run the algorithm with a wide range of different nodes. Specifically, 29

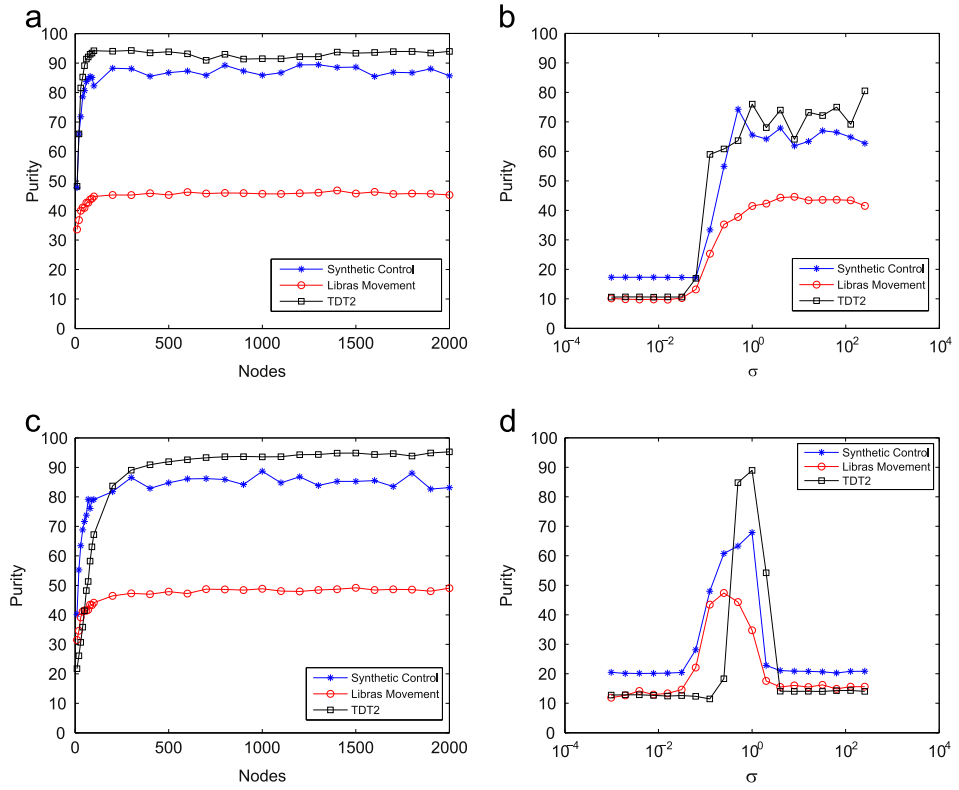


Fig. 2. Clustering performance under different parameters. (a) ELM kMeans algorithm. (b) Kernel kMeans algorithm. (c) ELM NMF clustering algorithm. (d) Kernel NMF clustering algorithm.

different values are used, they are {10,20,…,80,90,100,200,300, …,1900,2000}. For the Synthetic Control and Libras Movement data set, all the data are used to test the performance. As for the TDT2 data set, we choose the first 10 categories including 1000 documents. In the kernel based clustering algorithms, for the parameter σ, we choose 19 different values to test the performance, they are $\{2^{-10}, 2^{-9}, 2^{-8}, \ldots, 2^{7}, 2^{8}\}$. As the algorithm is stochastic, we run every algorithm 20 times and get the mean result.

The results of ELM kMeans algorithm under different number of nodes are shown in Fig. 2(a), it can be seen that, on all the three data sets, the performance of ELM kMeans algorithm is very stable as long as the number of hidden-layer nodes is large enough, that is, larger than 300. The results of kernel kMeans clustering algorithm with different σ values are shown in Fig. 2(b). For $\sigma > 1$, the performance is relatively stable, but σ cannot be extremely large, as the performance will deteriorate if it is too large. The results of ELM NMF algorithm under different number of nodes are shown in Fig. 2(c). Similarly, on all the three data sets, the performance of ELM NMF algorithm is very stable as long as the number of hidden-layer nodes is large enough, that is, larger than 300. The results of kernel NMF clustering algorithm with different σ values are shown in Fig. 2(d). It can be seen that its performance is very sensitive to the σ values, and different data sets require different σ to get good performance.

5. Conclusions

Inspired by the good results of clustering in Mercer kernel based feature space, this paper studies the clustering problem in ELM feature space. Two algorithms are presented, one is the ELM kMeans algorithm and the other is ELM NMF clustering, and both algorithms get better results compared to the Mercer kernel based methods on three benchmark data sets. Besides the good performance of ELM based algorithms, they are also very convenient for implementation and computation, which only require adding a ELM feature mapping process that is simpler than the kernel based feature mapping. Furthermore, seen from the three data sets used in this paper, ELM based methods require less human intervention as the algorithms performance is not sensitive to the number of hidden layer nodes, provided that a large enough number is selected (larger than 300 for the data in this paper). In this paper, two types of clustering algorithms using ELM feature mapping techniques are proposed, some other types of clustering may also be able to benefit from the ELM feature mapping techniques, which will be studied in the future.

Acknowledgments

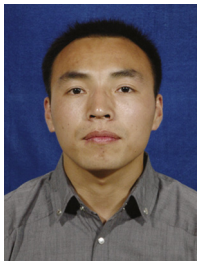
This work is supported by the National Natural Science Foundation of China (Nos. 61175052, 60975039, 61203297, 60933004, 61035003), National High-tech R&D Program of China (863 Program) (Nos. 2012AA011003, 2013AA01A606, 2014AA012205) and National Program on Key Basic Research Project (973 Program) (No. 2013CB329502).

References

[1] R. Sibson, Slink: an optimally efficient algorithm for the single-link cluster method, Comput. J. 16 (1) (1973) 30–34.
[2] D. Defays, An efficient algorithm for a complete link method, Comput. J. 20 (4) (1977) 364–366.
[3] S. Lloyd, Least squares quantization in PCM, IEEE Trans. Inf. Theory 28 (2) (1982) 129–137.
[4] T. Moon, The expectation-maximization algorithm, IEEE Signal Process. Mag. 13 (6) (1996) 47–60.
[5] M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the 2nd

International Conference on Knowledge Discovery and Data Mining, vol. 1996, AAAI Press, 1996, pp. 226–231.
[6] S. Roy, D. Bhattacharyya, An approach to find embedded clusters using density based techniques, in: Distributed Computing and Internet Technology, 2005, pp. 523–535.
[7] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, R. Harshman, Indexing by latent semantic analysis, J. Am. Soc. Inf. Sci. 41 (6) (1990) 391–407.
[8] U. Von Luxburg, A tutorial on spectral clustering, Stat. Comput. 17 (4) (2007) 395–416.
[9] W. Xu, X. Liu, Y. Gong, Document clustering based on non-negative matrix factorization, in: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2003, pp. 267–273.
[10] L. Zhang, W.-D. Zhou, L.-C. Jiao, Kernel clustering algorithm, Jisuanji Xuebao/Chin. J. Comput. 25 (6) (2002) 587–590.
[11] M. Girolami, Mercer kernel-based clustering in feature space, IEEE Trans. Neural Networks 13 (3) (2002) 780–784.
[12] F. Camastra, A. Verri, A novel kernel method for clustering, IEEE Trans. Pattern Anal. Mach. Intell. 27 (5) (2005) 801–805.
[13] G.-B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (2007) 3056–3062.
[14] G.-B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine, Neurocomputing 71 (2008) 3460–3468.
[15] Q. Liu, Q. He, Z. Shi, Extreme support vector machine classifier, in: T. Washio, E. Suzuki, K. Ting, A. Inokuchi (Eds.), Advances in Knowledge Discovery and Data Mining, Lecture Notes in Computer Science, vol. 5012, Springer, Berlin, Heidelberg, 2008, pp. 222–233.
[16] B. Frénay, M. Verleysen, Parameter-insensitive kernel in extreme learning for non-linear support vector regression, Neurocomputing 74 (16) (2011) 2526–2531.
[17] G. Huang, X. Ding, H. Zhou, Optimization method based extreme learning machine for classification, Neurocomputing 74 (1) (2010) 155–163.
[18] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of International Joint Conference on Neural Networks (IJCNN2004), vol. 2, Budapest, Hungary, 25–29 July 2004, pp. 985–990.
[19] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.
[20] D. Rumelhart, G. Hintont, R. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536.
[21] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 42 (April) (2012) 513–529.
[22] G.-B. Huang, D. Wang, Y. Lan, Extreme learning machines: a survey, Int. J. Mach. Learn. Cybern. 2 (2011) 107–122.
[23] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Networks 17 (4) (2006) 879–892.
[24] G.-B. Huang, Q.-Y. Zhu, K.Z. Mao, C.-K. Siew, P. Saratchandran, N. Sundararajan, Can threshold networks be trained directly? IEEE Trans. Circuits Syst. II 53 (3) (2006) 187–191.
[25] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Trans. Neural Networks 17 (November) (2006) 1411–1423.
[26] H.-J. Rong, G.-B. Huang, N. Sundararajan, P. Saratchandran, Online sequential fuzzy extreme learning machine for function approximation and classification problems, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 39 (4) (2009) 1067–1072.
[27] D. Zhang, W. Liu, An efficient nonnegative matrix factorization approach in flexible kernel space, in: Twenty-First International Joint Conference on Artificial Intelligence, 2009.
[28] D. Lee, H. Seung, et al., Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788–791.
[29] M. Berry, M. Browne, et al., Algorithms and applications for approximate nonnegative matrix factorization, Comput. Stat. Data Anal. 52 (1) (2007) 155–173.
[30] I. Buciu, N. Nikolaidis, I. Pitas, Nonnegative matrix factorization in polynomial feature space, IEEE Trans. Neural Networks 19 (6) (2008) 1090–1100.
[31] A. Asuncion, D. Newman, UCI Machine Learning Repository, Available at: ⟨http://archive.ics.uci.edu/ml/⟩, 2007.
[32] Q.-Y. Zhu, G.-B. Huang, Matlab Codes of ELM Algorithm, Available at: ⟨http://www.ntu.edu.sg/home/egbhuang/ELM_Codes.htm⟩, 2012.
[33] C. Cieri, S. Strassel, D. Graff, N. Martey, K. Rennert, M. Liberman, Corpora for topic detection and tracking, in: Topic Detection and Tracking, Kluwer Academic Publishers, Norwell, MA, USA, 2002, pp. 33–66.
[34] S. Wold, K. Esbensen, P. Geladi, Principal component analysis, Chemometrics Intell. Lab. Syst. 2 (1) (1987) 37–52.
[35] H. Xiong, J. Wu, J. Chen, K-means clustering versus validation measures: a data-distribution perspective, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 39 (April) (2009) 318–331.
[36] D. Seung, L. Lee, Algorithms for non-negative matrix factorization, Adv. Neural Inf. Process. Syst. 13 (2001) 556–562.

Qing He is a Professor at the Institute of Computing Technology, Chinese Academy of Sciences (CAS), and he is a Professor of Graduate University of Chinese Academy of Sciences (GUCAS). He received the B.S. degree from Hebei Normal University, Shijiazhuang, PR China, in 1985, and the M.S. degree from Zhengzhou University, Zhengzhou, PR China, in 1987, both in Mathematics. He received the Ph.D. degree in 2000 from Beijing Normal University in Fuzzy Mathematics and Artificial Intelligence, Beijing, PR China. From 1987 to 1997, he has been with Hebei University of Science and Technology. He is currently a doctoral tutor at the Institute of Computing Technology, CAS. His research interests include data mining, machine learning, cloud computing and big data.



Xin Jin is a Ph.D. candidate student in the Institute of Computing Technology, Chinese Academy of Sciences. He received the B.Eng degree and M.Eng degree in Computer Science from Shandong University of Science and Technology, Qingdao, China, in 2008 and 2011, respectively. His research interests include machine learning, data mining, distributed classification and clustering. He has been awarded in several scholarly contests, such as Mathematical Contest in Modeling of America, National Graduate's Mathematical Modeling Contest of China. Also, he has published several papers in relevant forums, such as Applied Mathematics and Computation.

Changying Du is a Ph.D. student in the Institute of Computing Technology, Chinese Academy of Sciences. He received the B.S. degree from the Department of Mathematics, Central South University, Changsha, China, in 2008. His research interests include machine learning, data mining, matrix factorization, nonparametric Bayesian statistics, multi-task and transfer learning, parallel and distributed algorithms, natural language processing and information retrieval. He has published several papers in relevant forums, such as Neurocomputing and IEEE ICDM.

Fuzhen Zhuang is an Assistant Professor in the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include transfer learning, machine learning, data mining, distributed classification and clustering, natural language processing. He has published several papers in some prestigious refereed journals and conference proceedings, such as IEEE Transactions on Knowledge and Data Engineering, Information Sciences, Neurocomputing, ACM CIKM, ACM WSDM, IEEE ICDM, SIAM SDM.

Zhongzhi Shi is a Professor in the Institute of Computing Technology, CAS, leading the Research Group of Intelligent Science. His research interests include intelligence science, multi-agent systems, semantic web, machine learning and neural computing. He has won a 2nd-Grade National Award at Science and Technology Progress of China in 2002, two 2nd-Grade Awards at Science and Technology Progress of the Chinese Academy of Sciences in 1998 and 2001, respectively. He is a senior member of IEEE, member of AAAI and ACM, Chair for the WG 12.2 of IFIP. He serves as the Vice President for Chinese Association of Artificial Intelligence.