



Representation learning using deep random vector functional link networks for clustering

Minghui Hu^a, P.N. Suganthan^{a,b,*}

^a School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

^b KINDI Center for Computing Research, College of Engineering, Qatar University, Doha, Qatar

ARTICLE INFO

Article history:

Received 5 March 2021

Revised 12 February 2022

Accepted 24 April 2022

Available online 26 April 2022

Keywords:

Random vector functional link

Unsupervised learning

Consensus clustering

Manifold regularization

ABSTRACT

Random Vector Functional Link (RVFL) Networks have received a lot of attention due to the fast training speed as the non-iterative solution characteristic. Currently, the main research direction of RVFLs has supervised learning, including semi-supervised and multi-label. There are hardly any unsupervised research results for RVFLs. In this paper, we propose the unsupervised RVFL (usRVFL), and the unsupervised framework is generic that can be used with other RVFL variants, thus we extend it to an ensemble deep variant, unsupervised deep RVFL (usdRVFL). The unsupervised method is based on the manifold regularization while the deep variant is related to the consensus clustering method, which can increase the capability and diversity of RVFLs. Our unsupervised approaches also benefit from fast training speed, even the deep variant offers a very competitive computation efficiency. Empirical experiments on several benchmark datasets demonstrated the effectiveness of the proposed algorithms.

© 2022 Elsevier Ltd. All rights reserved.

1. Introduction

Neural networks have triggered a wave of deep learning in recent years. Deep neural networks are mainly composed of many hidden layers used for nonlinear mapping. In particular, a single-layer feed-forward neural network (SLFN) is a shallow version of a neural network. It has been proven that any compact subset can be approximated by a single-layer neural network with a suitably chosen number of nodes [1,2]. However, most neural networks are typically optimized by back-propagation-based algorithms. This method involves iteration and requires a large amount of training data, leading to local optima and slow convergence. In addition to conventional neural networks, randomized-based neural networks employ closed-form solutions to conquer such problems. As a random version of the functional link neural network, Random Vector Functional Link [3,4] (RVFL) inherits the advantages of randomized neural networks and achieves competitive performance in classification, forecasting and regression. It has also been demonstrated that RVFL is a universal approximator for continuous functions on compact domains as well as other SLFNs [5]. The weights and biases of the hidden neurons in RVFL are randomly generated within a suitable range and kept fixed. Only the output weights need to

be updated in the training phase. And the original features are directly propagated to the final layer via the functional link. As the squared loss is employed in the RVFL, the solution can be regarded as a regularized least squares, or ridge regression, which is a simple closed-form solution.

Although many researchers have recently made sufficient contributions to the application and theory of RVFL [6–8], as well as many variants and extensions of RVFL [9,10], they are mostly supervised learning tasks, and there are few results for RVFL in the unsupervised domain. In practice, having enough well-labeled data is rare, and the difficulty of using unlabeled data can significantly reduce data efficiency. Therefore, the exploration of unsupervised tasks is necessary, and clustering is a major topic in unsupervised learning.

To fill the gap, we proposed using unsupervised learning methods to handle clustering tasks. We also extend it to ensemble deep variants based on consensus clustering methods, which can increase the diversity of the model. The unsupervised RVFL has a similar structure to the naive RVFL in that both solve for the output weights after mapping the inputs randomly, the only difference being how to solve for the output weights. The random mapping of input patterns plays a key role in RVFL, which allows the network to avoid overfitting problems and leads to fast non-linear mapping.

In the deep variants, we follow the description in [11], stacking multiple hidden layers in which weights and biases are randomly generated and kept fixed. The only part that needs to be updated is the weights of the output between the final layer and the multi-

* Corresponding author at: School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

E-mail address: epnsugan@ntu.edu.sg (P.N. Suganthan).

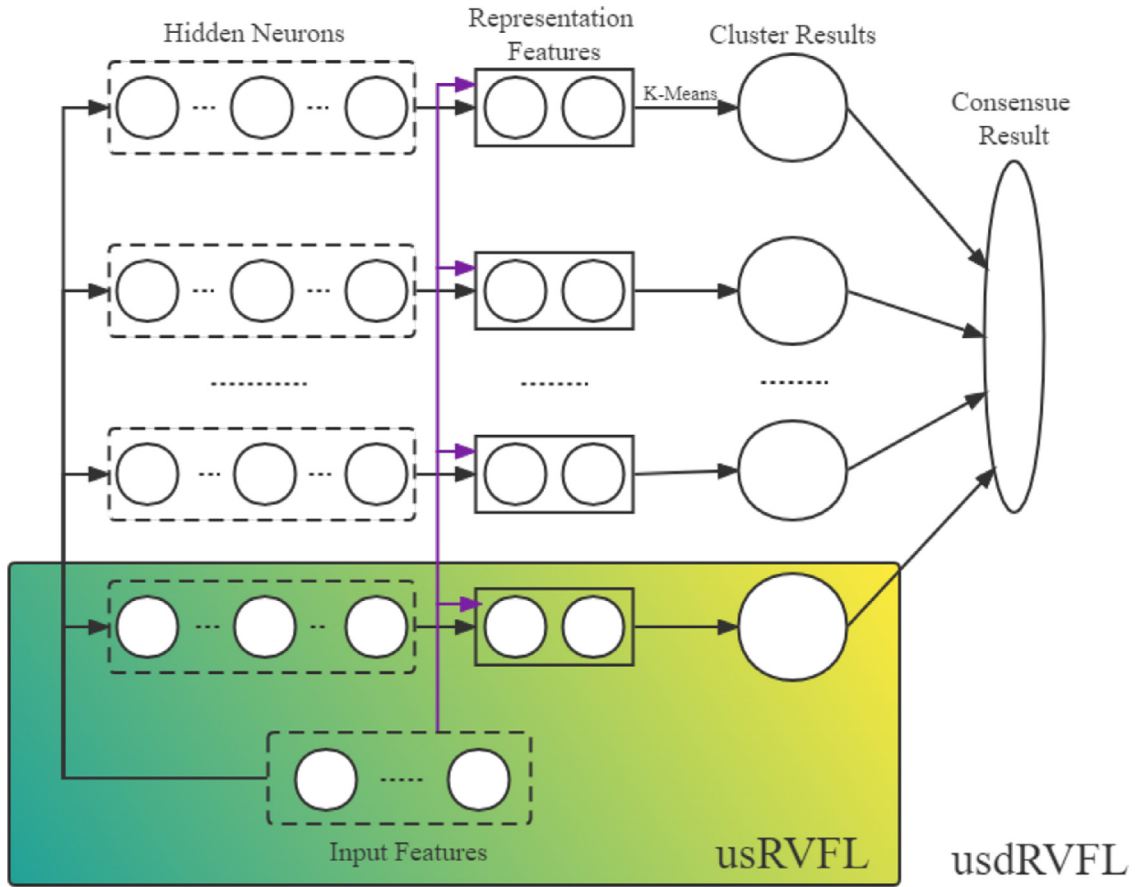


Fig. 1. The pipeline of proposed usRVFL and usdRVFL for clustering.

ple hidden layers. After obtaining the representation features from each layer, we clustered them one by one and ensemble them together via relabeling and voting strategy. The whole pipeline of our work is summarized in Fig. 1.

The content of this paper is summarized as follows: (i) We proposed an unsupervised solution for RVFL, the unsupervised method is based on manifold regularization. (ii) We extend the unsupervised RVFL to the ensemble deep version, which is inspired by the ensemble deep RVFL. (iii) We manifested that the performance of our unsupervised methods is robust and competitive on clustering tasks with extensive experiments on several real-world datasets.

2. Related work

In this section, we first demonstrate a brief overview of the fundamental spectral techniques and some related methods. We then review the base RVFL network and its deeper variants, and next we present some common approaches to consensus clustering.

2.1. Manifold learning and spectral techniques

The spectral Technique is an algorithm that evolved from graph theory and has since found widespread use in clustering. The essence of spectral clustering is manifold learning. Its main idea is to regard all the data as nodes in space, which can be connected to each other by edges. The edge weights between two nodes that are further apart have low values, while the edge weights between two closer nodes have higher values. The purpose of spectral clustering is to be achieved by cutting the graph composed of all data

nodes so that the edge weights between different subgraphs after the cut are as low as possible, while the edge weights within the subgraphs are as high as possible. Laplacian eigenmaps [12] is used for dimension reduction and it also plays an important role in manifold learning.

Spectral clustering (SC) [13] simply adds a clustering step to Laplacian eigenmaps. The Laplacian eigenmaps method requires the construction of a similarity graph G , followed by the use of an adjacency matrix W to determine the weights. When making the adjacency matrix, the weights of each edge need to be made with a Gaussian kernel function. The Laplacian matrix L can then be calculated, and the first m eigenvectors can be used for embedding. Meanwhile, spectral clustering conducts k means method after the Laplacian eigenmap embedding process.

Projected Clustering with Adaptive Neighbors (PCAN) [14] learns the similarity matrix G and the label matrix F simultaneously by introducing the probabilistic neighbor, where F is the final cluster output. The neighbor similarity matrix can be regarded as the adjacency matrix in SC.

The Constrained Laplacian Rank Algorithm for Graph-Based Clustering (CLR) [15] also learns directly the data similarity graph G with k connected components, where k is the number of clusters. The author imposed a Laplace rank constraint on the similarity graph to guarantee the existence of k connected components.

It's also based on the Laplacian matrix, and the solution of RVFL output weights can be turned into an eigenvalue problem with the same output weights and eigenvectors as in SC. The major difference between SC and usRVFL is that the eigenvectors are the weights of the output, which are used for obtaining the embedding features instead of being used for clustering directly.

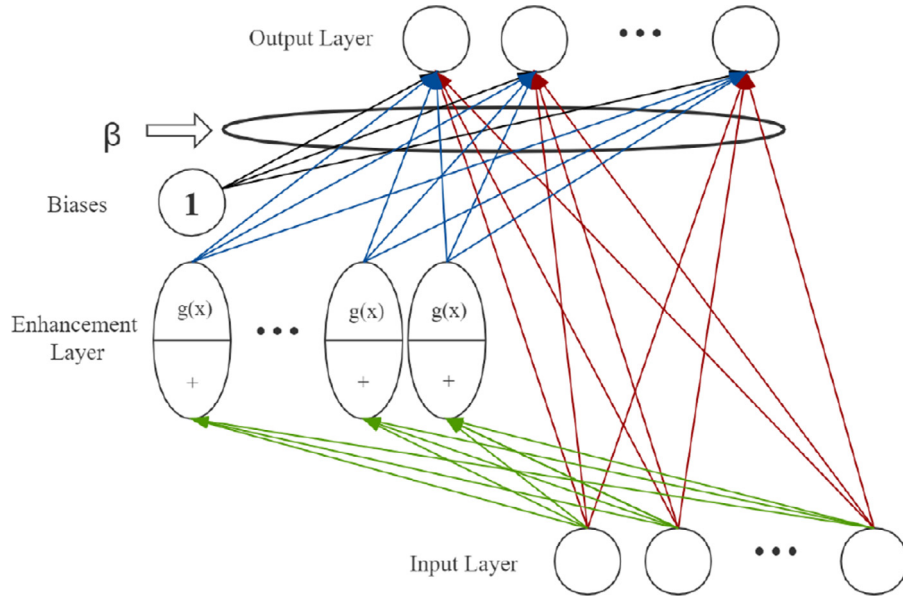


Fig. 2. The structure of RVFL network. Randomized links are the green lines between the input layer and the enhancement layer, direct links are in red between the input layer and output layer.

2.2. Random vector functional link

Random Vector Functional Link (RVFL) was proposed by Pao and Takefuji in [16]. The inputs to the final layer consist of the raw input patterns $\mathbf{X} \in \mathbb{R}^{k \times d}$, the corresponding label $\mathbf{Y}$ and the features after nonlinear transformation $\mathbf{A} : \mathbb{R}^{k \times d} \rightarrow \mathbb{R}^{k \times N}$, where k is the number of samples, d and N are the numbers of original input features and hidden neurons. The structure of RVFL is given in Fig. 2. The parameters for the hidden neurons are generated randomly and fixed. Only the weights of the output neurons β are to be optimized during the training phase by the following equation:

$$\min_{\beta} \|\mathbf{E}\beta - \mathbf{Y}\|^2 + \lambda \|\beta\|^2 \quad (1)$$

where $\mathbf{E} \in \mathbb{R}^{m \times n_e}$ is the linear combination of input patterns $\mathbf{X}$ and hidden features $\mathbf{A}$, which we termed enhanced features. And the hidden features $\mathbf{A}$ can be calculated by $g(a_j^T x + b_j)$, where $g(\cdot)$ is the activation function for non-linear mappings. Thus the objective function can be solved by Moore-Penrose pseudo-inverse:

$$\beta = \mathbf{E}^+ \mathbf{Y} \quad (2)$$

or ridge regression, which can be expressed by

$$\beta = (\mathbf{E}^T \mathbf{E} + \lambda \mathbf{I})^{-1} \mathbf{E}^T \mathbf{Y} \quad (3)$$

where $\mathbf{I}$ is the identity matrix and λ is the regularization parameter. The hidden weights and biases mentioned in this work are randomly generated from a uniform distribution $U \in [-1, 1]$ before scale.

Shi et al. [11] proposed deep variants of RVFL, one of the deep variants called ensemble deep RVFL (edRVFL), which adopts the strategy of the ensemble. The edRVFL network has multiple hidden layers hierarchically stacked. Input to each hidden layer in the stack is the output of the prior layer. The architecture is shown in Fig. 3, the input to the hidden layers $\mathbf{A}^{(i)}$ is defined as follows:

$$\mathbf{A}^{(i)} = \begin{cases} g(\mathbf{X}\mathbf{W}_r^{(1)}) & \text{if } i = 1 \\ g([\mathbf{A}^{(i-1)}; \mathbf{X}]\mathbf{W}_{re}^{(i)}) & \text{if } i > 1 \end{cases} \quad (4)$$

where $\mathbf{W}_{re}^{(i)} \in \mathbb{R}^{(d+N) \times N}$. The outputs weight β_{ed} can be solved by Eqs. (2) or (3). For the ease of notation, we omit the bias term in

the formulas, as shown in [3,17], compared with no bias, the network with bias has more freedom, it will have a better approximation to the data.

The RVFL network is demonstrated to be a universal approximator for a continuous function on a bounded finite-dimensional set. Its own and its variants perform well on supervised classification tasks in a variety of situations [11,18].

2.3. Randomized network based unsupervised learning

The Extreme Learning Machine (ELM) is an essential branch of randomized neural networks, which is also a single-layer feed-forward network (SLFN) whose neurons between the input and hidden layer are randomly generated and fixed, but without directly connected layers compared to RVFL [19]. In addition to the classification task, there are some later works that extend the application of ELM to unsupervised learning [20–22]. In unsupervised ELM (US-ELM) [20], the author employed US-ELM for feature embedding and using k -means for the clustering. Liu et al. [21] also used US-ELM for feature representation and clustering.

2.4. Consensus clustering

Consensus clustering is also known as ensemble clustering, the process is shown in Fig. 4. The reason why ensemble methods are employed is to combine some weak clusters into a strong one [23,24]. To create diverse clusters, several approaches can be applied: using different sampling data, using different subsets of the original features, and employing different clustering methods [25]. After receiving all the outputs from each cluster, various consensus functions can be chosen to obtain the final output based on the user's requirements [26].

There are many mathematical and computational tools to solve the consensus problems, which can be divided into two main parts: object co-occurrence and median partition. Objects Co-occurrence methods are more common, which include Re-labelling & voting approach [27], Information Theory approach [28], Hypergraph methods [29], etc.

Among them, Re-labelling & voting approach is a straightforward and effective method [30]. The first issue to be solved is that of corresponding label problems. Since the labels associated with

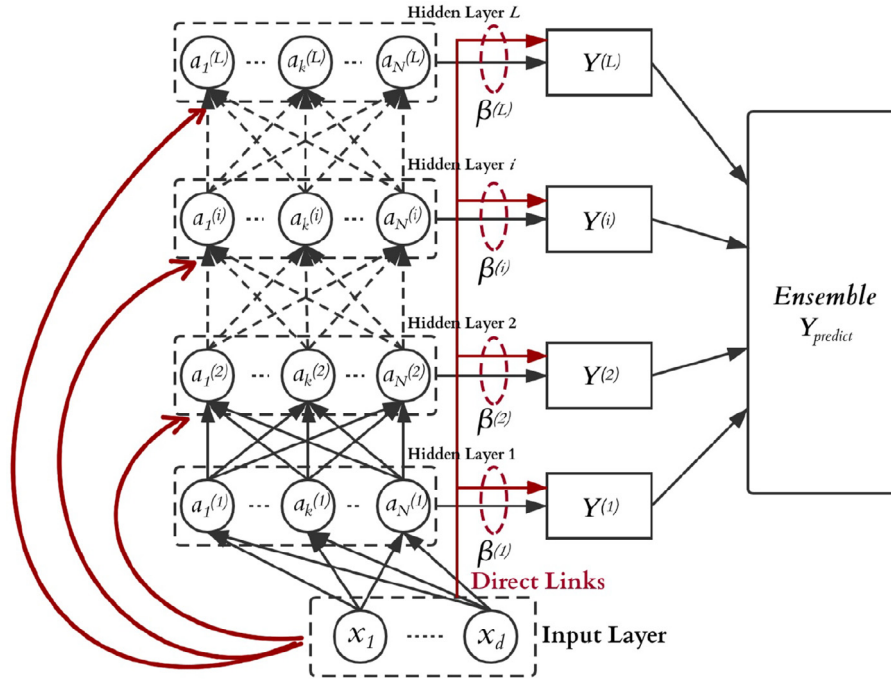


Fig. 3. The structure of edRVFL network. Compared to dRVFL, the β_d is separated into several β_{ed} . Each β_{ed} is calculated for an individual RVFL output layer. The input features concatenate with outputs of each hidden layer by direct links shown as red lines. The final output is obtained by aggregating predictions of all output layers.

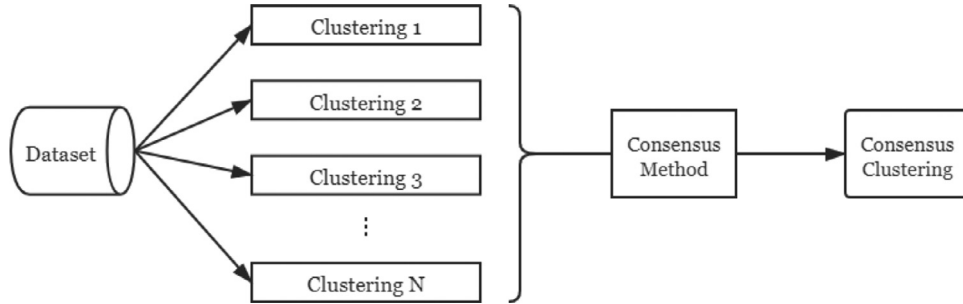


Fig. 4. The process of consensus clustering.

each object in the clustering are symbolic and there is no relationship between the sets of labels given by the clustering algorithm, we convert the relabelling problem to a bipartite graph matching problem and then use the Kuhn-Munkres (KM) algorithm to solve it. The KM algorithm transforms the problem of finding a maximum weight match into a complete match by giving each vertex a label, and it can solve the relabel task in strongly polynomial time. The winner is then obtained using a plurality voting method as the final clustering prediction. Details of the method we use for relabeling and voting are described in [Section 4.2](#)

3. Unsupervised RVFL for representation learning

In unsupervised learning, the objective function of RVFL (Eq. (1)) should be free of the labels Y . In order to obtain the representation results, we use manifold regularization to enable RVFL to obtain a non-linear mapping of the original data. Specifically, the normalized Laplacian as a regularizer is used to represent the data in the manifold space as shown in [Fig. 5](#). The following sections show the steps of manifold regularization in detail.

3.1. Manifold regularization

In the unsupervised case, given a set of data points without labels $\mathbf{X}_u = \{x_1 \dots x_k\}$, where the dimension of x_i is n_r , and all un-

labelled examples can be sampled from the same marginal distribution P_X . If two points x_1 and x_2 are similar, they will be close to the intrinsic geometry of P_X . We also want to include geometric information about P_X , thus an additional regularization term is added to smooth the ambient space and the marginal distribution P_X . Therefore, we can have an objective function with an extrinsic regularization term and an intrinsic regularization term, which can be mathematically expressed as:

$$\min_{\mathbf{f} \in \mathbb{H}_K} \lambda \|\mathbf{f}\|_K^2 + \|\mathbf{f}\|_f^2 \quad (5)$$

For a Mercer kernel $K : X \times X \rightarrow \mathbb{R}$, there is an corresponding norm $\|\cdot\|_K$ in the associated Reproducing Kernel Hilbert Space (RKHS) $\mathbb{H}_K$. Here $\mathbf{f}$ is the function in the hypothesis space $\mathbb{H}_K$ and $\|\mathbf{f}\|_K^2$ is the penalty term. $\|\mathbf{f}\|_f^2$ is an appropriate penalty term that should reflect the intrinsic structure [31].

Spontaneously, the intrinsic regularization term $\|\mathbf{f}\|_f^2$ can be convert to

$$\int_{x \in M} \|\nabla_M f\|^2 dP_X(x) \quad (6)$$

where ∇_M means the gradient and $M \subset \mathbb{R}$ is a compact submanifold of P_X support. There are different manifold smoothing regularizers, such as Frobenius norm of the Hessian [32], Diffusion regularizers [33] and Iterated Laplacians [34]. Here we can use graph Laplacian approximating the regularization term [Eq. \(6\)](#). Thus the

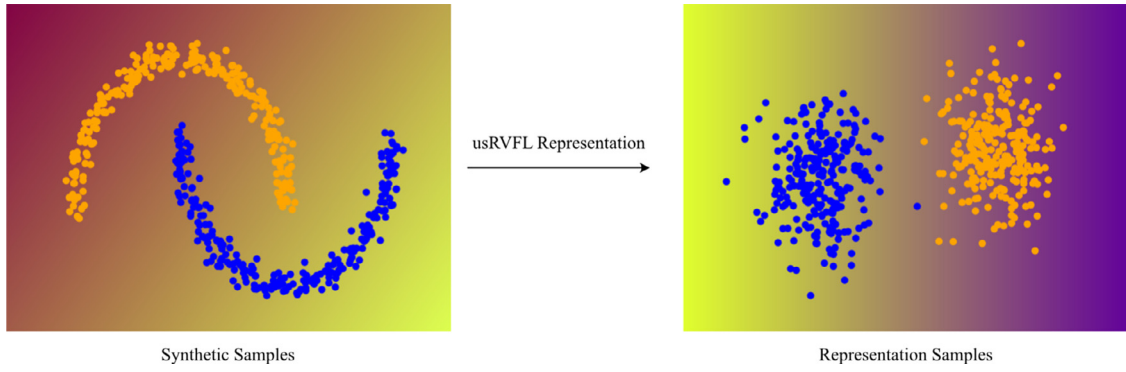


Fig. 5. The geometric intuition of usRVFL. The primary justification for mapping to high dimensional is make the difficult non-linear separation to become linear or easy separation. Gradient background colours represent transitions between classification areas.

objective function is:

$$\begin{aligned} L &= \min_{\mathbf{f} \in \mathbb{R}} \lambda \|\mathbf{f}\|_K^2 + \int_{x \in M} \|\nabla_M f\|^2 dP_X(x) \\ &= \min_{\mathbf{f} \in \mathbb{R}} \lambda \|\mathbf{f}\|_K^2 + \sum_i j W_{ij} \|f(x_i) - f(x_j)\|^2 \\ &= \min_{\mathbf{f} \in \mathbb{R}} \lambda \|\mathbf{f}\|_K^2 + \text{Tr}(\mathbf{f}^T \mathbf{L} \mathbf{f}) \end{aligned} \quad (7)$$

where L is the graph Laplacian and is obtained from $\mathbf{D} - \mathbf{W}$. Here $\mathbf{D}$ is the diagonal matrix and given by $D_{ii} = \sum_{j=1} W_{ij}$, and $\mathbf{W}$ is the similarity matrix, which represents the edge weight in adjacency graph. Normally, the similarity matrix is sparse consisting of 0,1. For each x_i , we find the n nearest neighbors and regard them as connected. $W_{ij} = 1$ if vertices i and j are connected by an edge and $W_{ij} = 0$ if vertices i and j are not connected. And $\text{Tr}(\cdot)$ means the trace of the matrix.

It is also worth noting that the graph Laplacian can be replaced by the normalized Laplacian

$$L_{\text{norm}} = \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{1/2} \quad (8)$$

Using normalized Laplacian instead of graph Laplacian can make the convergence easier and perform better in many practical tasks [35]. Furthermore, there is some experimental evidence demonstrated the regularized graph Laplacian L_{norm} yields better empirical results [36]. Therefore, the remaining L in this paper represents the normalized graph Laplacian.

3.2. Unsupervised random vector functional link network

Since all the data X are unlabelled, the first term in the original RVFL solution Eq. (1) can be removed and the remainder can be modified according to the manifold regularization mentioned in Eq. (8) as follows:

$$\begin{aligned} \min_{\beta \in \mathbb{R}^{n_e \times n_o}} \quad & \lambda \|\beta\|^2 + \text{Tr}(\mathbf{f}^T \mathbf{L} \mathbf{f}) \\ \text{s.t.} \quad & f_i = e_i \beta \quad e_i \in \mathbf{E} \end{aligned} \quad (9)$$

where β follows the notion in Eq. (1), is the output weights to be learnt, while n_e and n_o are the dimensions of the enhanced feature and the embedding output, respectively. We can express it in matrix form

$$\min_{\beta \in \mathbb{R}^{n_e \times n_o}} \lambda \|\beta\|^2 + \text{Tr}(\beta^T \mathbf{E}^T \mathbf{L} \mathbf{E} \beta) \quad (10)$$

To conquer the degenerated problem that $\mathbf{f}$ going arbitrary large, we follow the method proposed by Belkin and Niyogi [37], which impose a constraint $\beta^T \mathbf{E}^T \mathbf{L} \mathbf{E} \beta = I$. Therefore, our objection function can be expressed as

$$\min_{\beta \in \mathbb{R}^{n_e \times n_o}, \beta^T \mathbf{E}^T \mathbf{L} \mathbf{E} \beta = I} \text{Tr}(\beta^T \mathbf{E}^T \mathbf{L} \mathbf{E} \beta) \quad (11)$$

According to Ghogh et al. [38], the eigenvalue optimization for the trace of the matrix $\beta^T \mathbf{E}^T \mathbf{L} \mathbf{E} \beta$ can be regarded as a generalized

eigenvalue problem:

$$\mathbf{L} \mathbf{E} \beta = \gamma \mathbf{D} \mathbf{E} \beta \quad (12)$$

According to the Croot [39], a common method to obtain the optimized solution by the Rayleigh principle, which is stacking the n_o eigenvectors corresponding to the corresponding n_o lowest eigenvalues of the generalized eigenvalue problem solution Eq. (12).

In our solution, we rewrite Eq. (11) as:

$$\min_{\beta \in \mathbb{R}^{n_e \times n_o}, \beta^T \Phi \beta = I} \text{Tr}(\beta^T \Lambda \beta) \quad (13)$$

where $\Lambda = \lambda \mathbf{I} + \mathbf{E}^T \mathbf{L} \mathbf{E}$ and $\Phi = \mathbf{E}^T \mathbf{D} \mathbf{E}$. And the generalized eigenvalue problem solution is

$$\Lambda \mathbf{v} = \gamma \Phi \mathbf{v} \quad (14)$$

$$(\mathbf{I} + \lambda \mathbf{E}^T \mathbf{L} \mathbf{E}) \mathbf{v} = \gamma \mathbf{E}^T \mathbf{D} \mathbf{E} \mathbf{v} \quad (15)$$

We can also solve such trace minimization problem by the Rayleigh theorem due to Λ and Φ are also self-adjoint, which is the as the solution of Eq. (12). And it is obvious that $\mathbf{1}$ is an eigenvector corresponding to eigenvalue 0, in order to eliminate this trivial solution, the eigenvector $\mathbf{1}$ is discarded and the optimal solution is stacking up the following n_o eigenvectors.

Refer back to Eq. (14), let $\mathbf{v}_0, \dots, \mathbf{v}_{n_o}$ are the eigenvectors, corresponding to their eigenvalues $\gamma_0, \dots, \gamma_{n_o}$

$$\begin{aligned} \Lambda \mathbf{v}_0 &= \gamma_0 \Phi \mathbf{v}_0 \\ \Lambda \mathbf{v}_1 &= \gamma_1 \Phi \mathbf{v}_1 \\ &\dots \end{aligned} \quad (16)$$

$$\begin{aligned} \Lambda \mathbf{v}_{n_o+1} &= \gamma_{n_o} \Phi \mathbf{v}_{n_o} \\ 0 &= \gamma_0 \leq \gamma_1 \leq \dots \leq \gamma_{n_o} \end{aligned}$$

The final solution to the embedding weights β is the stacking of n_o smallest normalized eigenvectors without $\mathbf{v}_0$.

$$\beta = [\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \dots, \tilde{\mathbf{v}}_{n_o}] \quad (17)$$

where $\tilde{\mathbf{v}}_i = \mathbf{v}_i / \|\mathbf{E} \mathbf{v}_i\|$, $i = 1, 2, \dots, n_o$.

If the number of observations is smaller than the dimension of enhanced features $\mathbf{E}$, we can transfer Eq. (14) from primal space into dual space by defining $\Lambda = \mathbf{I} + \lambda \mathbf{L} \mathbf{E} \mathbf{E}^T$ and $\Phi = \mathbf{E} \mathbf{E}^T$, thus the equation can be shown as

$$(\mathbf{I} + \lambda \mathbf{L} \mathbf{E} \mathbf{E}^T) \mathbf{v} = \gamma \mathbf{E} \mathbf{E}^T \mathbf{v} \quad (18)$$

and the optimize solution is

$$\beta = \mathbf{E}^T [\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \dots, \tilde{\mathbf{v}}_{n_o}] \quad (19)$$

where $\tilde{\mathbf{v}}_i = \mathbf{v}_i / \|\mathbf{E} \mathbf{E}^T \mathbf{v}_i\|$, $i = 1, 2, \dots, n_o$.

The final output $\mathbf{O}$ of unsupervised RVFL can be straightforward calculated by

$$\mathbf{O} = \mathbf{E} \beta \quad (20)$$

where $\mathbf{O} \in \mathbb{R}^{k \times n_o}$ is the embedded vector with the size n_o , and the whole process is shown in Algorithm 1.

Algorithm 1: Unsupervised RVFL for representation learning.

Input: Dataset $\mathbf{X}_u \in \mathbb{R}^{k \times n_r}$
Output: Embedded vector $\mathbf{O} \in \mathbb{R}^{k \times n_o}$
 Compute the similarity matrix $\mathbf{W}$;
 Construct the normalized graph Laplacian $\mathbf{L}$;
 Initialize the usRVFL network, calculate the outputs $\mathbf{A}$ from hidden layer, and concatenate it with the original input feature $\mathbf{X}$ to get the enhanced feature $\mathbf{E} \in \mathbb{R}^{k \times n_e}$;
if $n_e \leq k$ **then**
 Find the n_o smallest eigenvalues γ and corresponding eigenvectors $\mathbf{v}$ by Eq.-(15);
 Compute the normalized eigenvectors $\tilde{\mathbf{v}}_i = \mathbf{v}_i / \|\mathbf{E}\mathbf{v}_i\|$;
 Compute $\beta = [\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \dots, \tilde{\mathbf{v}}_{n_o}]$;
else
 Find the n_o smallest eigenvalues γ and corresponding eigenvectors $\mathbf{v}$ by Eq.-(18);
 Compute the normalized eigenvectors $\tilde{\mathbf{v}}_i = \mathbf{v}_i / \|\mathbf{E}\mathbf{E}^T \mathbf{v}_i\|$;
 Compute $\beta = \mathbf{E}^T [\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \dots, \tilde{\mathbf{v}}_{n_o}]$;
end
 Compute embedded vectors $\mathbf{O} = \mathbf{E}\beta$;
return Embedded vector $\mathbf{O}$

4. Unsupervised deep RVFL for clustering

Above, we described the approach for the unsupervised RVFL (usRVFL) representation task. After we obtain the embedding vector $\mathbf{O}$ from usRVFL, we can perform clustering of these vectors in embedding space using some convention clustering algorithms. The clustering task requires the similarity of a given number of data points to be evaluated, and the clustering algorithm is also highly dependent on the data points involved. We also extend the usRVFL framework for clustering to a deep version, which is called usdRVFL. It employs the consensus clustering strategy based on the usRVFL.

4.1. Unsupervised RVFL for clustering

In our clustering task, we use the k -means technique to cluster the embedded vectors, as shown in Algorithm 2. And we initialize the centroids by the k -means++ [40], which basic principle is to make the initial clustering centers as far away from each other as possible.

4.2. Unsupervised deep RVFL for clustering

The proposed unsupervised framework is generic and can be adopted on the ensemble deep RVFL network (edRVFL). We treat each hidden layer in the unsupervised deep RVFL (usdRVFL) and its corresponding output layer as a separate usRVFL representation operation. Therefore, for an RVFL with l hidden layers, there will be l sets of embedding vectors. And we do k -means clustering for each set of these embedded vectors. Instead of training several networks or using different clustering methods independently, we use one usdRVFL network to obtain multiple clustering results at once.

After obtaining l clustering results, each of which is a weak cluster, a more precise clustering can then be made by means of consensus clustering. To ensemble the above clustering results, we employ the relabelling and voting strategy. The details are given in Algorithm 3. We can construct $l - 1$ weighted bipartite graphs $\mathbf{B}_i = \{\hat{\mathbf{y}}_i, \mathbf{y}_1\}$ between the first layer predictions, or called criteria

Algorithm 2: Unsupervised RVFL for clustering.

Data: Dataset $\mathbf{X}_u \in \mathbb{R}^{k \times n_r}$
Input: Number of clusters $\mathbf{K}$
Output: List of cluster index $\hat{\mathbf{y}} \in \mathbb{N}^{k \times 1}$
 Compute the similarity matrix $\mathbf{W}$;
 Construct the normalized graph Laplacian $\mathbf{L}$;
 Initialize the usRVFL network, calculate the outputs $\mathbf{A}$ from hidden layer, and concatenate it with the original input feature $\mathbf{X}$ to get the enhanced feature $\mathbf{E} \in \mathbb{R}^{k \times n_e}$;
if $n_e \leq k$ **then**
 Find the n_o smallest eigenvalues γ and corresponding eigenvectors $\mathbf{v}$ by Eq.-(15);
 Compute the normalized eigenvectors $\tilde{\mathbf{v}}_i = \mathbf{v}_i / \|\mathbf{E}\mathbf{v}_i\|$;
 Compute $\beta = [\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \dots, \tilde{\mathbf{v}}_{n_o}]$;
else
 Find the n_o smallest eigenvalues γ and corresponding eigenvectors $\mathbf{v}$ by Eq.-(18);
 Compute the normalized eigenvectors $\tilde{\mathbf{v}}_i = \mathbf{v}_i / \|\mathbf{E}\mathbf{E}^T \mathbf{v}_i\|$;
 Compute $\beta = \mathbf{E}^T [\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \dots, \tilde{\mathbf{v}}_{n_o}]$;
end
 Compute embedded vectors $\mathbf{O} = \mathbf{E}\beta$;
 Initialize $\mathbf{K}$ centroids;
while the centroids are changing **do**
 Assign each embedded vector to the closest centroid;
 Compute the new centroid for each cluster;
end
return List of cluster index $\hat{\mathbf{y}}$

Algorithm 3: Unsupervised deep RVFL for clustering.

Data: Dataset $\mathbf{X}_u \in \mathbb{R}^{k \times n_r}$
Input: Number of clusters $\mathbf{K}$; Number of hidden layers l
Output: List of cluster index $\hat{\mathbf{y}} \in \mathbb{N}^{k \times 1}$
 Compute the similarity matrix $\mathbf{W}$;
 Construct the normalized graph Laplacian $\mathbf{L}$;
 Initialize the usRVFL network, $m \leftarrow 0$;
for $m < l$ **do**
 Calculate the outputs $\mathbf{A}_m$ from the m th hidden layer, and concatenate it with the original input feature $\mathbf{X}$ to get the enhanced feature $\mathbf{E}_m \in \mathbb{R}^{k \times n_e}$;
 if $n_e \leq k$ **then**
 Find the n_o smallest eigenvalues γ and corresponding eigenvectors $\mathbf{v}$ by Eq.-(15);
 Compute the normalized eigenvectors $\tilde{\mathbf{v}}_i = \mathbf{v}_i / \|\mathbf{E}_m \mathbf{v}_i\|$;
 Compute $\beta = [\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \dots, \tilde{\mathbf{v}}_{n_o}]$;
 else
 Find the n_o smallest eigenvalues γ and corresponding eigenvectors $\mathbf{v}$ by Eq.-(18);
 Compute the normalized eigenvectors
 $\tilde{\mathbf{v}}_i = \mathbf{v}_i / \|\mathbf{E}_m \mathbf{E}_m^T \mathbf{v}_i\|$;
 Compute $\beta = \mathbf{E}_m^T [\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \dots, \tilde{\mathbf{v}}_{n_o}]$;
 end
 Compute embedded vectors $\mathbf{O} = \mathbf{E}_m \beta$;
 Cluster the embedded vectors and record the clustering index of the m th layer $\mathbf{y}_m$ in $\mathbb{Y}$;
 $m \leftarrow m + 1$;
end
 Relabel the clustering results $\mathbb{Y}$ to get the new predictions $\hat{\mathbb{Y}}$;
 Use the voting strategy to determine the winner $\hat{\mathbf{y}}$ among $\hat{\mathbb{Y}}$;
return List of cluster index $\hat{\mathbf{y}}$

layer, and the i th layer with the weight matrix $\mathbf{C}_i$, each value in the matrix $\mathbf{C}_i$ represents the number of overlaps between the i th layer predictions and the criteria predictions. Then we can use the Kuhn-Munkres (KM) algorithm [41] to find the maximum weight matching of each graph $\mathbf{B}_i$ and get the relabelled indices for each layer with the first layer outputs as the criteria. Then we make the final ensemble prediction based on a majority voting strategy.

5. Experiments

In this section, we evaluate the clustering performance of our proposed unsupervised RVFL(usRVFL) and its variant unsupervised deep RVFL on several benchmark datasets. We also include some comparisons between some other randomized network-based clustering methods and spectral clustering methods.

5.1. Comparison methods

To gauge the quality of clustering, externally validated approaches are used, whereby the quality of the clustering solution is evaluated based on the labels. The evaluation metric used in the experiments is clustering accuracy (ACC).

If $\hat{y}_i$ is the relabelled prediction index for i th patterns in clustering algorithms and y_i is the corresponding ground truth, ACC measures the fraction of patterns properly predicted versus the ground truth label in the clustering solution. Mathematically, the ACC is defined as:

$$\text{accuracy}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{k} \sum_{i=0}^{k-1} \mathbb{1}(\hat{y}_i = y_i) \quad (21)$$

where $\mathbb{1}$ is the indicator function, indicating which of these elements belong to the given subset.

To validate the effectiveness of the proposed unsupervised RVFL framework, we have conducted a comparison of related algorithms. The compared methods are enumerated as follows:

1. k -Means: The baseline clustering method
2. Unsupervised Extreme Learning Machine (US-ELM) [20]
3. Spectral Clustering (SC) [13]
4. Projected Clustering with Adaptive Neighbors (PCAN) [14]
5. Constrained Laplacian Rank (CLR) [15]
6. ELM based Constrained Laplacian Rank (ELM-CLR) [21]
7. Low Rank Representation (LRR) [42]
8. The Separation Capacity of Random Neural Networks (RM) [43]
9. Unsupervised Random Vector Functional Link (usRVFL)
10. Unsupervised deep Random Vector Functional Link (usdRVFL)

It is worth noting that No. 8 method is under the hypothesis of the dichotomous case, we will abuse it here in the case of multiple classifications. We follow the method in [43] to construct a random ReLU-layer $\mathbb{F}(\cdot)$ in our experiments:

$$\mathbb{F}(\mathbf{x}) = \sqrt{\frac{2}{n_o}} \text{ReLU}(\mathbf{W}\mathbf{x} + \mathbf{b}) \quad (22)$$

where weight matrix $\mathbf{W} \in \mathbb{R}^{n_e \times n_o}$ has standard Gaussian entries, and the bias $\mathbf{b}$ obeys a uniform distribution on $[-1, 1]$ independently. For multi-class tasks, we also follow this approach to construct random hidden layers that perform a random mapping of the initial features. The identical k means++ method is used to cluster the features after random mapping.

5.2. Experiment settings

In order to fairly compare different algorithms, the following experimental setting is used: The grid search method is used for parameter tuning and we selected the best results in tuning

phases; As the performance of k -means is sensitive to centroid initialization, we repeat the k -means 100 times and take the median results in the algorithms that comprise k -means, including SC, US-ELM, ALC Series and usRVFL Series; For randomized networks-based algorithms, such as US-ELM, ALC-ELM and usRVFL Series, we repeat the weight randomization for 10 times and obtain the means of the accuracy; All the data points are normalized in the range of $[-1, 1]$; We follow the experiments setting for ELM based algorithms described in [21]. The search range of hyper-parameters is given in Appendix.

We conducted the four-fold cross-validation on all datasets to find the optimal hyperparameters. To be specific, we will detach a portion of the training set as a validation dataset, the configuration that achieves the best accuracy on this validation set is selected as the optimal setting. We will then retrain the model with the complete training set and the optimal settings and then use this model for testing.

For randomized neural networks, we stop training when we have traversed all hyperparameters; for other methods, the stopping criteria is the maximum number of epochs

5.3. Performance comparison and analysis

The average accuracy of usRVFL/usdRVFL and other baseline algorithms is demonstrated in Table 1. From the table, we can conclude that usdRVFL proposed in the paper yields the best performance in 10 datasets out of 11, considering the criterion of accuracy. Besides, usdRVFL performs notably well on some low dimensional tabular datasets, such as AutoMPG(4.12%), SEGMENT(7.19%) and CONTROL(9.33%). However, in the ECOLI dataset, our method demonstrated similar accuracy with ELM-CLR and LRR, one possible reason is both methods are subspace clustering approaches and the representation in latent space plays an important role. For the Diabetes task, the random neural network achieved excellent results, this reveals that random mapping can tackle the task of binary classification as the recent work [43] shown.

We also perform the Wilcoxon Signed-rank Test for ranking. The pairwise statistical comparison of all mentioned algorithms is shown in Table 2. We can see proposed method is significantly better than any other algorithms in comparison.

In addition, usRVFL exhibits similar performance to US-ELM, as they both use manifold regularization to embed the input patterns and employ the same method for clustering embedded features. However, usRVFL outperforms US-ELM in general, possibly due to usRVFL holding a direct link that provides a better way to represent the patterns. Figs. 6 and 7 show a more detailed comparison between usRVFL and US-ELM in UCI datasets (low dimension) and image dataset (high dimension). Here we capture the minimum value of 100 repeats k -means experiments. We can see usRVFL offers better overall results for all benchmark datasets, and usRVFL out stands US-ELM regarding all embedding dimensions in several special datasets, like USPST and ETH80. We can assume that RVFL is superior when confronted with high-dimensional clustering problems.

5.4. Computational complexity analysis

Because of the characteristics of randomized networks, usRVFL also possesses superior computational complexity. The RVFL only needs matrix inverse once during training for the $\beta \in \mathbb{R}^{(d+N) \times (d+N)}$ and the computation complexity is $\mathcal{O}((d+N)^3)$. In the edRVFL, each sub-layer requires an output weight β_i , and this weight matrix dimension is the same as single layer RVFL: $\beta_i \in \mathbb{R}^{(d+N \times L)^2}$, thus the time consumption for all β_i calculation in edRVFL only requires $\mathcal{O}(L(d+N)^3)$. The time complexity of k -means is known

Table 1
Performance comparison of the Proposed usRVFL/usdRVFL and related algorithms.

	k-Means	US-ELM	usRVFL	SC	PCAN	CLR	LRR	ELM-CLR	RM	usdRVFL
UMNIST	43.19	65.94	65.34	68.64	68.17	74.09	45.36	76.45	44.52	80.26
ORL	54.6	66.27	66.39	67.08	59.50	63.25	73.63	68.25	58.51	73.75
POLLEN	47.86	44.92	48.03	50.32	43.49	41.11	44.37	54.41	40.79	55.91
USPST	67.23	77.36	77.52	67.41	64.52	76.73	73.00	84.67	60.44	85.56
MNIST05	52.66	64.62	65.63	66.98	55.22	57.34	56.46	67.90	42.13	72.69
ETH80	50.05	50.94	53.57	55.02	49.39	41.07	53.97	56.77	48.47	60.95
ECOLI	56.25	67.69	67.50	58.93	77.68	67.26	58.13	79.49	61.61	73.85
AutoMPG	44.13	51.41	58.69	46.43	61.48	46.43	57.14	46.94	62.51	65.60
Diabetes	66.8	58.82	58.82	64.58	66.02	64.84	67.97	67.41	66.97	67.97
SEGMENT	66.57	69.51	69.35	67.9	57.57	52.94	72.16	69.78	69.39	79.35
VEHICLE	36.69	44.16	48.06	42.32	41.25	42.67	41.96	41.97	40.78	64.77
CONTROL	56.83	60.50	60.50	67.33	68.00	72.17	60.50	71.42	75.83	81.50

Table 2
Performance comparison of the Proposed usRVFL/usdRVFL and related algorithms. + means the method in the corresponding row is statistically better than the method in the corresponding column, and vice versa.

	k-Means	US-ELM	usRVFL	SC	PCAN	CLR	RM	LRR	ELM-CLR	usdRVFL
k-Means								-	-	-
US-ELM								-	-	-
usRVFL								-	-	-
SC								-	-	-
PCAN								-	-	-
CLR								-	-	-
RM								-	-	-
LRR	+							-	-	-
ELM-CLR	+	+	+	+	+	+		-	-	-
usdRVFL	+	+	+	+	+	+	+	+	+	-

as $\mathcal{O}(TKn)$, where T is the number of iterations, K is the desired number of centroids and n is the the number of samples. As $T \propto n$, the time complexity of kmeans is also usually considered to be $\mathcal{O}(n^2)$. The complexity of eigenvalue decomposition is $\mathcal{O}((NK)^2)$. Since N is usually greater than d, n and K in our experiments, the complexity of our model can be approximated as $\mathcal{O}(N^3)$, where N is the number of neurons.

Table 3 gives a comparison of the time consumption of our method with other clustering methods in four image datasets. We repeated the algorithm 10 times using the parameters yielding the optimal performance. The time consumption of usRVFL is comparable as it does not require an iterative method of optimization

Table 3
The computational time (in seconds) of the Proposed usRVFL/usdRVFL and related algorithms.

	UMIST	USPST	ORL	ETH80	MNIST05
k-Means	0.05	0.07	0.10	1.25	1.37
US-ELM	0.32	1.21	0.39	4.89	5.26
usRVFL	0.33	1.24	0.39	4.95	5.34
SC	0.40	1.19	0.58	190.96	2.18
PCAN	3.63	20.13	23.43	88.28	31.28
CLR	2.30	9.81	1.07	173.31	46.03
LRR	26.30	19.85	13.22	5430.87	7423.12
ELM-CLR	3.57	21.32	4.19	46.78	53.67
usdRVFL	2.09	5.89	2.72	20.51	25.31

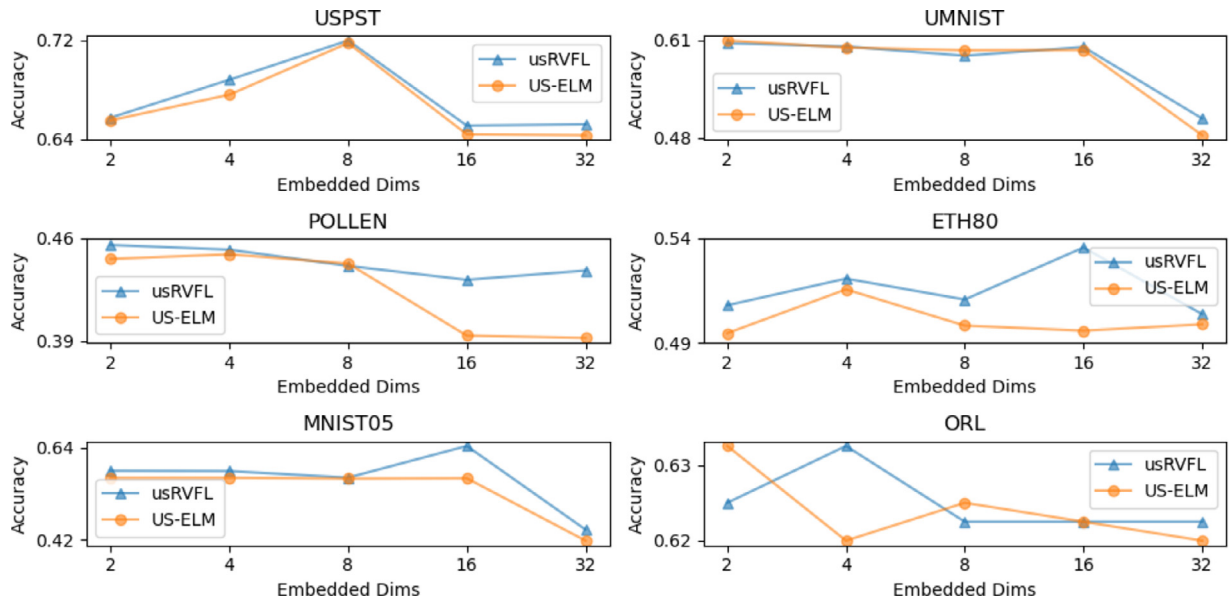


Fig. 6. Accuracy at different embedding dimensions on the Image dataset.

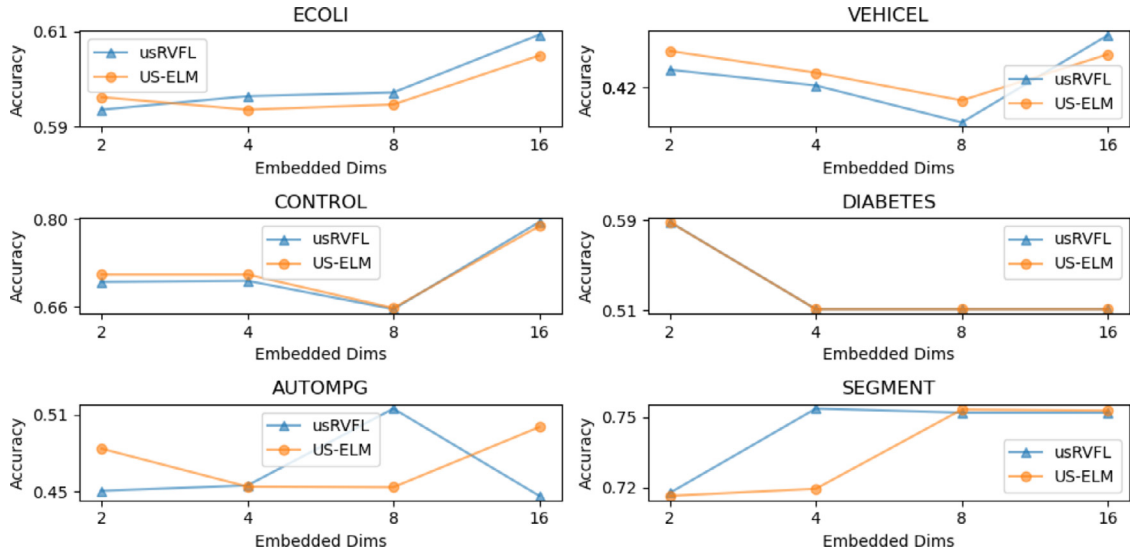


Fig. 7. Accuracy at different embedding dimensions on the UCI dataset.

Table 4

The computational time (in seconds) of usdRVFL for different hidden layers.

#layers	1	5	10	50	150	250
Time(s)	4.95	10.56	17.13	66.68	185.68	308.63

and does not require pre-processing of the data. However, usdRVFL requires multiple clustering as well as a relabelling process and is less advantageous in this respect. As the optimal layer of usdRVFL is various across the datasets, the time consumed varies widely. We have also conducted experiments with different layers of dRVFL on a relatively large dataset ETH80, even considering the extreme case of 250 hidden layers, and the calculation times are shown in Table 4.

6. Conclusion

In this paper, we propose an unsupervised RVFL framework based on manifold regularization, which enables RVFL to perform unsupervised representation learning and allows RVFL to address some popular unsupervised tasks such as clustering. We also adapt the unsupervised RVFL to a deep variant for the clustering task, which is inspired by the consensus clustering strategy.

Although there has been some work that has extended the randomized networks to unsupervised learning, to the best of our knowledge, this is the first work that proposed unsupervised RVFL extension for representation learning and clustering based on the spectral method. Evaluations of empirical evidence on a wide range of benchmark datasets have shown that usdRVFL achieves equivalent or superior clustering performance to state-of-the-art spectral clustering or graph-based methods.

For usRVFL, the clustering performance may be improved by adopting other clustering algorithms, such as graph clustering. We can try to construct some clearer graphs for embedded vectors and obtain better results. Considering usdRVFL, the adoption of different algorithms for consensus clustering is also a future research direction. Soft clustering, or Fuzzy clustering, can also be considered to make the results more practically meaningful. Moreover, the effectiveness of usdRVFL shows the promise of combining edRVFL with other methods to solve complex practical problems.

In addition to being used for clustering, the features from usRVFL are also more suitable for other downstream tasks, e.g., re-

placing the raw feature from the input domain to manifold domain, using for contrastive learning, etc.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:[10.1016/j.patcog.2022.108744](https://doi.org/10.1016/j.patcog.2022.108744).

References

- [1] G. Cybenko, Approximation by superpositions of a sigmoidal function, *Math. Control Signals Syst.* 2 (4) (1989) 303–314.
- [2] A.R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, *IEEE Trans. Inf. Theory* 39 (3) (1993) 930–945.
- [3] Y.H. Pao, G.H. Park, D.J. Sobajic, Learning and generalization characteristics of the random vector functional-link net, *Neurocomputing* 6 (2) (1994) 163–180.
- [4] P.N. Suganthan, R. Katuwal, On the origins of randomization-based feedforward neural networks, *Appl. Soft Comput.* 105 (2021) 107239.
- [5] D. Needell, A.A. Nelson, R. Saab, P. Salanevich, Random vector functional link networks for function approximation on manifolds, *arXiv preprint arXiv:2007.15776* (2020).
- [6] X. Qiu, P.N. Suganthan, G.A. Amarantunga, Ensemble incremental learning random vector functional link network for short-term electric load forecasting, *Knowl. Based Syst.* 145 (2018) 182–196.
- [7] I. Majumder, P. Dash, S. Dhar, Real-time energy management for PV-battery-wind based microgrid using on-line sequential kernel based robust random vector functional link network, *Appl. Soft Comput.* 101 (2021) 107059.
- [8] B.B. Hazarika, D. Gupta, Modelling and forecasting of COVID-19 spread using wavelet-coupled random vector functional link networks, *Appl. Soft Comput.* 96 (2020) 106626.
- [9] R. Katuwal, P.N. Suganthan, L. Zhang, An ensemble of decision trees with random vector functional link networks for multi-class classification, *Appl. Soft Comput.* 70 (2018) 1146–1153.
- [10] R. Katuwal, P.N. Suganthan, Stacked autoencoder based deep random vector functional link neural network for classification, *Appl. Soft Comput.* 85 (2019) 105854.
- [11] Q. Shi, R. Katuwal, P.N. Suganthan, M. Tanveer, Random vector functional link neural network based ensemble deep learning, *Pattern Recognit.* 117 (2021) 107978.
- [12] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in: *Nips*, vol. 14, 2001, pp. 585–591.
- [13] A.Y. Ng, M.I. Jordan, Y. Weiss, et al., On spectral clustering: analysis and an algorithm, *Adv. Neural Inf. Process. Syst.* 2 (2002) 849–856.
- [14] F. Nie, X. Wang, H. Huang, Clustering and projected clustering with adaptive neighbors, in: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 977–986.

- [15] F. Nie, X. Wang, M. Jordan, H. Huang, The constrained Laplacian rank algorithm for graph-based clustering, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.
- [16] Y.H. Pao, Y. Takefuji, Functional-link net computing: theory, system architecture, and functionalities, *Computer* 25 (5) (1992) 76–79.
- [17] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, *Neural Netw.* 2 (5) (1989) 359–366.
- [18] L. Zhang, P.N. Suganthan, Benchmarking ensemble classifiers with novel co-trained kernel ridge regression and random vector functional link ensembles [research frontier], *IEEE Comput. Intell. Mag.* 12 (4) (2017) 61–72.
- [19] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, *Neurocomputing* 70 (1–3) (2006) 489–501.
- [20] G. Huang, S. Song, J.N. Gupta, C. Wu, Semi-supervised and unsupervised extreme learning machines, *IEEE Trans. Cybern.* 44 (12) (2014) 2405–2417.
- [21] T. Liu, C.K.L. Lekamalage, G.B. Huang, Z. Lin, An adaptive graph learning method based on dual data representations for clustering, *Pattern Recognit.* 77 (2018) 126–139.
- [22] Y. Zeng, J. Chen, Y. Li, Y. Qing, G.B. Huang, Clustering via adaptive and locality-constrained graph learning and unsupervised elm, *Neurocomputing* 401 (2020) 224–235.
- [23] M. Woźniak, M. Grana, E. Corchado, A survey of multiple classifier systems as hybrid systems, *Inf. Fusion* 16 (2014) 3–17.
- [24] N.C. Oza, K. Tumer, Classifier ensembles: select real-world applications, *Inf. Fusion* 9 (1) (2008) 4–20.
- [25] B. Minaei-Bidgoli, A.P. Topchy, W.F. Punch, A comparison of resampling methods for clustering ensembles, in: *IC-AI*, 2004, pp. 939–945.
- [26] S. Vega-Pons, J. Ruiz-Shulcloper, A survey of clustering ensemble algorithms, *Int. J. Pattern Recognit. Artif. Intell.* 25 (03) (2011) 337–372.
- [27] H.G. Ayad, M.S. Kamel, Cumulative voting consensus method for partitions with variable number of clusters, *IEEE Trans. Pattern Anal. Mach. Intell.* 30 (1) (2007) 160–173.
- [28] S.T. Hadjitodorov, L.I. Kuncheva, L.P. Todorova, Moderate diversity for better cluster ensembles, *Inf. Fusion* 7 (3) (2006) 264–275.
- [29] Y. Gao, Z. Zhang, H. Lin, X. Zhao, S. Du, C. Zou, Hypergraph learning: methods and practices, *IEEE Trans. Pattern Anal. Mach. Intell.* (2020).
- [30] H.G. Ayad, M.S. Kamel, On voting-based consensus of cluster ensembles, *Pattern Recognit.* 43 (5) (2010) 1943–1953.
- [31] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, *J. Mach. Learn. Res.* 7 (11) (2006).
- [32] D.L. Donoho, C. Grimes, Hessian eigenmaps: locally linear embedding techniques for high-dimensional data, *Proc. Natl. Acad. Sci.* 100 (10) (2003) 5591–5596.
- [33] R.R. Coifman, S. Lafon, A.B. Lee, M. Maggioni, B. Nadler, F. Warner, S.W. Zucker, Geometric diffusions as a tool for harmonic analysis and structure definition of data: diffusion maps, *Proc. Natl. Acad. Sci.* 102 (21) (2005) 7426–7431.
- [34] A. Grigor'yan, Heat kernels on weighted manifolds and applications, *Cont. Math* 398 (2006) (2006) 93–191.
- [35] U. Von Luxburg, O. Bousquet, M. Belkin, Limits of spectral clustering, *NIPS*, vol. 17, Citeseer, 2004.
- [36] U. Von Luxburg, A tutorial on spectral clustering, *Stat. Comput.* 17 (4) (2007) 395–416.
- [37] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, *Neural Comput.* 15 (6) (2003) 1373–1396.
- [38] B. Ghogh, F. Karray, M. Crowley, Eigenvalue and generalized eigenvalue problems: tutorial, *arXiv preprint arXiv:1903.11240* (2019).
- [39] E. Croot, The Rayleigh Principle for Finding Eigenvalues, Technical Report, Georgia Institute of Technology, School of Mathematics, 2019.
- [40] D. Arthur, S. Vassilvitskii, k-means++: The Advantages of Careful Seeding, *Stanford*, 2006.
- [41] J. Munkres, Algorithms for the assignment and transportation problems, *J. Soc. Ind. Appl. Math.* 5 (1) (1957) 32–38.
- [42] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, Y. Ma, Robust recovery of subspace structures by low-rank representation, *IEEE Trans. Pattern Anal. Mach. Intell.* 35 (1) (2012) 171–184.
- [43] S. Dirksen, M. Genzel, L. Jacques, A. Stollenwerk, The separation capacity of random neural networks, *arXiv preprint arXiv:2108.00207* (2021).

Minghui Hu received his B.Eng. degree from Dalian Maritime University, China in 2018 and his M.Sc. degree from Nanyang Technological University, Singapore. Currently, he is a PhD student at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interest includes randomized neural networks, deep learning and computer vision.

Ponnuthurai Nagaratnam Suganthan (or P N Suganthan) received the B.A. degree, Postgraduate Certificate and M.A. degree in Electrical and Information Engineering from the University of Cambridge, UK in 1990, 1992 and 1994, respectively. He received an honorary doctorate (i.e. Doctor Honoris Causa) in 2020 from University of Maribor, Slovenia. After completing his PhD research in 1995, he served as a pre-doctoral Research Assistant in the Dept of Electrical Engineering, University of Sydney in 1995–96 and a lecturer in the Dept of Computer Science and Electrical Engineering, University of Queensland in 1996–99. He was an Editorial Board Member of the *Evolutionary Computation Journal*, MIT Press (2013–2018), *IEEE Trans on Cybernetics* (2012 – 2018), *IEEE Trans on Evolutionary Computation* (2005 –2021). He is an associate editor of *Engineering Applications of Artificial Intelligence* (Elsevier, 2022 –). *Applied Soft Computing* (Elsevier, 2018–), *Neurocomputing* (Elsevier, 2018–), *Information Sciences* (Elsevier, 2009 –), *Pattern Recognition* (Elsevier, 2001 –) and *IEEE Trans on SMC: Systems* (2020 –) Journals. He is a founding co-editor-in-chief of *Swarm and Evolutionary Computation* (2010 –), an SCI Indexed Elsevier Journal. His co-authored SaDE paper (published in April 2009) won the "IEEE Trans. on Evolutionary Computation outstanding paper award" in 2012. His research interests include randomization-based learning methods, swarm and evolutionary algorithms, pattern recognition, deep learning and applications of swarm, evolutionary & machine learning algorithms. He was selected as one of the highly cited researchers by Thomson Reuters Science Citations yearly from 2015 to 2022 in computer science. He served as the General Chair of the IEEE SSCI 2013. He has been a member of the IEEE (S'90, M'92, SM'00, Fellow 2015) since 1991 and an elected AdCom member of the IEEE Computational Intelligence Society (CIS) in 2014–2016. He is an IEEE CIS distinguished lecturer (DLP) in 2018–2021.