NEESAE: Neighboring Envelope Embedded Stacked Autoencoder

Data availability

The data and codes can be found in https://github.com/CUTEZZZOE/ESSAE

Abstract

A stacked autoencoder (SAE) is a widely used deep network since it has unique and excellent performance in feature learning. However, existing deep SAEs focus on original samples without considering the hierarchical structural information between samples. This limits the accuracy of the SAE. In recent years, state-of-the-art SAEs have suggested improvements in network structure, cost function, parameter optimization, and so on, and thereby the accuracy has been enhanced. However, the problem mentioned above is still not solved. Therefore, this paper is concerned with how to design a SAE that can conduct deep learning on hierarchically structured samples. This proposed SAE - neighboring envelope embedded stacked autoencoder (NE_ESAE) mainly consists of two parts. The first is the neighboring sample envelope learning mechanism (NSELM) which constructs sample pairs by combining neighboring samples. In addition, the NSELM constructs multilayer sample spaces by multilayer iterative mean clustering, which considers similar samples and generates layers of envelope samples with hierarchical structural information. The second part is an embedded stacked autoencoder (ESAE) to consider the original samples during training and in the network structure, thereby finding the relationship of the samples with original features and deep features in a better manner45. Different from existing SAEs, the proposed NE_ESAE can realize deep learning on hierarchical structured samples. The experimental results show that our method has significantly better performance than existing feature learning methods and some representative stacked autoencoders. Thus this paper makes SAE able to conduct cooperative deep sample and feature learning, and the advantage that has been gained can be applied to other deep neural networks.

The Impact Statement

SAE is a widely used deep neural network. However, the existing deep SAEs do not consider the relationship between neighboring samples, similar samples, and the relationship between samples with deep features and original features (we call them hierarchical structural information). This limitation makes SAEs only obtain deep features for every original sample (we call this 'flat' deep learning). With an improvement of 2.5% - 34% over the existing representative SAEs, the proposed NE_ESAE can realize deep learning on hierarchical structured samples, thereby obtaining deep features for different layers of structured samples (which we call envelope samples) and with more complementarity with the original features. In other words, the proposed NE_ESAE can realize cooperative deep sample and feature transformation. More importantly, this proposed model can be considered to be a framework rather than concrete model, in that the innovations achieved in this framework can be applied to other deep neural networks and transform their current work patterns.

Key words: Feature learning; Envelope learning; Stacked autoencoder; Deep learning.

1. Introduction

In recent years, the growth in deep learning has led many researchers to try to use them for feature learning. They have been implemented in the fields of image classification [1]-[5], speech recognition [6], machine translation [7] and time series prediction [8][9]. Typical deep neural networks include convolutional neural networks (CNNs) [1], recurrent neural networks (RNNs) [9], stacked autoencoders (SAEs) [10], deep belief networks (DBNs) [11], and long short-term memory (LSTM) [12]. Compared with traditional feature learning, the main advantage of deep learning is its powerful feature self-learning ability. Through multilayer nonlinear transformations, the network can learn the main driving variables in the input data and obtain more essential information about the original data [13].

Among the deep feature learning methods, SAE, as an unsupervised learning network, can automatically learn effective features from a large amount of unlabeled data and is widely used. To date, various revised SAEs have also been successively proposed by researchers. Rui Li *et al*. [14] proposed a supervised autoencoder, by adding an additional classification layer on top of the representation layer to jointly predict the target and reconstruct the input, They then built a stacked supervised autoencoder for the classification task. Their model is able to learn to recognize features, which significantly improves the classification performance. To solve the problems of finding feature representations that minimize the distance between source and target domains and avoiding data redundancy that may lead to the performance degradation of transfer learning, Yi Zhu *et al*. [15] proposed a new SAE framework for semisupervised representations for transfer learning. To solve the problem of imbalanced learning, Nima Farajian *et al*. [16] proposed an SAE-based imbalanced learning method that includes feature learning and classification steps. In the feature learning step, by stacking two regularized autoencoders, a one-class learning method was used to extract meaningful features from a small number of data and capture its underlying manifold. The proposed method achieves quite good performance. Ahmad M. Karim [17] integrated the postprocessing procedure into the data classification framework and proposed a new data classification framework that combines a postprocessing system composed of sparse SAEs and a linear system model based on particle swarm optimization (PSO). The framework can be applied to any data classification problem with only minor updates, such as changing some parameters, including input features, hidden neurons, and output classes. Wenjuan Wang *et al*. [18] proposed an efficient stacked contractive autoencoder (SCAE) method for unsupervised feature extraction in view of the large-scale, high-dimensional, and highly redundant network traffic characteristics in cloud computing environments. By using SCAE methods, better and more robust low-dimensional features can be learned automatically from raw network traffic. Weifeng Liu *et al*. [19] proposed a large-capacity autoencoder (LMAE) to further improve the discriminability by enforcing a large marginal distribution of samples of different classes in the hidden feature space. Finally, these features were put into a softmax classifier for classification. Due to the unsupervised self-construction of SAE in the pretraining stage, the correlation of deep features with fault types cannot be ensured. Wang Y *et al.* [20] proposed a cascaded supervised autoencoder to pretrain deep networks and obtain deep fault-related features from raw input data. In each supervised autoencoder, informative features are learned from the input data with the aim of distinguishing different failure types to a large extent. The network gradually learns high-order fault-related features from the original input data, which improves the classification accuracy of the classifier.

Although SAEs have achieved great success in many applications [21]-[23], it is still a challenging problem to design a deep autoencoder for effective feature learning [24]. Existing SAEs mainly consider minimizing the error between each input sample and its output with reconstructed deep features, but the structural information between samples is ignored. As shown in Fig. 1, the existing autoencoder does not consider the relationship (structure information) between the ith sample and the jth sample. However, this will lead to a decrease in separability between samples, limit the search for optimal samples, and limit the quality of optimal features, thereby affecting the classification performance of the algorithm [25]. Therefore, it is necessary to consider the structural information among samples for SAE.

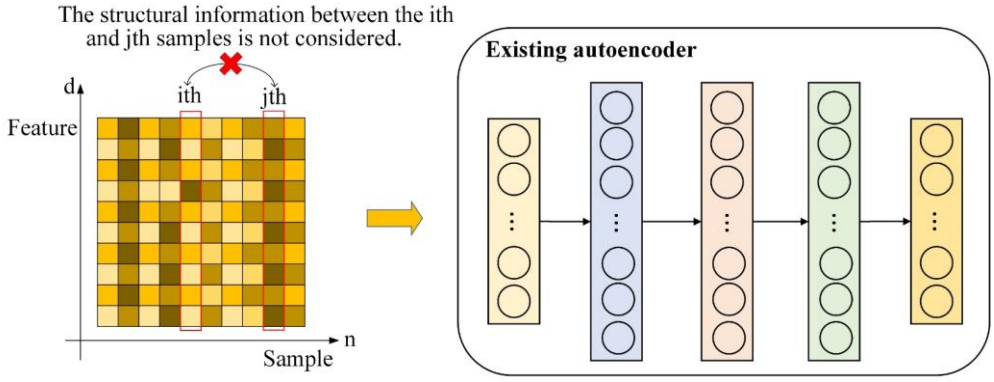
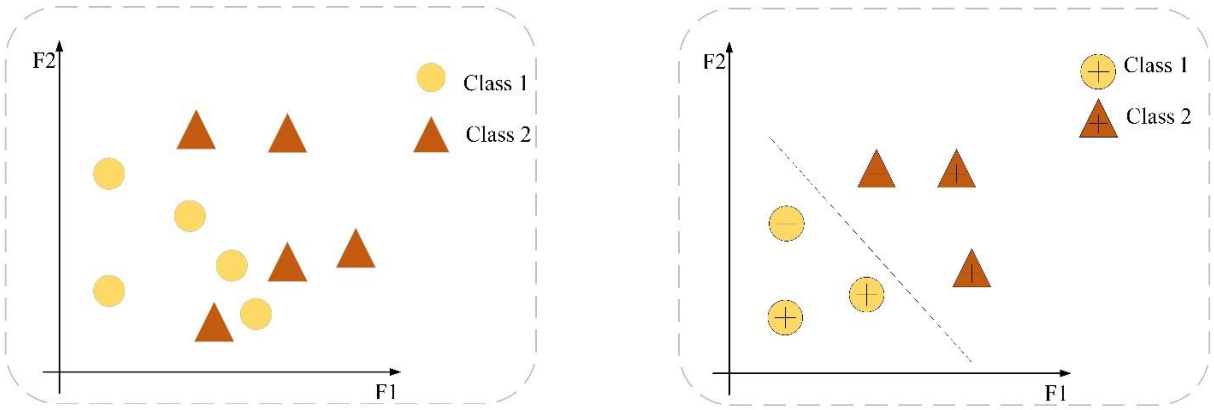


Fig. 1 The existing autoencoder model.

It is challenging to find the optimal features for the original samples. Fig. 2 shows the distribution of the original samples. Fig. 2(a) shows the distribution of the original samples under feature spaces F1 and F2. As shown in Fig. 2 (a), it is difficult to achieve linear separability and to find a reliable optimal feature set. Fig. 2(b) shows the distribution of the clustering-based transformed samples under F1 and F2. As shown in Fig. 2 (b), the separability is significantly improved by sample transformation under the

same features. Sample transformation is helpful to mine the structural information between samples and construct 'bigger' samples with better separability. This means that the transformed samples should be considered for deep learning, apart from the original samples.

For some studies, SAE input conducted batches of original samples. The SAE can possibly learn the structural information indirectly. However, there are some limitations. 1) Suppose there are relationships between the ith and jth samples, but not all the samples may be in the same batch and hence the SAE cannot extract structural information. 2) Generally, the order of input samples is disrupted during training and thus the structural information among samples will be destroyed. 3) Even if structural information exists, the number of possible combinations will explode, and the risk of overfitting will increase significantly. For example, suppose the number of features is d, the combination of features is recorded as $C1(d)$, the number of samples is n, and the combination of samples is recorded as $C2(n)$. Then, the number of combinations of the two will be $C1(d) \times C2(n)$, and the risk of overfitting will grow significantly. Therefore, it is important to combine the sample transformation and SAE to consider the structural information among samples well. Unfortunately, the existing SAEs neglect this point.



(a) Before sample transformation. (b) After sample transformation.

Fig. 2 Separability before and after sample transformation.

The existing sample transformation methods mainly include the nearest neighbor rule based [26], the density based [27], the random sampling based [28] and the clustering based [29]. The nearest neighbor-based and density-based methods have very high computational complexity. Random sampling-based methods generally only obtain local optimal values and depend on the selection of the initial dataset. Clustering-based methods can separate a finite unlabeled dataset into a finite and discrete set of data structures [31], analyze the internal relationships of the data and aggregate similar samples together so that the classifier focuses on samples belonging to different categories in the same cluster, thereby reducing the difficulty of classification facing a confusing boundary [32]. Clustering can be considered to mine sample structure information and takes the generated cluster center as a new sample. Existing clustering methods mainly include the k-means algorithm [33], hierarchical clustering [34], density-based clustering [35], grid-based clustering [36], self-organizing map (SOM) [37], *etc. A*mong them, k-means clustering can prune the tree according to the category of the lesser-known clustering samples and has the advantages of simplicity, easy implementation, fast convergence speed, excellent clustering effect, and strong interpretability. Therefore, we consider design iterative means clustering based on k-means clustering to mine the hierarchical structure of the sample space and construct multilayer sample spaces to obtain effective structural information. However, there will be differences in the distribution of samples before and after clustering [38]. Hence we introduce the maximum mean difference (MMD)[49] to reduce the distribution difference of samples before and after clustering. The simple MMD is not sufficient and needs to be combined with the local and global distribution difference. The measured distribution differences can be used to impose constraints on the training model, and the model can be optimized [40].

In addition, complementarity between samples with original features and with deep features is important. However, existing SAEs did not consider this point. To solve this problem, this paper proposes a new SAE model that considers structural information among samples well. The proposed SAE is called the 'neighboring envelope embedded stacked autoencoder'. First, the neighboring sample envelope learning mechanism (NSELM) is designed to preprocess the original samples and to construct layers of envelope samples for the input of SAE. Second, the embedded stacked autoencoder (ESAE) is designed to mine the relationship between

samples with original features and samples with deep features during training and within the network. Third, several base classifiers are conducted on the layers of samples. Fourth, a multilayer space ensemble mechanism (MSEM) is designed to fuse the results from layers of classification. The main contribution of this paper can be expressed as follows.

(1) The neighboring sample envelope learning mechanism (NSELM) is proposed to construct hierarchical new samples (called 'deep samples' or 'envelope samples'). First, a sample-pair connection mechanism (SPC) is proposed to construct a sample pair for neighboring samples. Then, multilayer iterative mean clustering with an interlayer consistency mechanism (ICMC) is proposed to achieve deep sample transformation and mine the structural information of the sample. This mechanism deeply mines the information among the samples, thereby constructing more powerful 'bigger' samples (called 'envelope samples').

(2) SPC is proposed to mine the relationship between neighboring samples, thereby considering neighborhood structural information among samples and obtaining 'larger' samples with more abundant features.

(3) ICMC is proposed to mine the relationship between similar samples, thereby considering similarity structural information among samples and obtaining more powerful samples.

(4) The deep features obtained by the existing stacked autoencoders have poor quality and insufficient complementarity with the original features, which limits the feature complementarity fusion performance. The embedded stacked autoencoder (ESAE) is designed in this paper to mine the relationship between samples with original features and samples with deep features during training and within the network. In other words, the ESAE is able to mine the structure information with different feature sets, thereby achieving high-quality deep features.

(5) Based on the NSELM and ESAE, a new SAE model is proposed and is called the neighboring envelope embedded stacked autoencoder (NE_ESAE). This model realizes the sample-feature cooperative transformation, while existing SAEs only realize the feature transformation.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 describes the proposed method. Section 4 presents the experimental results. The last section provides the conclusions.

2. Related works

This section briefly introduces the related works of the autoencoders.

2.1 Stacked autoencoder (SAE)

In the SAE [41], multiple autoencoders are stacked together and trained separately, and the features output by each layer are used as input to the next layer. To use the learned features for classification, a deep multilayer network is created by adding a softmax layer on top of the last autoencoder, and then the whole network is trained and fine-tuned.

An autoencoder consists of two stages: encoding and decoding. The encoding stage converts the input data into a hidden layer representation through a linear transformation and a nonlinear activation function. The process can be described as

$$\boldsymbol{h} = \sigma(\mathbf{W}_1^T \boldsymbol{x} + \mathbf{b}_1) \tag{1}$$

where $\boldsymbol{x}$ is the input data and $\boldsymbol{h}$ is the hidden layer feature. $\mathbf{W}_1$, $\mathbf{b}_1$ and $\sigma(\cdot)$ represent the weight, bias and the activity function of the encoding stage. In the decoding stage, the hidden layer features are mapped to the original data space, and the original data are reconstructed. The decoding process can be described as

$$\boldsymbol{x}' = g(\mathbf{W}_2^T h + \mathbf{b}_2) \tag{2}$$

where $\boldsymbol{x}'$ is the reconstructed data and $\mathbf{W}_2$, $\mathbf{b}_2$ and $g(\cdot)$ are the activity functions of the decoding stage. The training goal of the autoencoder is to minimize the reconstruction error between the output data $\boldsymbol{x}'$ and the original data $\boldsymbol{x}$. The objective function is defined as

$$\arg\min_{\theta} \frac{1}{n}\sum_{i=1}^{n} \left\| \boldsymbol{x}_i - \boldsymbol{x}_i' \right\|^2 + \lambda(\left\| \mathbf{W}_1 \right\|_2 + \left\| \mathbf{W}_2 \right\|_2) \tag{3}$$

where n is the number of samples and λ is the coefficient of the weight regularization term, which can alleviate overfitting to a certain extent.

2.2 Stacked sparse autoencoder (SSAE)

In the case of using the sparse autoencoder and considering their stacking, this method is called the stacked sparse autoencoder (SSAE) [42]. A sparse autoencoder is obtained by adding some sparse constraints to the traditional autoencoder, which can suppress most of the output of hidden-layer neurons. The objective of training the sparse autoencoder is formulated as follows:

$$\arg\min_{\theta} \frac{1}{n}\sum_{i=1}^{n}\left\|\boldsymbol{x}_i - \boldsymbol{x}_i'\right\|^2 + \lambda(\left\|\mathbf{W}_1\right\|_2 + \left\|\mathbf{W}_2\right\|_2) + \beta\sum_{j=1}^{d}\mathrm{KL}(\rho\left\|\hat{\rho}_j\right) \tag{4}$$

where β is the coefficient for the sparsity regularization term, d is the number of neurons in the hidden layer, and $\hat{\rho}_j$ is the average activation value of the *j*th hidden layer neuron on all training samples, which is forced to be similar to the given sparseness through KL divergence. When $\hat{\rho}_j = \rho$, $\hat{\rho}_j$ there is $\mathrm{KL}(\rho\left\|\hat{\rho}_j\right) = 0$. Therefore, by setting a sufficiently small parameter ρ, the output of most neurons in the hidden layer can be 0, and then a sparse representation can be obtained.

2.3 Weight-clustering sparse autoencoder (WCSAE)

Fan *et al*. [43] proposed a weight-clustering sparse autoencoder (WCSAE) combining object-oriented classification and differential images. They believed that in high-order feature extraction, when there are similar weights in the weight matrix, redundant features will be extracted, and the performance of data representation cannot be improved. Therefore, the similar weights in the hidden layer of the WCSAE are clustered layer by layer. Specifically, the weights of SAE are coordinated to the optimal value through the objective function given by Eq. (4). Then, to estimate the similarity between weights, the similarity function is defined as follows:

$$C(\boldsymbol{w}_i, \boldsymbol{w}_j) = \frac{\sum_{p=1}^{d}(\boldsymbol{w}_{ip} - \bar{\boldsymbol{w}}_i)(\boldsymbol{w}_{jp} - \bar{\boldsymbol{w}}_j)}{\sqrt{\sum_{p=1}^{d}(\boldsymbol{w}_{ip} - \bar{\boldsymbol{w}}_i)^2 \sum_{p=1}^{d}(\boldsymbol{w}_{jp} - \bar{\boldsymbol{w}}_j)^2}} \tag{5}$$

where $\boldsymbol{w}_{ip} \in \boldsymbol{w}_i, \boldsymbol{w}_{jp} \in \boldsymbol{w}_j, i, j \in [1, n], i \neq j$. For the sake of merged weights that possess similarity, Fan adopts a bottom-up clustering algorithm called hierarchical clustering. Then, the clustered weight and bias can be obtained, which will be fine-tuned by Eq. (4) to obtain a new weight and bias using the pretraining method. WCSAE can serve the tasks of classification when the procedures of clustering and fine-tuning stop. The termination conditions satisfy the following formula:

$$\begin{cases} \sum C(\boldsymbol{w}_i, \boldsymbol{w}_j) \geq \tau \\ n' \geq 0.1 \times n \end{cases} \tag{6}$$

where τ is the threshold of the sum of weight similarity and n' is the number of rows of the new weight matrix by clustering.

2.4 Distance constrained sparse stacked autoencoder (DCSSAE)

Shi Y *et al*. [44] introduced distance constraints into sparse stacked autoencoders to construct distance-constrained sparse stacked autoencoder (DCSSAE). The DCSSAE network can largely separate the target from the background in the new feature space by adding distance constraints whose goal is to maximize the difference between the target pixel and other background pixels in the feature space.

For each background spectrum in the new feature space, we expect the distance between the target and background to be as large as possible, described as follows.

$$\max_{\theta} \sum_{j=1}^{q} s_j \left\|\mathbf{b}_j - \mathbf{d}'\right\|_2^2. \tag{7}$$

where $s_j \in (0,1)$, $\mathbf{b}_j'$ and $\mathbf{d}'$ are expressions of $\mathbf{b}_j$ and $\mathbf{d}$ in the new feature space, defined as

$$s_j = \exp(-\frac{\left\|\mathbf{b}_j - \mathbf{d}\right\|^2}{\sigma 1}) \tag{8}$$

where $\sigma 1$ is an adjustable parameter. Generally, the smaller $\sigma 1$ is, the larger the distances between the target and background in the new feature space.

To simplify (12), we define a new matrix $\mathbf{P} \in \mathbf{R}^{n\times(q+1)}$

$$\mathbf{P}_{i,j} = \begin{cases} 1, & \text{if } i = i\mathbf{d}x_j \\ 0, & \text{else} \end{cases} \tag{9}$$

Then, we have

$$[\mathbf{d}'\mathbf{b}_1'\cdots\mathbf{b}_q'] = \mathbf{HP} \tag{10}$$

where $\mathbf{H} = [\boldsymbol{h}^1, \boldsymbol{h}^2, \cdots, \boldsymbol{h}^n]$ represents the output matrix corresponding to the original input matrix. Combining (12) and (14),

$$\begin{aligned} &\max_{\theta} \sum_{j=1}^{q} s_j tr((\mathbf{b}_j' - \mathbf{d}')(\mathbf{b}_j' - \mathbf{d}')^{\mathbf{T}}) \\ &= \max_{\theta} tr(\mathbf{HP}\begin{bmatrix} -\mathbf{e}_q \\ \mathbf{I}_q \end{bmatrix} diag(s_j)[-\mathbf{e}_q \ \mathbf{I}_q]\mathbf{P}^{\mathbf{T}}\mathbf{H}^{\mathbf{T}}) \\ &= \max_{\theta} tr(\mathbf{HZH}^{\mathbf{T}}) \end{aligned} \tag{11}$$

where $\mathbf{Z} = \mathbf{P}\begin{bmatrix} -\mathbf{e}_q \\ \mathbf{I}_q \end{bmatrix} diag(s_j)[-\mathbf{e}_q \ \mathbf{I}_q]\mathbf{P}^{\mathbf{T}}$ and $\mathbf{e}_q = \underbrace{[1\cdots 1]^T}_{q}$, and $\mathbf{I}_q$ is the identity matrix.

Finally, Eq. (4) and Eq. (11) are concatenated to form a range-constrained sparse stacked autoencoder (DCSAE), and a compromise parameter δ is introduced to control the effect of additional distance constraints

$$\arg\min_{\theta} \frac{1}{n}\sum_{i=1}^{n} \left\| x_i - x_i' \right\|^2 + \lambda(\left\|\mathbf{W}_1\right\|_2 + \left\|\mathbf{W}_2\right\|_2) + \beta\sum_{j=1}^{d} KL(\rho \left\| \hat{\rho}_j \right.) - \delta tr(\mathbf{HZH}^{\mathbf{T}}) \tag{12}$$

2.5 Latent relationship guided stacked sparse autoencoder (LRSSAE)

Liu L *et al*. [45] improved the feature learning ability and extracted more representative feature representations than the basic model by proposing the latent relationship guided stacked sparse autoencoder (LRSSAE). In LRSSAE, the manifold learning algorithm is used to learn latent relationships to guide feature learning. LRSSAE integrates the loss function constructed by superpixels into the SAE objective function and stacks the stacked sparse autoencoder. An SSAE is combined with graph regularization of the latent relations of each hidden layer and superpixel constraints at the top layer to extract feature representations in an unsupervised manner.

The objective function of LRSSAE can be defined as follows:

$$\begin{aligned} \arg\min_{\theta,\mathbf{R}} \ & \sum_{t=1}^{\vartheta-1} (J_{SAE}(\theta) + \xi_1 J_{LR}(\theta, \ \mathbf{R}) \\ & + \sum_{t=T} (J_{SAE}(\theta) + \xi_1 J_{LR}(\theta, \ \mathbf{R}) + \xi_2 J_{SP}(\theta, \ \mathbf{R})) \\ s.t. \ & \mathbf{R} = \mathbf{U}(\theta) \end{aligned} \tag{13}$$

where $\mathbf{U}$ represents the similarity of HSI pixels, which is determined by the combination of spectral correlation and spatial correlation. ϑ denotes the number of SAEs used in the LRSSAE model, and ζ_1 and ζ_2 are the trade-off factors. J_{SAE} denotes the SAE objective function, which is defined in Eq. (4). J_{LR} denotes the regularization term of the latent relationship, which is constructed in each layer of the SSAE.

$$\begin{aligned} J_{LR} &= \frac{1}{2}\sum_{i,j}^{S^{(l)}} \mathbf{R} \left\| (a)_i^{(l)} - (a)_j^{(l)} \right\|_F^2 \\ &= Tr(\mathbf{a}^{(l)}\mathbf{L}(\mathbf{a}^{(l)})^{\vartheta} \end{aligned} \tag{14}$$

where the Laplacian matrix of the latent relationship graph can be denoted by $\mathbf{L}$, which is constructed by $\mathbf{L} = \mathbf{D} - \mathbf{R}$. $\mathbf{D}$ is a diagonal matrix, in which the diagonal element (i,i) equals the sum of the ith row of $\mathbf{R}$. The superpixel regularization term is represented by J_{SP}, which is used only in the top layer of the SSAE.

$$J_{LR} = \sum_{j=1}^{S^{(t)}} \exp\left(\frac{\left\|\mathbf{a}_j - \zeta_k\right\|_2^2}{\gamma_2}\right) + \varphi \left\|\mathbf{R} - \bar{\mathbf{R}}\right\|_F^2 \quad (15)$$

where $\mathbf{a}_j$ denotes the feature representations, ζ_k represents the feature of the *k*th superpixel, and γ_2 denotes the heat kernel. φ is a trade-off parameter, and $\bar{\mathbf{R}}$ is the statistical latent relationship of superpixel *k*.

2.6 Stacked denoising sparse autoencoder (SDSAE)

Pelin Görgel *et al*. [46] proposed a stacked denoising sparse autoencoder (SDSAE) that stacks the denoising and sparse autoencoders together to form a robust deep mode. Each denoising sparse autoencoder takes input from the activations of the previous layer and is pretrained independently. The goal of pretraining is to optimize some similar objective so that the parameters of all layers are layered in a region of the parameter space. The stacked autoencoder can extract useful features through a series of learning stages. A stacked autoencoder is trained using greedy hierarchical training, and the fine-tuned ensemble output is used as input to the classification algorithm. This study uses both a (multiclass) support vector machine and a softmax classifier.

2.7 Stacked Pruned Sparse Autoencoder (SPSAE)

Zhu *et al*. [47]proposed a new stacked pruned sparse autoencoder (SPSAE) model. Different from traditional autoencoders, this model contains a fully connected autoencoder, which utilizes the dominant features extracted in all layers to participate in subsequent layers thereby reducing the loss of information. In addition, to solve the problem of increasing the number of SAE units and training time, a pruning operation is added to the model. That is, the input data of units with large reconstruction errors are prohibited from participating in the subsequent training process during pretraining.

2.8 Gated Stacked Target-Related Autoencoder (GSTAE)

Sun *et al*.[48] proposed a gated stacked target-related autoencoder (GSTSAE), which is a novel deep feature extraction and layer wise ensemble method. For the output prediction, it is a more reasonable AE to pay more attention to the extraction of highly correlated features. Therefore, when extracting features layer by layer, they directly add the prediction error term of the target value to the loss function of the AE model instead of only considering the reconstruction error, which is called the target-related autoencoder (TAE). In this way, the characteristics of each layer are expected to be not only a representation of the initial input data but also be a valid explanatory variable of the target value.

The encoded features of TAE are not only decoded into the reconstructed input data through Eq. (2) but also utilized for predicting the target value through the mapping $\tilde{\mathbf{y}} = f(\mathbf{W}_p \bullet \mathbf{h} + b_p)$. The training loss function is accordingly defined as follows:

$$J_{TAE} = \frac{1}{n}\sum_{i=1}^{n} \left\|\boldsymbol{x}_i - \boldsymbol{x}_i'\right\|^2 + \frac{\omega}{n}\sum_{i=1}^{n} \left\|\tilde{\mathbf{y}}_i - \mathbf{y}_i\right\|^2 \quad (16)$$

where ω is a tradeoff coefficient to balance the weights of reconstruction loss and prediction loss in the loss function. The STAE stacks multiple TAEs together and trains separately, and the features output by each layer are used as input to the next layer. The loss function of the k*th* (k>1) TAE can be similarly defined as follows.

$$J_{k-TAE} = \frac{1}{n}\sum_{i=1}^{n} \left\|\tilde{\mathbf{h}}_i^{k-1} - \mathbf{h}_i^{k-1}\right\|^2 + \frac{\lambda}{n}\sum_{i=1}^{n} \left\|\tilde{\mathbf{y}}_i - \mathbf{y}_i\right\|^2 \quad (17)$$

where $\tilde{\mathbf{h}}_i^{k-1}$ and $\mathbf{h}_i^{k-1}$ are the input and reconstructed input data of the k*th* TAE.

The STAE considers the target correlation information in the learning process. To improve the performance of the model, Sun et al. focus on making full use of different levels of feature representation at different layers and introduce gating neurons to further mine and integrate information from different layers.

Suppose the depth of the STAE is l layers, and $\mathbf{H}_1, \mathbf{H}_2, \cdots, \mathbf{H}_l$ denote the features from the first layer to the last layer. For the k*th* layer, the features are first used to generate a gated value as follows:

$$gk = f(\mathbf{W}_{gk} \bullet \mathbf{H}_k + \mathbf{b}_{gk}) \tag{18}$$

where $\mathbf{W}_{gk} \in \mathbf{R}^{1 \times lk}$, $\mathbf{H}_k \in \mathbf{R}^{lk \times 1}, \mathbf{b}_{gk} \in \mathbf{R}^{1 \times 1}$ and lk is the number of neurons in the k*th* hidden layer. After that, a backward fine-tuning procedure is conducted to solve the parameter set $\theta_g = \{\mathbf{W}_{e1}, \mathbf{b}_{e1}; \cdots, \mathbf{W}_{el}, \mathbf{b}_{el}; \mathbf{W}_{g1}, \mathbf{b}_{g1}; \cdots; \mathbf{W}_{gl}, \mathbf{b}_{gl}; \mathbf{W}_{o1}, \mathbf{b}_{o1}; \cdots; \mathbf{W}_{ol}, \mathbf{b}_{ol}\}$ by solving the following unconstrained optimization problem:

$$\arg\min_{\theta_g} \frac{1}{n} \sum_{i=1}^{n} \left\| \tilde{\mathbf{y}}_i - \mathbf{y}_i \right\|^2 \tag{19}$$

3. Proposed Method

This section describes the two aspects of the proposed method. In the following, Section 3.1 describes the proposed NE_ESAE. Section 3.2 describes the classification algorithm based on NE_ESAE.

The main symbols used in this paper and their meanings are listed in Table 1; the main proper nouns used in this paper and their meanings are listed in Table 2.

Table 1 Main symbols and their meanings

Symbol	Meaning
$\mathbf{X}$	Original training data, $\mathbf{X} \in \mathbf{R}^{n \times d}$ n and d are the sample size and feature size, respectively.
$\mathbf{X}^{Test}$	Original testing data.
$\boldsymbol{x}$	The original sample.
$\phi(\cdot)$	SPC operator, which is used to reconstruct the original sample.
$\mathbf{X}_{OS}$	Generated sample-pair. $\mathbf{X}_{OS} \in \mathbf{R}^{n \times 2d}$ n and $2d$ are the sample size and feature size, respectively.
$\mathbf{S}_P$	Clustering sample set.
$\mathbf{Q}$	The generative transfer matrix.
$\varphi(\mathbf{X}_{PS})$	The training set of the primary clustering sample.
$\varphi(\mathbf{X}_{OS})$	The training set of the original sample-pair.
$\varphi(\mathbf{X}_{GIS})$	The training set of the generated intermediate sample.
$\varphi(\mathbf{X}_{GIS}^{i})$	A sample in the GIS.
$\varphi(\mathbf{X}_{OS}^{j})$	A sample in the OS.
Ψ	Linearly transformed projection matrix.
$\boldsymbol{\Phi}$	Linear combination coefficient matrix in the first layer of ICMC.
$\boldsymbol{\Phi}_1$	Linear combination coefficient matrix in the second layer of ICMC.
$\boldsymbol{\Theta}$	Auxiliary variable.
$\mathbf{E}$	Augment Lagrangian multipliers.
$\rho 1$	Penalty parameter.

$\mathbf{H}$	Hidden feature, $\mathbf{H} \in \mathbf{R}^{n \times d^k}$. d^k is the number of hidden-layer units of the kth AE.
λ	Coefficient for the L2 weight regularization term.
β	Coefficient for the sparsity regularization term.
ρ	Sparse parameter.
$\hat{\rho}_j$	Average activation value of all training samples on the *j*th hidden neuron.
$\mathbf{X}_{DF}$	Data extracted by ESAE, $\mathbf{X}_{DF} \in \mathbf{R}^{n \times q}$.
$\mathbf{X}_{PF}$	The matrix after the feature reduction.

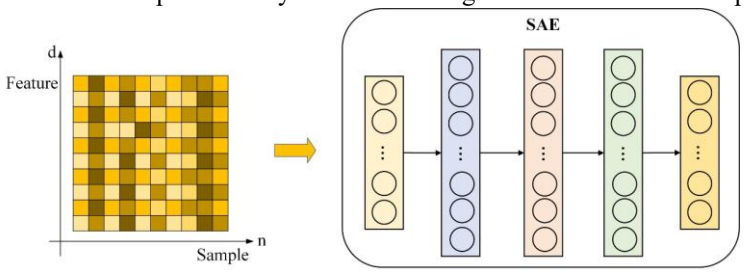
Table 2 Main proper nouns and their meanings

Symbol	Meaning
NE_ESAE	Neighboring envelope embedded stacked autoencoder.
NSELM	Neighborin sample envelope learning mechanism,
ESAE	Embedded stacked autoencoder.
SPC	Sample-pair connection based on near-neighbor.
ICMC	Interlayer consistency mechanism based on multilayer iterative mean clustering.
IMC	Iterative mean clustering.
ICM	Interlayer consistency mechanism.
OS	The original sample-pair space.
PS	The primary clustering sample space.
SS	The secondary clustering sample space.
GIS	The generated intermediate sample space.
ES	Envelope sample (or envelope samples).
GDD	Global distribution difference.
LDD	Local distribution difference.
MSEM	Multilayer space ensemble mechanism.
WF	Weighted fusion.
MV	Majority vote.

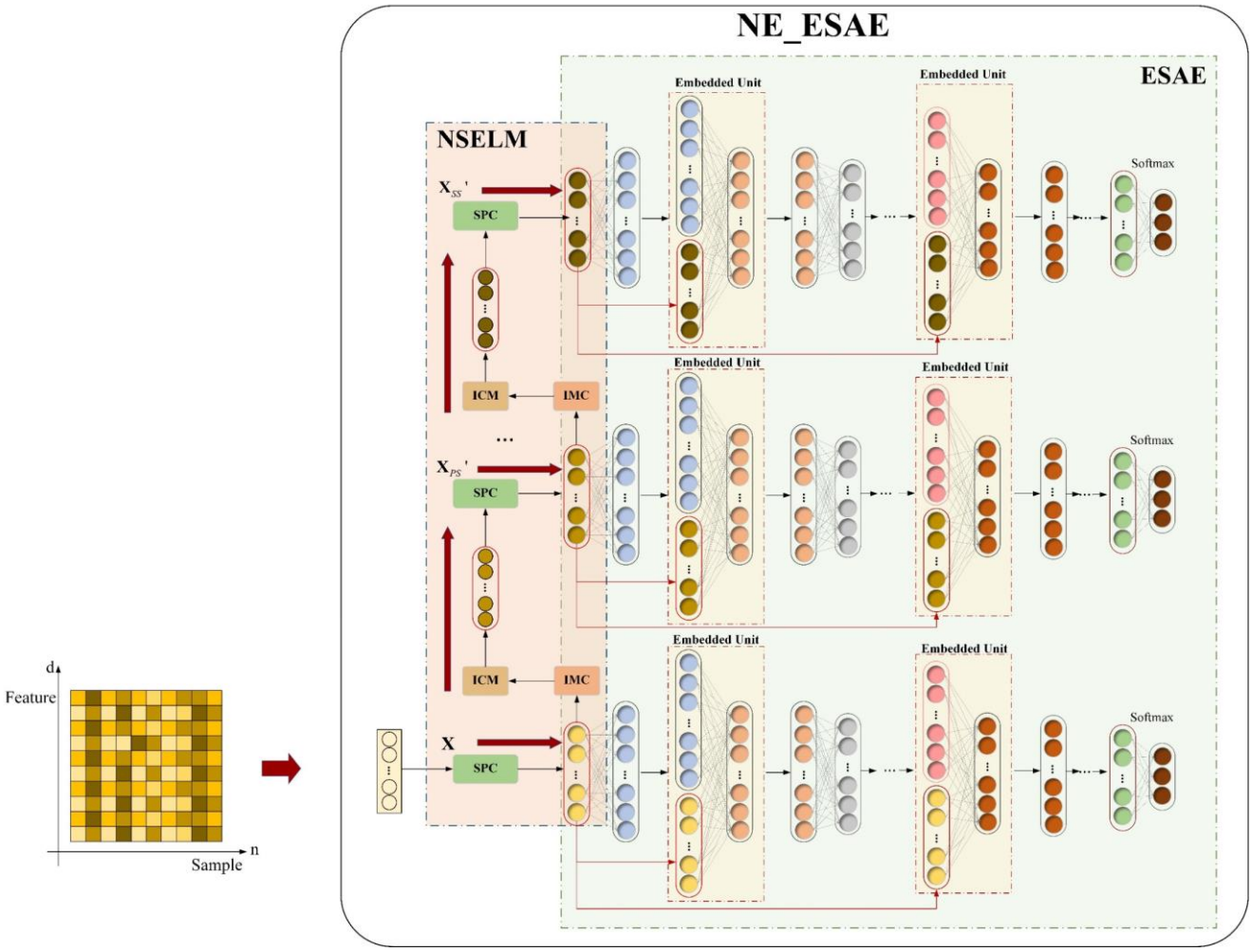
3.1 Neighboring Envelope Embedded Stacked Autoencoder (NE_ESAE)

Fig. 3(a) shows the flowchart of the traditional SAE. Fig. 3 (b) shows the flowchart of the proposed NE_ESAE, which mainly includes the neighboring sample envelope learning mechanism (NSELM) and embedded stacked autoencoder (ESAE). The NSELM mines the spatial information of the samples to realize the multilayer transformation of samples. ESAE is designed and trained to

extract multilayer deep features. Different from the traditional SAE in Fig. 3 (a), NE_ESAE can mine the hierarchical structure information of the original samples and the complementarity between the original features and the deep features.



(a) The traditional SAE



(b) The proposed NS_ESAE.

Fig. 3 Flowchart of the traditional SAE and NE_ESAE.

3.1.1 Neighboring Sample Envelope Learning Mechanism (NSELM)

The proposed NSELM consists of two parts. The first part is the sample-pair connection mechanism (SPC) based on a near-neighbor sample, which constructs a sample pair by connecting the original sample with the nearest-neighbor sample. The second part is ICMC, which can be seen as a combination of iterative mean clustering (IMC) and the interlayer consistency mechanism (ICM). ICMC constructs multilayer sample spaces and achieves local and global consistency of structural information between interlayers to better realize more powerful and 'bigger' samples.

A. Sample-pair connection mechanism based on near-neighbor (SPC)

To mine the neighborhood relationship between samples, this paper proposes SPC. We propose the $\phi(\cdot)$ operator for transforming (reconstructing) the original sample. It selects a neighboring original sample, then captures the intrinsic structure of near-neighbor samples and finally connects the original sample with the near-neighbor sample to generate a sample pair.

Given the original training data $\mathbf{X}$, the sample pair is constructed as follows. First, we randomly pick $\boldsymbol{x}_j$ from the same class sample set of $\boldsymbol{x}_i$ in $\mathbf{X}$ and calculate the Euclidean distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Based on this, we can find the neighboring sample $\boldsymbol{x}_j'$, which is closest to $\boldsymbol{x}_i$. We use $\mathbf{D}$ to represent the distance, and the objective function can then be expressed as

$$\min_{\mathbf{D}} \left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|_2 \tag{20}$$

Therefore, concatenating $\boldsymbol{x}_i$ and $\boldsymbol{x}_j'$, the SPC can be expressed as

$$\begin{aligned} \mathbf{X}_{OS} &= \phi(\boldsymbol{x}_i) \\ &= [\boldsymbol{x}_i, \boldsymbol{x}_j'] \end{aligned} \tag{21}$$

The calculation process of $\phi(\cdot)$ is shown in Fig. 4. In the figure, sample pair $\mathbf{X}_{OS}$ is generated by SPC. We refer to the sample space consisting of $\mathbf{X}_{OS}$ as the original sample-pair space (OS), which contains neighborhood structural information among the original samples (relationship between neighboring original samples).

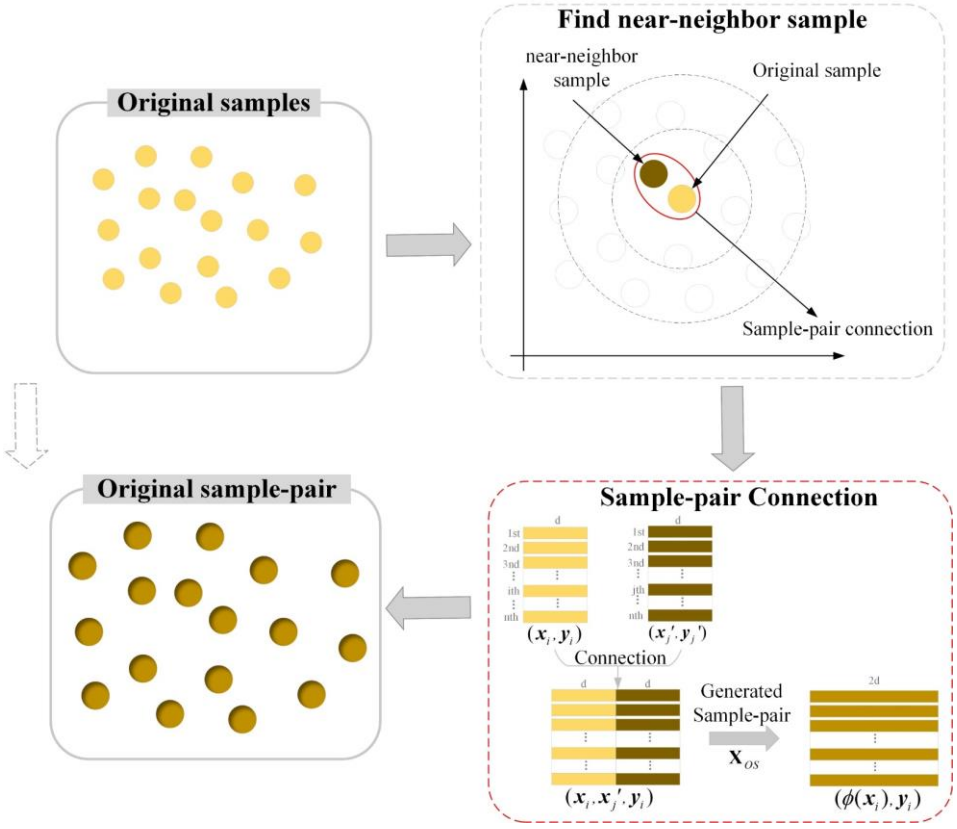


Fig. 4 Sample-pair connection mechanism. First, an original sample is selected; then, the intrinsic structure of similar neighbor samples is captured; and finally, the original sample and the near-neighbor sample are connected to generate a sample pair.

B. Multilayer iterative mean clustering (ICMC) based on the interlayer consistency mechanism

To mine the relationship between similar samples, this paper proposes the ICMC and is shown in Fig. 5. As shown in the figure, ICMC mainly consists of IMC and ICM. IMCs are used to construct the multilayer sample spaces, and ICM is designed to effectively reduce the difference in the distribution of interlayer samples.

The proposed ICMC method is described in detail as follows.

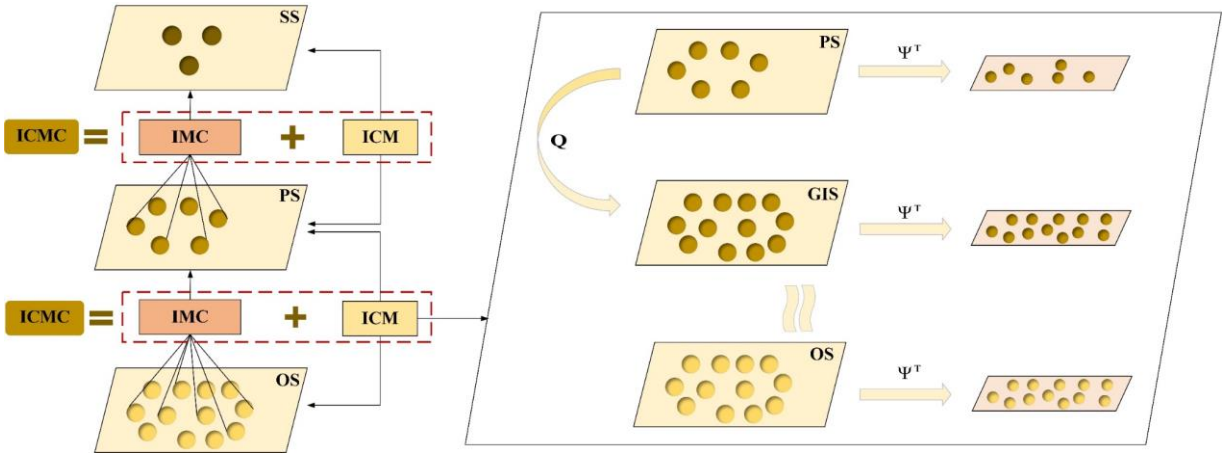


Fig. 5 Flowchart of the proposed ICMC. ICMC consists of IMC and ICM. IMCs are used to construct the multilayer sample spaces, and ICM is designed to effectively reduce the distribution difference of interlayer samples. In ICM, PS is used to generate GIS, and its sample distribution is similar to OS. GIS generation is carried out by the generation matrix $\mathbf{Q}$ in an unsupervised manner.

Let $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_u]$ be the u cluster centers. The formula of IMC is as follows:

$$\min_{\mathbf{S}} \sum_{P=1}^{u} \sum_{\mathbf{Y} \in \mathbf{S}_P} \left\| \mathbf{Y} - \mathbf{M}_P \right\|^2 \tag{22}$$

where $\mathbf{S}_P$ is the sample set with $\mathbf{M}_u$ as the cluster center. To determine $\mathbf{S}_P$, the following two steps need to be repeated continuously:

Step 1: For each sample pair $\phi(\boldsymbol{x}_i)$, the Euclidean distance between $\phi(\boldsymbol{x}_i)$ and each cluster center is calculated in turn, and $\phi(\boldsymbol{x}_i)$ is divided into the cluster corresponding to the nearest sample center. Let the pth ($1 \le p \le u$) sample cluster generated in the tth clustering process be recorded as $\mathbf{S}_p^{\,t}$. Its allocation result can be expressed as

$$\mathbf{S}_p^{\,t} = \{\phi(\boldsymbol{x}_i) : \min \left\| \phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_p^{\,t}) \right\|^2 \} \tag{23}$$

Step 2: After completing step 1, u sample clusters are obtained, and the sample center of each cluster is recalculated by taking the mean value of all dimensions of all samples in the current cluster.

$$\mathbf{M}_u^{\,t+1} = \frac{1}{card(\mathbf{S}_p^t)} \sum_{\phi(\boldsymbol{x}_i) \in \mathbf{S}_p^t} \phi(\boldsymbol{x}_i) \tag{24}$$

where $card(\cdot)$ is the number of samples in. $\mathbf{S}_p$. The above two steps are repeated until convergence, and the final u sample centers are taken as new samples (a new sample set $\mathbf{X}_{PS} \in \mathbf{R}^{u \times 2d}$).

Based on the above steps, the primary clustering sample space (PS) can be obtained based on the original sample-pair space (OS). By repeating the IMC process, we can obtain a secondary clustering sample space (SS). The multilayer sample spaces constructed by ICMs are shown in Eq. (25). The OS is obtained by the SPC, in which the two neighboring samples are looked upon as a neighboring envelope, and the PS and SS are obtained by the SPC and IMC, in which the clustered samples are looked upon as a similar envelope. Therefore, the three layers of samples (OS, PS, and SS) are called three layers of 'envelope samples' (ES).

$$\text{Multilayer sample spaces} = \begin{cases} \mathbf{OS},\ \mathbf{X}_{OS} \in \mathbf{R}^{n \times 2d} \\ \mathbf{PS},\ \mathbf{X}_{PS} \in \mathbf{R}^{u \times 2d} \\ \mathbf{SS},\ \mathbf{X}_{SS} \in \mathbf{R}^{e \times 2d} \end{cases} \tag{25}$$

To ensure the consistency of the sample distribution among multilayer sample spaces, we design an ICM between neighboring sample spaces. Before doing so, we need to transpose the matrix in each sample space (i.e., $\mathbf{X}_{OS} \in \mathbf{R}^{2d \times n}$, $\mathbf{X}_{PS} \in \mathbf{R}^{2d \times u}$) to ensure the subsequent work. As shown in Fig. 5, in ICM, we believe that PS can generate an intermediate sample space (GIS) with a similar

distribution to OS by the generative transfer matrix $\mathbf{Q} \in \mathbf{R}^{u \times n}$. Our main purpose is to align the PS and OS sample distributions through GIS.

We use the implicit but generic transformation φ to represent the training set of PS, OS and GIS, which are defined as $\varphi(\mathbf{X}_{PS})$, $\varphi(\mathbf{X}_{OS})$ and $\varphi(\mathbf{X}_{GIS})$. For convenience, $\varphi(\mathbf{X}_{GIS}^{i})$ and $\varphi(\mathbf{X}_{OS}^{j})$ are defined as a sample in GIS and OS, respectively, considering that $\varphi(\mathbf{X}_{GIS}) = \varphi(\mathbf{X}_{PS})\mathbf{Q}$. Therefore, ICM expression is

$$ICM(\mathrm{GIS},\mathrm{OS}) = \frac{1}{n}\sum_{j=1}^{n}\left\|\varphi(\mathbf{X}_{GIS}^{j}) - \varphi(\mathbf{X}_{OS}^{j})\right\|_2^2 + \sum_{i,j}^{n}\mathbf{W}_{\mathrm{ij}}\left\|\varphi(\mathbf{X}_{GIS}^{i}) - \varphi(\mathbf{X}_{OS}^{j})\right\|_2^2 + \left\|\mathbf{Q}\right\|_* \tag{26}$$

Overall, ICM consists of three items. The first is the global distribution difference (GDD) loss, which minimizes the global difference in the edge distribution between GIS and OS. The second item is local distribution difference (LDD) loss, which uses the locality of OS to measure the local area difference from GIS. The third part is LRC regularization, which is performed to maintain the generalization of the generator matrix $\mathbf{Q}$ and can effectively display the global structure of samples in OS and PS. Considering that the nonconvexity of the rank function is NP-hard, this paper uses the nuclear norm $\left\|\mathbf{Q}\right\|_*$ as the rank approximation. The ICM is described as follows.

1) Global distribution difference (GDD)

The first part in Eq. (26) is the GDD loss to minimize the global difference in marginal distributions between the GIS and the OS. We use a linearly transformed projection matrix Ψ to find the potential common subspace of $\varphi(\mathbf{X}_{GIS}^{i})$ and $\varphi(\mathbf{X}_{OS}^{j})$. At the same time, we introduce the generative transfer matrix $\mathbf{Q}$; by substituting it in (26), the GDD loss after projection can be rewritten as

$$\min_{\Psi,\mathbf{Q}} = \frac{1}{n}\left\|\Psi^{T}\varphi(\mathbf{X}_{PS})\mathbf{Q} - \Psi^{T}\varphi(\mathbf{X}_{OS})\mathbf{1}\right\|_2^2 \tag{27}$$

where Ψ can be expressed as some linear combinations, i.e., $\Psi^{T} = \Phi^{T}\varphi(\mathbf{X}_{OP})^{T}$. Φ denotes the linear combination coefficient matrix and $\mathbf{X}_{OP} = [\mathbf{X}_{PS}, \mathbf{X}_{OS}]$, $\varphi(\mathbf{X}_{OP}) \in \mathbf{R}^{2d \times (u+n)}$. Then, the projected PS can be expressed as $\Phi^{\mathbf{T}}\varphi(\mathbf{X}_{OP})^{T}\varphi(\mathbf{X}_{PS})$, and the projected OS can be expressed as $\Phi^{\mathbf{T}}\varphi(\mathbf{X}_{OP})^{T}\varphi(\mathbf{X}_{OS})$. Let $\mathbf{K}_{PS} = \varphi(\mathbf{X}_{OP})^{T}\varphi(\mathbf{X}_{PS})$ and $\mathbf{K}_{OS} = \varphi(\mathbf{X}_{OP})^{T}\varphi(\mathbf{X}_{OS})$; the PS and OS can be expressed simply as $\Phi^{\mathbf{T}}\mathbf{K}_{PS}$ and $\Phi^{\mathbf{T}}\mathbf{K}_{OS}$, respectively. Therefore, the GDD loss is formulated as

$$\begin{aligned}\min_{\Phi,\mathbf{Q}} &= \frac{1}{n}\left\|\Phi^{T}\varphi(\mathbf{X}_{OP})^{T}\varphi(\mathbf{X}_{PS})\mathbf{Q} - \Phi^{T}\varphi(\mathbf{X}_{OP})^{T}\varphi(\mathbf{X}_{OS})\mathbf{1}\right\|_2^2 \\ &= \frac{1}{n}\left\|\Phi^{T}(\mathbf{K}_{PS}\mathbf{Q} - \mathbf{K}_{OS})\mathbf{1}\right\|_2^2\end{aligned} \tag{28}$$

2) Local distribution difference (LDD)

The second part in Eq. (26) is the LDD. The LDD loss exploits the locality of the OS to measure local differences and indirectly enhances the distribution consistency between the GIS and the OS. The LDD loss in Eq. (26) can be rewritten as

$$\begin{aligned}&\sum_{i,j}^{n}\mathbf{W}_{\mathrm{ij}}\left\|\varphi(\mathbf{X}_{GIS}^{i}) - \varphi(\mathbf{X}_{OS}^{j})\right\|_2^2 \\ &= Tr(\varphi(\mathbf{X}_{GIS})\mathbf{D}(\varphi(\mathbf{X}_{GIS})^{T}) + Tr(\varphi(\mathbf{X}_{OS})\mathbf{D}(\varphi(\mathbf{X}_{OS})^{T}) \\ &- 2Tr(\varphi(\mathbf{X}_{GIS})\mathbf{W}(\varphi(\mathbf{X}_{OS})^{T})\end{aligned} \tag{29}$$

The matrix $\mathbf{D} \in \mathbf{R}^{n \times n}$ is a diagonal matrix with entries $\mathbf{D}_{ii} = \sum_{j}\mathbf{W}_{ij}, i = 1, \cdots, n$, and $Tr(\cdot)$ is the trace of a matrix. $\mathbf{W}_{ij} \in \mathbf{R}^{n \times n}$ is the affinity matrix, described as

$$\mathbf{W}_{\mathrm{ij}} = \begin{cases}1, & \text{if } \mathbf{X}_{GIS}^{i} \in N_z(\mathbf{X}_{OS}^{j}) \parallel \mathbf{X}_{OS}^{j} \in N_z(\mathbf{X}_{GIPS}^{i}) \\ 0, & \text{otherwise}\end{cases} \tag{30}$$

where $N_z(\mathbf{X})$ represents the zth nearest neighbors of sample $\mathbf{X}$. It is the same as GDD. By substituting $\varphi(\mathbf{X}_{GIS}) = \varphi(\mathbf{X}_{PS})\mathbf{Q}$ and using the linearly transformed projection matrix Ψ to find the potential common subspace of $\varphi(\mathbf{X}_{GIS}^{i})$ and $\varphi(\mathbf{X}_{OS}^{j})$, the LDD loss can be rewritten as

$$\begin{aligned}\min_{\Phi,\mathbf{Q}} = \frac{1}{n^2}[&Tr(\Phi^T\mathbf{K}_{PS}\mathbf{Q}\mathbf{D}(\Phi^T\mathbf{K}_{PS}\mathbf{Q})^T)\\&+Tr(\Phi^T\mathbf{K}_{OS}\mathbf{D}(\Phi^T\mathbf{K}_{OS})^T)\\&-2Tr(\Phi^T\mathbf{K}_{PS}\mathbf{Q}\mathbf{W}(\Phi^T\mathbf{K}_{OS})^T)]\end{aligned} \tag{31}$$

3) ICM based on a combination of GDD and LDD

Finally, combining the above three points and introducing the trade-off parameters ∂ and γ at the same time, the objective of our ICM can be rewritten as

$$\begin{aligned}\min_{\Phi,\mathbf{Q}}\ &\partial\frac{1}{n}\left\|\Phi^T(\mathbf{K}_{PS}\mathbf{Q}-\mathbf{K}_{OS})\mathbf{1}\right\|_2^2+\frac{1}{n^2}[Tr(\Phi^T\mathbf{K}_{PS}\mathbf{Q}\mathbf{D}(\Phi^T\mathbf{K}_{PS}\mathbf{Q})^T)\\&+Tr(\Phi^T\mathbf{K}_{OS}\mathbf{D}(\Phi^T\mathbf{K}_{OS})^T)-2Tr(\Phi^T\mathbf{K}_{PS}\mathbf{Q}\mathbf{W}(\Phi^T\mathbf{K}_{OS})^T)]+\gamma\left\|\mathbf{Q}\right\|_*\\ s.t.&\ \Phi^T\mathbf{K}\Phi=\mathbf{I}\end{aligned} \tag{32}$$

The rows of Ψ must be orthogonal and normalized to the unit norm to prevent trivial solutions by enforcing $\Psi^T\Psi=\mathbf{I}$, which can be further rewritten as $\Phi^T\mathbf{K}\Phi=\mathbf{I}$, an equality constraint.

4) Optimization of ICM

Obviously, there are two variables $\mathbf{\Phi}$ and $\mathbf{Q}$ to solve in ICM. Generally, it can be effectively solved by the alternating directional method of multipliers (ADMM) [49]. By introducing auxiliary variable $\mathbf{\Theta}$, furthermore, using the augmented Lagrangian method (ALM) [50], ICM can be rewritten as

$$\begin{aligned}\min_{\Phi,\mathbf{Q},\mathbf{\Theta}}\ &\partial\frac{1}{n^2}\Phi^T\Phi(\mathbf{K}_{PS}\mathbf{Q}\mathbf{1}(\mathbf{K}_{PS}\mathbf{Q})^T-\mathbf{K}_{PS}\mathbf{Q}\mathbf{1}(\mathbf{K}_{PS})^T-\mathbf{K}_{OS}\mathbf{1}\mathbf{Q}^T(\mathbf{K}_{PS})^T)\\&+\mathbf{K}_{OS}\mathbf{1}(\mathbf{K}_{OS})^T)+\frac{1}{n^2}(Tr(\Phi^T\mathbf{K}_{PS}\mathbf{Q}\mathbf{D}(\Phi^T\mathbf{K}_{PS}\mathbf{Q})^T)+Tr(\Phi^T\mathbf{K}_{OS}\mathbf{D}(\Phi^T\mathbf{K}_{OS})^T)\\&-2Tr(\Phi^T\mathbf{K}_{PS}\mathbf{Q}\mathbf{W}(\Phi^T\mathbf{K}_{OS})^T))+\gamma\left\|\mathbf{\Theta}\right\|_*+Tr(\mathbf{E}^T(\mathbf{Q}-\mathbf{\Theta}))+\frac{\rho 1}{2}(\left\|\mathbf{Q}-\mathbf{\Theta}\right\|_F^2)\end{aligned} \tag{33}$$

where $\mathbf{E}$ and $\rho 1$ are the augmented Lagrangian multipliers and penalty parameters, respectively, and $\mathbf{1}$ represents a full one matrix. Then, the variables $\mathbf{\Phi}$, $\mathbf{Q}$ and $\mathbf{\Theta}$ in ICM can be alternatively solved as follows.

Step 1): Update for $\mathbf{\Phi}$: When $\mathbf{Q}$ and $\mathbf{\Theta}$ are fixed, the objective function in (33) is converted into

$$\begin{aligned}\min_{\Phi}\ &\frac{\partial}{n^2}\Phi^T(\mathbf{K}_{PS}\mathbf{Q}\mathbf{1}(\mathbf{K}_{PS}\mathbf{Q})^T-\mathbf{K}_{PS}\mathbf{Q}\mathbf{1}(\mathbf{K}_{PS})^T-\mathbf{K}_{OS}\mathbf{1}\mathbf{Q}^T(\mathbf{K}_{PS})^T)\\&+\mathbf{K}_{OS}\mathbf{1}(\mathbf{K}_{OS})^T)\Phi+\frac{1}{n^2}[Tr(\Phi^T\mathbf{K}_{PS}\mathbf{Q}\mathbf{D}(\Phi^T\mathbf{K}_{PS}\mathbf{Q})^T)\\&+Tr(\Phi^T\mathbf{K}_{OS}\mathbf{D}(\Phi^T\mathbf{K}_{OS})^T)-2Tr(\Phi^T\mathbf{K}_{PS}\mathbf{Q}\mathbf{W}(\Phi^T\mathbf{K}_{OS})^T)]\\ \text{s.t.}\ &\ \Phi^TK\Phi=\mathbf{I}\end{aligned} \tag{34}$$

we can derive the solution Φ_j of the *j*th iterations column wise. To obtain the *i*th column vector in Φ_j by setting the partial derivative of Eq. (34) with respect to $\Phi_{j(:,i)}$ to be zero yields

$$\begin{aligned}&\frac{1}{n^2}Tr(\Phi^T\mathbf{K}_{PS}\mathbf{Q}\mathbf{D}(\Phi^T\mathbf{K}_{PS}\mathbf{Q})^T)+\mathbf{K}_{OS}\mathbf{D}(\mathbf{K}_{OS})^T-\mathbf{K}_{PS}\mathbf{Q}\mathbf{W}(\mathbf{K}_{OS})^T\\&-\mathbf{K}_{OS}\mathbf{W}\mathbf{Q}^T(K_{PS})^T)\Phi_{j(:,i)}+\frac{\partial}{n^2}(\mathbf{K}_{PS}\mathbf{Q}\mathbf{1}\mathbf{Q}^T(\mathbf{K}_{PS})^T-K_{PS}\mathbf{Q}\mathbf{1}(\mathbf{K}_{OS})^T\\&-\mathbf{K}_{OS}\mathbf{1}\mathbf{Q}^T(\mathbf{K}_{PS})^T+\mathbf{K}_{OS}\mathbf{1}(\mathbf{K}_{OS})^T)\Phi_{j(:,i)}\\&=-\lambda\mathbf{K}\Phi_{j(:,i)}\end{aligned} \tag{35}$$

Step 2): Update for $\mathbf{Q}$: When $\mathbf{\Phi}$ and $\mathbf{\Theta}$ are fixed, the objective function in (33) is converted into

$$\begin{aligned}\min_{\mathbf{Q}} \quad & \frac{\partial}{n^2}\Phi^T(\mathbf{K}_{PS}\mathbf{Q}\mathbf{1}(\mathbf{K}_{PS}\mathbf{Q})^T - \mathbf{K}_{PS}\mathbf{Q}\mathbf{1}(\mathbf{K}_{PS})^T - \mathbf{K}_{OS}\mathbf{1}\mathbf{Q}^T(\mathbf{K}_{PS})^T))\Phi \\ & + \frac{1}{n^2}(Tr(\Phi^T\mathbf{K}_{PS}\mathbf{Q}\mathbf{D}(\Phi^T\mathbf{K}_{PS}\mathbf{Q})^T) - 2Tr(\Phi^T\mathbf{K}_{PS}\mathbf{Q}\mathbf{W}(\Phi^T\mathbf{K}_{OS})^T)) \\ & + Tr(\mathbf{E}^T(\mathbf{Q}-\mathbf{\Theta})) + \frac{\rho 1}{2}(\|\mathbf{Q}-\mathbf{\Theta}\|_F^2)\end{aligned} \tag{36}$$

we can use the gradient descent algorithm [51] to solve the closed-form solution of $\mathbf{Q}$, and after *(j+1)*th iterations, $\mathbf{Q}_{j+1}$ can be updated as

$$\begin{aligned}\mathbf{Q}_{j+1} = \mathbf{Q}_j & - \frac{2}{n^2}((\mathbf{K}_{PS})^T\Phi\Phi^T\mathbf{K}_{PS}\mathbf{Q}\mathbf{D} - (\mathbf{K}_{PS})^T\Phi\Phi^T(\mathbf{K}_{OS}\mathbf{W}) \\ & + \mathbf{E} + \rho 1(\mathbf{Q}-\mathbf{L}) + \frac{2\partial}{n^2}[(\mathbf{K}_{PS})^T\Phi\Phi^T\mathbf{K}_{PS}\mathbf{Q}\mathbf{1} - (\mathbf{K}_{PS})^T\Phi\Phi^T\mathbf{K}_{OS}\mathbf{1}]\end{aligned} \tag{37}$$

Step 3): Update for $\mathbf{\Theta}$: When $\mathbf{\Phi}$ and $\mathbf{Q}$ are fixed, the objective function in (33) is converted into

$$\min_{\mathbf{\Theta}} \quad \gamma\|\mathbf{\Theta}\|_* + Tr(\mathbf{E}^T(\mathbf{Q}-\mathbf{\Theta})) + \frac{\alpha}{2}(\|\mathbf{Q}-\mathbf{\Theta}\|_F^2) \tag{38}$$

After the *(j+1)*th iteration, $\mathbf{\Theta}_{j+1}$ can be updated as

$$\mathbf{\Theta}_{j+1} = \min_{\mathbf{\Theta}j} \quad \lambda_1 Tr(\mathbf{\Theta}_j{}^T\mathbf{\Theta}_j)^{\frac{1}{2}} + \mathbf{Tr}(\mathbf{E}_{\mathbf{1}j}^T(\mathbf{Q}_j - \mathbf{\Theta}_j)) + \frac{\alpha_j}{2}(\|\mathbf{Q}_j - \mathbf{\Theta}_j\|_F^2) \tag{39}$$

We can use the singular value threshold (SVT) operator [52] to solve $\mathbf{\Theta}$. The specific optimization process of variables $\mathbf{\Theta}$, $\mathbf{\Phi}$ and $\mathbf{Q}$ can be referred to [53].

The pseudocode of Algorithm 1 - the proposed ICMC is described as follows.

Algorithm 1 ICMC

Input: Data matrix $\mathbf{X}_{OS} \in \mathbf{R}^{n\times 2d}$

Procedure:

1. Setting parameters: u

2. Calculate $\mathbf{S}_p$ by Eq. (23)

3. Get $\mathbf{M}_u$ by Eq. (24)

4. Constructing multilayer sample spaces as shown in Eq. (25)

5. Get new transposed data $\mathbf{X}_{OS} \in \mathbf{R}^{2d\times n}$, $\mathbf{X}_{PS} \in \mathbf{R}^{2d\times u}$

6. Compute $\mathbf{X}_{OP} = [\mathbf{X}_{PS}, \mathbf{X}_{OS}]$, $\mathbf{K}_{OS} = \varphi(\mathbf{X}_{OP})^T\varphi(\mathbf{X}_{OS})$, $\mathbf{K}_{PS} = \varphi(\mathbf{X}_{OP})^T\varphi(\mathbf{X}_{PS})$, $\mathbf{K} = \varphi(\mathbf{X}_{OP})^T\varphi(\mathbf{X}_{OP})$

7. Initialize: $\mathbf{Q} = \mathbf{\Theta} = 0$

8. **While** not converge **do**

9. Update $\mathbf{\Phi}$, fix $\mathbf{Q}$ and $\mathbf{\Theta}$ by Eq. (34)-(35)

10. Update $\mathbf{Q}$, fix $\mathbf{\Phi}$ and $\mathbf{\Theta}$ by Eq. (36)-(37)

11. Update $\mathbf{\Theta}$, fix $\mathbf{\Phi}$ and $\mathbf{Q}$ by Eq. (38)-(39)

12. Update the multiplier $\mathbf{E}$ and parameter $\rho 1$:

13. $\mathbf{E} \leftarrow \mathbf{E} + \rho 1(\mathbf{Q}-\mathbf{\Theta})$

14. $\rho 1 : \rho 1 \leftarrow \min(\rho 1 \times 1.01, \max_{\rho 1})$

15. Check convergence

16. **End while**

17. Compute $\mathbf{X}_{PS}{}' = [\mathbf{\Phi}^T\varphi(\mathbf{X}_{OP})^T\varphi(\mathbf{X}_{PS})]^T$

Output: $\mathbf{X}_{PS}' \in \mathbf{R}^{u \times d'}$

The projected PS can be represented as $\mathbf{X}_{PS}' \in \mathbf{R}^{u \times d'}$, where d' denotes sample dimensionality after ICMC. ICMC allows us to maintain the local and global consistency of the sample data structure information in PS and OS to better realize efficient sample transformation.

3.1.2 Embedded Stacked Autoencoder (ESAE)

To mine the complementarity between the samples with deep features and original features, this paper proposes an embedded stacked autoencoder (ESAE). We designed an embedded unit in the SAE structure. The main function of this embedded unit is to embed the input features into the encoder network and consider the input features during training. In the next layer, the input features are still considered. The ESAE model is shown in Fig. 6.

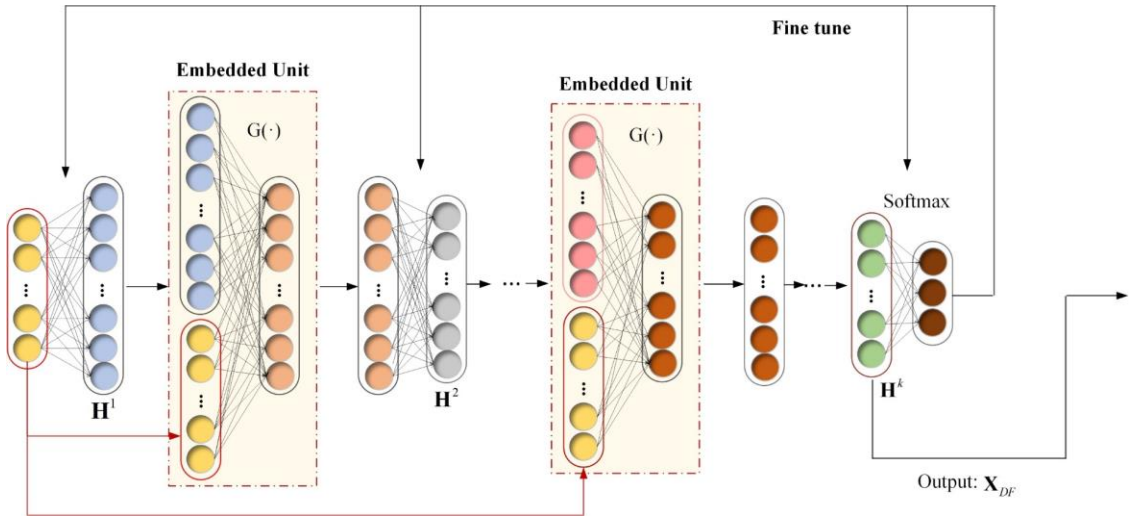


Fig. 6 Structure of the proposed ESAE. The embedded unit introduces the original input feature information into the training process of the encoder network and performs feature transformation through embedding criteria.

As shown in Fig. 6, ESAE consists of *K* encoders cascaded together. Different from the traditional SAE, the key part of the ESAE is the embedded unit between two adjacent hidden layers. Assume that the input data of the ESAE network are $\mathbf{X}_{OS}$, where the output matrix of the hidden layer in the *k*th encoder is $\mathbf{H}^k = [\boldsymbol{h}_1^k, \boldsymbol{h}_2^k, \cdots, \boldsymbol{h}_n^k] \in \mathbf{R}^{n \times d^k}$, 1<*k*<*K*, and d^k represents the number of hidden layer neurons of the *k*th encoder. In ESAE, the first layer of encoder training does not introduce an embedded unit, and its optimization goal is still to minimize the difference between the reconstructed data and the input data $\mathbf{X}_{OS}$. Starting from the second layer of the network, the encoded output of the previous encoder is not directly used as the input data of the next encoder but transformed by the embedded unit. The embedded unit can be expressed as

$$\mathbf{L}(\mathbf{V}) = \mathbf{G}^{\mathrm{T}}(\mathbf{V}) \tag{40}$$

where $\mathbf{V} = \phi(\boldsymbol{x}) \oplus \boldsymbol{h}$, $\oplus$ means to concatenate the original input feature $\phi(\boldsymbol{x})$ and the hidden layer output feature $\boldsymbol{h}$ of the encoder, and $\mathbf{G}$ is the corresponding transformation matrix, consisting of 0's and 1's. After transformation, the hidden layer output vector of the *k*th encoder in the ESAE network can be expressed as

$$\boldsymbol{h}^k = f(\mathbf{W}_{k1}^{\mathrm{T}} \mathbf{L}(\mathbf{V}^{k-1}) + \mathbf{b}_{k1}) \tag{41}$$

where $\mathbf{W}_{k1}$ and $\mathbf{b}_{k1}$ are the connection weight matrix and bias vector between the input layer and the hidden layer in the *k*th AE, respectively. f represents the activation function. The activation function of the encoder in this paper adopts the sigmoid function, which is $f(x) = 1/(1+\exp(-x))$. The purpose of the embedding unit is to introduce certain original information constraints in the layer-by-layer training process of the AE network and filter some hidden layer outputs with weak class representation capabilities. Therefore, the objective function of the proposed embedded unit is expressed as

$$\max_{\mathbf{G}} \ \mathbf{Tr}(\mathbf{G}^T\mathbf{V}\mathbf{V}^T\mathbf{G}) \quad s.t. \ \ \sum \mathbf{G}_{ij} = 2d \tag{42}$$

We then calculate the covariance matrix $\mathbf{D}$ of the feature matrix $\mathbf{V}$, sort its diagonal elements in descending order, and take the first $2d$ values to form a vector $s \in \mathbf{R}^{2d}$. The transformation matrix $\mathbf{G}$ is obtained according to the following formula:

$$\mathbf{G}_{ij} = \begin{cases} 1, & \text{if } \mathrm{s}_j = \mathbf{D}_{ii} \\ 0, & \text{otherwise} \end{cases} \tag{43}$$

where $\mathbf{D}_{ii}$ is the ith diagonal element of the covariance matrix $\mathbf{D}$, s_j is the jth element of the vector, and $i \in [1, n+2d]$, $j \in [1, 2d]$. By combining Eq. (40)-(43), we can obtain the hidden layer output of the kth encoder. Then, the decoding function of the kth encoder can be rewritten as

$$\mathbf{L}'(\mathbf{V}^{k-1}) = f(\mathbf{W}_{k2}^T \boldsymbol{h}^k + \mathbf{b}_{k2}) \tag{44}$$

where $\mathbf{L}'(\mathbf{V}^{k-1})$ is the data obtained by reconstructing the output of the embedded unit. $\mathbf{W}_{k2}$ and $\mathbf{b}_{k2}$ are the connection weight matrix and bias vector between the hidden and output layers in the kth encoder, respectively.

To achieve an "overcomplete" nonlinear mapping of the input vector, a sparse criterion can be imposed in the hidden layers of ESAE. We introduce KL divergence to enable the hidden layer of the encoder to learn the sparse representation. The formula for KL divergence is expressed as follows:

$$\begin{aligned} \mathrm{KL}(\rho \| \rho_j) &= \sum_{j=1}^{2d} \left(\rho \log(\frac{\rho}{\rho_j}) + (1-\rho)\log(\frac{1-\rho}{1-\rho_j}) \right) \\ \rho_j &= \frac{1}{2d} \sum_{1}^{2d} f^j(x^i) \end{aligned} \tag{45}$$

where $f^j(x^i)$ is the activation value of the ith input vector on the jth neuron in the hidden layer.

After introducing the embedded unit in the structure and introducing the sparse criterion in the training process, the optimization objective function of the kth encoder of ESAE is written as

$$\arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \left\| \mathbf{L}(\mathbf{V}^{k-1}) - \mathbf{L}'(\mathbf{V}^{k-1}) \right\|^2 + \lambda(\left\| \mathbf{W}_{k1} \right\|_2 + \left\| \mathbf{W}_{k2} \right\|_2) + \beta(\sum_{j=1}^{d^k} \mathrm{KL}(\rho \| \rho_j)) \tag{46}$$

where λ and β are the penalty parameters of the regularization item and the sparse criterion item, respectively. The training with Eq. (46) as the objective function is the pretraining stage of each layer in the ESAE network. After that, the pretrained hidden layers are cascaded to form a deep network model, and the model parameters obtained from the pretraining are used. The pretraining stage is an unsupervised learning process. To obtain features with stronger classification ability, discriminative information is introduced into the network; that is, a softmax layer is connected to the hidden layer of the last encoder as a classification layer, and the classification layer is supervised.

The proposed ESAE not only takes advantage of the aspect that deep encoder networks can automatically learn latent relationships between data but also improves by introducing initial information and category information into network training. This improves the robustness of deep features.

After two stages of network pretraining and fine-tuning, for the ith input vector $\mathbf{X}_{iOS} = [\phi(\boldsymbol{x}_{i1}), \phi(\boldsymbol{x}_{i2}), \cdots, \phi(\boldsymbol{x}_{i2d})]$, each hidden layer in the network outputs a new feature vector, representing different levels of information. We take the output of the last hidden layer as the deep feature learned by the network, which can be expressed as follows:

$$\mathbf{X}_{DF} = [\boldsymbol{x}_1', \boldsymbol{x}_2', \cdots, \boldsymbol{x}_d'] \in \mathbf{R}^{n \times q} \tag{47}$$

where q is the number of neurons in the last hidden layer. The ESAE is shown in Algorithm 2.

Algorithm 2 ESAE

Input: Data matrix $\mathbf{X}_{OS}$

Procedure:

1. Setting parameters: λ, β, ρ and the number of neurons in each hidden layer, the number of iterations of network training, etc.

2. **Pretraining:**

3. Training the first layer of ESAE, the loss function is Eq. (4), and the output of the hidden layer is recorded as $\mathbf{H}^1$

4. **For** $k = 2,3,\cdots,K$ **do**

5. Calculate $\mathbf{G}$ according to Eq. (42)-(43)

6. Embed the hidden layer output $\mathbf{H}^{k-1}$ of the previous encoder according to the original features of Eq. (40)

7. Train the *k*th layer of the network with Eq. (46) as the objective function

8. Obtain the encoder output of the kth hidden layer and denote it as $\mathbf{H}^k$

9. **End for**

10. **End pretraining**

11. Stack the hidden layers and add a softmax layer on top

12. Network Fine-tuning

Output $\mathbf{X}_{DF}$

The construction method of the training data and test data in each layer of the sample space are described as follows. First, in OS, sample pair $\mathbf{X}_{OS}$ is obtained from the original training data $\mathbf{X}$ by training the first SPC. As described in Section 3.1.1. B, two layers of ICMC construct multilayer samples $\mathbf{X}_{OS}$, $\mathbf{X}_{PS}$', and $\mathbf{X}_{SS}$' and obtain the optimal linear combination coefficient matrix $\mathbf{\Phi}$ and $\mathbf{\Phi}_1$. After that, $\mathbf{X}_{PS}$' and $\mathbf{X}_{SS}$' are combined with the second and third SPC to obtain $\text{sample-pair}(\mathbf{X}_{PS})$ and $\text{sample-pair}(\mathbf{X}_{SS})$ in the PS and SS, respectively. Finally, three layers of envelope samples ($\mathbf{X}_{OS}$, $\text{sample-pair}(\mathbf{X}_{PS})$ and $\text{sample-pair}(\mathbf{X}_{SS})$) will be input into each ESAE layer for training.

Then, the performance of NE_ESAE is verified by test data. First, the test data $\mathbf{X}^{Test}$ are input into the first trained SPC to obtain $\mathbf{X}_{OS}^{Test}$. Then, in PS, $\mathbf{X}_{OS}^{Test}$ is input into the first layer of trained ICMC to obtain $\mathbf{X}_{PS}^{Test}$, where $\mathbf{X}_{PS}^{Test} = [\mathbf{\Phi}^T \varphi(\mathbf{X}_{OP})^T \varphi(\mathbf{X}_{OS}^{Test})]^T$. In SS, $\mathbf{X}_{PS}^{Test}$ is input into the second layer of the trained ICMC to obtain $\mathbf{X}_{SS}^{Test} = [\mathbf{\Phi}_1{}^T \varphi(\mathbf{X}_{SP})^T \varphi(\mathbf{X}_{PS}^{Test})]^T$, where $\mathbf{X}_{SP} = [\mathbf{X}_{SS}, \mathbf{X}_{PS}]$. Then, $\mathbf{X}_{PS}^{Test}$ and $\mathbf{X}_{SS}^{Test}$ are input into the second and third trained SPC to obtain $\text{sample-pair}(\mathbf{X}_{PS}^{Test})$ and $\text{sample-pair}(\mathbf{X}_{SS}^{Test})$ in the PS and SS, respectively. Finally, three layers of envelope samples ($\mathbf{X}_{OS}^{Test}$, $\text{sample-pair}(\mathbf{X}_{PS}^{Test})$ and $\text{sample-pair}(\mathbf{X}_{SS}^{Test})$) will be input into each layer of the trained ESAE to extract deep mixed features F1, F2 and F3 and to obtain classification results. The results from three layers of envelope samples ($\mathbf{X}_{OS}^{Test}$, $\text{sample-pair}(\mathbf{X}_{PS}^{Test})$ and $\text{sample-pair}(\mathbf{X}_{SS}^{Test})$) are fused by the MSEM in Section 3.2.1 to obtain the final results.

The mixed feature vector obtained by Eq. (47) has richer category information. However, simple feature merging will lead to high-dimensional problems and greatly increase the computational complexity of the classifier,. Hence we consider feature reduction (FR) to obtain $\mathbf{X}_{PF}$.

3.2 Classification algorithm based on NE_ESAE

3.2.1 Multilayer Space Ensemble Mechanism (MSEM)

To improve the complementarity of the multilayer sample space, this paper proposes the multilayer space ensemble mechanism (MSEM). MSEM fuses the results under each layer of the sample space to achieve the final result. This MSEM will involve two commonly used ensemble mechanisms: weighted fusion (WF) and majority voting (MV). By NSELM, NE_ESAE contains three layers of sample spaces, which are OS, PS and SS, and which contain structured samples (called envelope samples). Suppose that

the prediction result of the model in the OS is y_0, the prediction result of the model in the PS is y_1, and the prediction result of the model in the SS is y_2. The realization process of the multilayer space ensemble mechanism is as follows.

A. Weighted fusion (WF)

The key to WF is to determine the weight $\mathbf{w}_i$ of the classifier model in each sample space, which requires the use of a validation set. Assuming that the prediction result of each encoder on the validation set is $\mathbf{r} = (\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2)$, the optimal weight vector is $\mathbf{w}_f = (\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2)$, and the optimal weight vector can be obtained by the following formula:

$$\begin{aligned} &\arg\max_{\mathbf{w}_f} = \sum_{i=0}^{\Lambda_v} \eta(round(\mathbf{w}_i{}^T \mathbf{r}_i), y_i) \\ &s.t. \mathbf{w}^T \mathbf{I} = 1, \mathbf{I} = (1,1,...,1) \end{aligned} \tag{48}$$

where Λ_v is the number of validation sets, $\eta(a,b) = \begin{cases} 1, a = b \\ 0, a \neq b \end{cases}$, and with $y = (y_0, y_1, y_2)$. The prediction result of the final test sample can be expressed as

$$y_F = round(\mathbf{w}_f^T y) \tag{49}$$

B. Majority vote (MV)

Assuming that the dataset has a total of C categories, for each prediction sample, a statistical vector $count$ is given, and the initial value is

$$count = (n_1, \cdots, n_j, ..., n_C), n_j = 0 \Big|_{j=1}^{C} \tag{50}$$

The prediction result of the classifier model under each sample space is $y_i \begin{cases} n \\ i = 0 \end{cases}$; then,

$$n_j = \Big|_{j=1}^{C} = \begin{cases} n_j + 1, & \text{if } y_i = j \\ n_j, & \text{others} \end{cases}, i = 1, \cdots, n \tag{51}$$

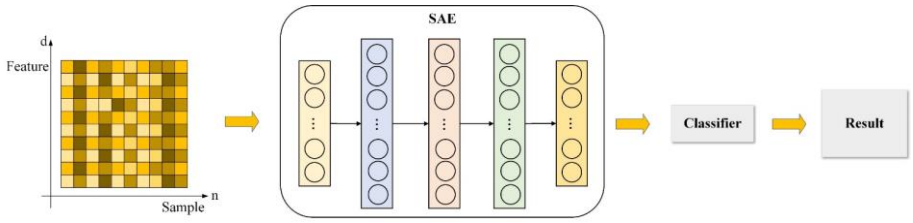
The final classification label y_F is the subscript j corresponding to the largest component in $count$, namely,

$$y_F = j \tag{52}$$

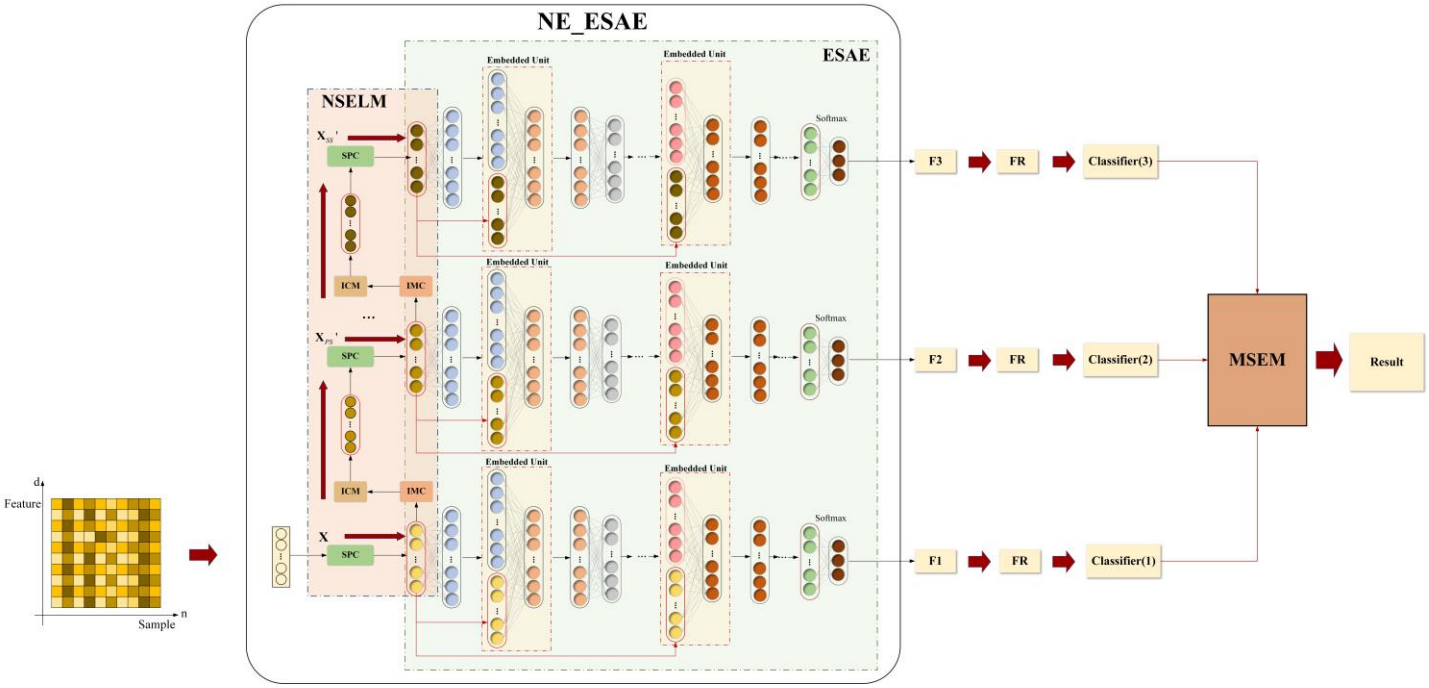
where $n_j = \max(count)$.

3.2.2 Classification algorithm based on NE_ESAE

The traditional classification algorithm based on SAE is shown in Fig. 7 (a), which directly inputs the deep features learned by SAE into the classifier to obtain the final results. In contrast, our NE_ESAE outputs three layers of deep features F1, F2 and F3 by constructing a multilayer of envelope samples. To improve the complementarity among multilayer features, we combine NE_ESAE with MSEM to construct a classification algorithm based on NE_ESAE (Fig. 7 (b)). As shown in the figure, MSEM fuses the classification results from the three layers of ESAEs to realize the sample-feature cooperative transformation.



(a) The classification algorithm based on SAE



(b) The classification algorithm based on NE_ESAE

Fig. 7. The classification algorithm based on traditional SAE and proposed NE_ESAE.

The classification algorithm based on NE_ESAE proposed in this paper is summarized in Algorithm 3.

Algorithm 3 NE_ESAE

Input: Data matrix $\mathbf{X}$

Procedure:

1. Get $\mathbf{X}_{OS}$ according to SPC method

2. Construct multilayer sample spaces OS, PS, SS based on IMCs

3. Construct a new sample $\mathbf{X}_{PS}'$, $\mathbf{X}_{SS}'$ based on ICM

4. Get sample-pair $(\mathbf{X}_{PS})$, sample-pair $(\mathbf{X}_{SS})$ according to SPC method

5. **For** $i = 0,1,2$ **do**

6. Train ESAE network in multilayer sample space and extract multilayer deep features

7. Get the $\mathbf{X}_{DF}$ in the current sample space

8. Obtain multilayer deep features F1, F2 and F3.

9. **End for**

10. Obtain the new data matrix $\mathbf{X}_{PF}$ according to FR

11. Fusing classification results from multilayer sample spaces

12. Determine $\mathbf{w}_f$ based on the validation set and Eq. (48), and obtain the final label according to Eq. (49)

13. Calculate the statistical vector *count* according to Eq. (50), and obtain the final label according to Eq. (52)

Output: Predicted label y_F

4. Experimental results and analysis

To verify the effectiveness of the proposed NE_ESAE, four groups of experiments were conducted. The first group of experiments based on an ablation study was conducted to verify the effectiveness of SPC and ICMC, the effectiveness of MSEM, and the effectiveness of NSELM and ESAE. The second group of experiments compares the proposed algorithm with existing representative feature learning algorithms and representative SAE models. The third group of experiments analyzes the effects of some parameters, including the type of classifier and the proportion of cluster center samples. The fourth group of experiments involves the confusion matrix.

4.1 Experimental conditions

The performance of the proposed algorithm was tested on 12 representative datasets. These datasets have both large and small samples, binary and multiple class labels, and high-, medium- and low-dimensional features. The main information concerning all datasets is shown in Table 3.

Table 3 Basic information of the datasets used in the study

Dataset	Instances	Attributes	Class	Relevant paper
Alzheimer's disease (AD)	90	32	3	Reference [54]
LSVT Voice Rehabilitation Dataset (LSVT)	126	310	2	Reference [55]
Parkinson Speech Dataset (PD)	1040	26	2	Reference [56]
Pen-Based Recognition of Handwritten Digits（Pendigits）	10992	16	10	Reference [57]
Statlog_Landsat_Satellite （Statlog）	6435	36	6	Reference [58]
Statlog_Vehicle Silhouettes (Vehicle)	846	18	4	Reference [58]
Statlog Heart Dataset (heart)	270	13	2	Reference [58]
Maxlittle Parkinson Dataset (Maxlittle)	195	22	2	Reference [59]
Urban land cover (Urban)	675	147	9	Reference [60]
Breast Cancer Wisconsin Diagnostic (WDBC)	569	30	2	Reference [61]
Breast Cancer Wisconsin Original (Wisconsin)	683	9	2	Reference [62]
Pima Indians Diabetes Dataset (PID)	768	8	2	Reference [63]

In the following experiments, the number of layers of the encoder in the proposed NE_ESAE network was set to 3 for all experiments since the sample size was not large for some datasets; the number of hidden layer neurons of each encoder was determined according to the number of samples and feature dimensions of the dataset, and the optimal structure was determined by the grid search method.

The experiment adopted the hold-out cross-validation method, and each dataset was divided into three equal parts. A total of 1/3 of the samples were for the training set, 1/3 of the samples were for the validation set, and 1/3 of the samples were for the test set. The classifier used in the experiment was a support vector machine (SVM) for fair comparison. To eliminate the influence of randomness, each experiment was repeated five times to obtain the average and variance (stand deviation) to characterize the classification effect of the algorithm in this paper.

4.2 Ablation study

4.2.1 Effectiveness analysis of SPC

After the SPC method, we obtained a transformed sample pair. To verify the effectiveness of the sample pair (or SPC), we used the ablation method to compare the classification result of the original sample and the sample pair. Considering that the original features directly used for classification may have some problems that affect the experimental results, such as feature redundancy, feature anomaly, and large feature dimension, we designed three sets of experiments. They were experiments that only used original features (OF), experiments that used features learned by PCA (OF&PCA), and experiments that used features learned by LDA (OF&LDA). The experimental results are listed in Table 4.

Table 4 Comparison of the original sample and sample pair

Dataset		OF (%)	OF&PCA (%)	OF&LDA (%)
AD	Original sample	54.00±9.55	60.00±8.50	62.67±4.94
	Sample-pair	**64.67±4.47**	**70.67±4.35**	**72.00±6.91**
LSVT	Original sample	80.48±6.39	90.48±3.37	87.14±3.61
	Sample-pair	**94.29±3.98**	**96.67±1.30**	**95.71±3.91**
PD	Original sample	62.70±1.86	64.94±1.95	64.20±1.74
	Sample-pair	**70.75±1.74**	**71.90±1.06**	**73.22±1.67**
Pendigits	Original sample	**98.13±0.05**	**98.07±0.13**	**97.87±0.23**
	Sample-pair	66.36±0.98	68.56±2.40	64.63±1.84
Statlog	Original sample	**86.13±0.53**	**87.23±0.54**	**87.09±0.63**
	Sample-pair	82.07±1.10	85.85±0.36	82.89±1.07
Vehicle	Original sample	80.35±1.31	82.34±1.05	82.55±0.92
	Sample-pair	**83.90±0.19**	**85.32±0.78**	**85.53±1.05**
heart	Original sample	80.89±4.26	85.11±3.30	83.33±3.60
	Sample-pair	**85.56±2.83**	**90.67±2.02**	**90.67±2.30**
Maxlittle	Original sample	85.54±4.01	88.00±2.28	88.62±4.16
	Sample-pair	**86.77±2.57**	**91.08±3.15**	**90.78±1.09**
Urban	Original sample	**79.91±3.87**	**82.40±2.80**	**83.38±2.10**
	Sample-pair	73.42±2.48	75.73±2.87	75.02±2.02
WDBC	Original sample	95.66±1.52	97.88±0.84	97.46±0.69
	Sample-pair	**97.57±1.38**	**98.73±1.16**	**99.58±0.44**
Wisconsin	Original sample	96.30±1.72	97.18±1.19	96.83±1.26
	Sample-pair	**97.18±1.48**	**98.06±0.74**	**97.89±1.26**
PID	Original sample	70.39±2.74	72.34±1.98	75.78±3.49
	Sample-pair	**74.14±4.27**	**80.16±1.78**	**82.34±1.88**

The experimental results in Table 4 show that most of the sample pairs are better than the original sample for most datasets. The result demonstrates that the sample pair can learn more efficient and discriminative feature representations than the original sample. We found that the sample pair showed lower accuracy in both the Pendigits dataset with 10992 samples and the Statlog dataset with 6435 samples. We hypothesize that the possible reason is the high sample size and redundancy in the datasets. Sample pairs performed better in small and medium sample datasets, especially in 90 samples with AD, which improved accuracy by approximately 10% in all cases. This shows that our proposed SPC method has obvious advantages on small and medium sample

size datasets. In summary, these experimental results illustrate that the proposed SPC method can realize effective sample transformation, which is very important for sample classification.

4.2.2 Effectiveness analysis of ICMC

To verify the effectiveness of ICMC, we used the ablation method to compare the classification accuracy between sample-pair and sample-pair after ICMC processing (sample-pair &ICMC). The experimental results are listed in Table 5.

Table 5 Comparison of sample-pair and sample-pair &ICMC

Dataset	Sample-pair (%)	Sample-pair &ICMC (%)
AD	64.67±4.47	**66.00±5.96**
LSVT	94.29±3.98	**95.71±3.91**
PD	70.75±1.74	**70.86±1.63**
Pendigits	66.36±0.98	**69.61±1.62**
Statlog	82.07±1.10	**85.13±0.30**
Vehicle	**83.90±0.19**	77.66±0.43
heart	85.56±2.83	**92.00±3.08**
Maxlittle	86.77±2.57	**87.69±3.61**
Urban	73.42±2.48	**76.49±1.90**
WDBC	97.57±1.38	**99.58±0.24**
Wisconsin	97.18±1.48	**98.24±0.82**
PID	74.14±4.27	**79.14±1.42**

According to the experimental results shown in Table 5, it is obvious that the sample-pair &ICMC outperforms the sample-pair on eleven of all datasets. Among them, the classification accuracy of the heart dataset is improved by 6.44%. This shows that our proposed ICMC mechanism has obvious advantages on most datasets.

4.2.3 Effectiveness analysis of MSEM

This section discusses the effectiveness of the MSEM. Specifically, the ESAE is trained in the OS, PS and SS, and the prediction results of each model are fused to obtain the final classification result. The fusion strategy MSEM adopts the MV method and the WF method.

Table 6 MSEM effectiveness analysis experimental comparison

Dataset	OS (%)	PS (%)	SS (%)	MSEM (%)	
				MV	WF
AD	73.33±6.24	84.67±10.70	83.33±8.50	**90.67±6.41**	88.00±7.30
LSVT	96.19±3.61	90.00±13.08	87.62±12.19	**96.67±3.61**	93.33±4.26
PD	74.43±1.61	72.36±1.68	73.76±6.64	77.76±3.79	**78.74±4.22**
Vehicle	80.08±2.49	83.59±4.96	86.02±3.79	87.66±2.97	**89.53±1.96**
Pendigits	83.74±0.62	67.47±5.11	70.93±6.92	79.61±7.99	**84.21±0.83**
Statlog	88.33±0.18	78.29±4.52	80.73±4.41	86.03±1.05	**87.93±0.93**
Urban	82.02±2.50	54.84±6.66	51.11±5.63	65.69±6.06	**84.29±3.38**
heart	92.52±3.19	94.72±2.46	99.17±1.06	97.78±2.57	**99.44±1.11**
WDBC	97.62±2.68	99.74±0.31	100.00±0.00	99.87±0.26	**100.00±0.00**
Maxlittle	95.08±2.28	91.38±6.21	94.78±4.56	**96.92±1.09**	96.25±5.07

Wisconsin	99.19±1.28	98.68±0.88	99.11±0.87	**99.89±0.21**	98.88±3.07
PID	80.08±2.49	85.31±1.58	86.41±3.80	88.20±2.63	**89.38±1.88**

As seen from Table 6, the classification accuracies of the three different sample spaces are different. This is because the sample structure information of different sample spaces has certain complementarity and difference, which leads to different sample partition boundaries of classifier learning. The prediction results of each model can be fused by MSEM to improve the classification accuracy to a certain extent. As shown in Table 6, the classification performance of all the datasets was significantly improved after the fusion of MSEM. Compared with OS, MSEM improves the performance of all datasets by 17.34%, 0.48%, 4.34%, 9.48%, 0.47%, 1%, 2.27%, 6.92%, 2.38%, 1.84%, 0.7%, and 9.3%, respectively. Compared with the PS, the MSEM improves the performance of all datasets by 6.00%, 6.67%, 6.38%, 5.94%, 16.74%, 9.64%, 29.45%, 1.72%, 0.26%, 5.54%, 1.21% and 4.07%. Compared with the SS, the MSEM improves the performance of all datasets by 7.34%, 9.05%, 4.98%, 3.51%, 13.28%, 7.20%, 33.18%, 0.27%, 2.14%, 0.78% and 2.97%, respectively.

In addition, it can be seen from Table 6 that the improvement rates of the classification results of MV and WF are different in different datasets. For example, the MV method in AD performs 2.67% better than the WF method, while in Vehicle, it performs 1.87% worse than the MF method. We think the possible reason is that the data structures of different datasets are not the same. Combined with the above experimental results, it is proven that the proposed MSEM is effective and can significantly improve the accuracy and stability of the classification model.

4.2.4 Effectiveness analysis of the NSELM and ESAE

To verify the effectiveness of the proposed NSELM and ESAE, we used the ablation method to compare the classification performance of NE_ESAE, NE_ESAE without NSELM (no NSELM), and NE_ESAE without ESAE (no ESAE) under the same network structure. The multilayer sample space in NE_ESAE without NSELM was constructed by the bagging method. The experimental results are shown in Table 7.

Table 7 NSELM and ESAE effectiveness analysis experimental comparison

Dataset	NE_ESAE without NSELM (%)	NE_ESAE without ESAE (%)	NE_ESAE (%)
AD	60.00±3.33	79.33±14.41	**90.67±6.41**
LSVT	80.48±8.81	96.19±5.98	**96.67±3.61**
PD	63.33±4.80	72.82±3.18	**78.74±4.22**
Vehicle	78.58±6.70	80.99±2.85	**89.53±1.96**
Pendigits	**98.51±0.27**	67.40±1.49	84.21±0.83
Statlog	85.32±2.83	83.53±0.63	**87.93±0.93**
Urban	74.29±1.32	74.82±5.66	**84.29±3.38**
heart	84.44±5.88	96.89±2.53	**99.44±1.11**
WDBC	96.17±1.43	99.79±0.29	**100.00±0.00**
Maxlittle	86.15±3.08	89.23±5.76	**96.92±1.09**
Wisconsin	97.35±0.88	99.50±0.48	**99.89±0.21**
PID	73.44±7.14	82.50±2.28	**89.38±1.88**

As we can see from Table 7, having NSELM leads to higher classification accuracy for most datasets. This result means that the proposed NSELM can deeply mine the structural information of samples, thereby effectively improving the classification accuracy. However, we found that the accuracy of the dataset Pendigits is not better in the proposed NSELM. As before, the possible reason is that there are too many samples in the dataset Pendigits (the number of samples in this dataset is 10992). Too many samples will cause data redundancy after ESLEM processing, resulting in poor performance. In contrast, for small sample datasets such as AD, LSVT, Maxlittle, and heart, our method shows clear advantages; at the highest level, the classification accuracy of the AD

dataset is improved by 30.67%. Furthermore, the comparison between NE_ESAE and NE_ESAE without ESAE verifies that the proposed ESAE can effectively enhance the feature extraction ability.

4.3 Algorithm comparison

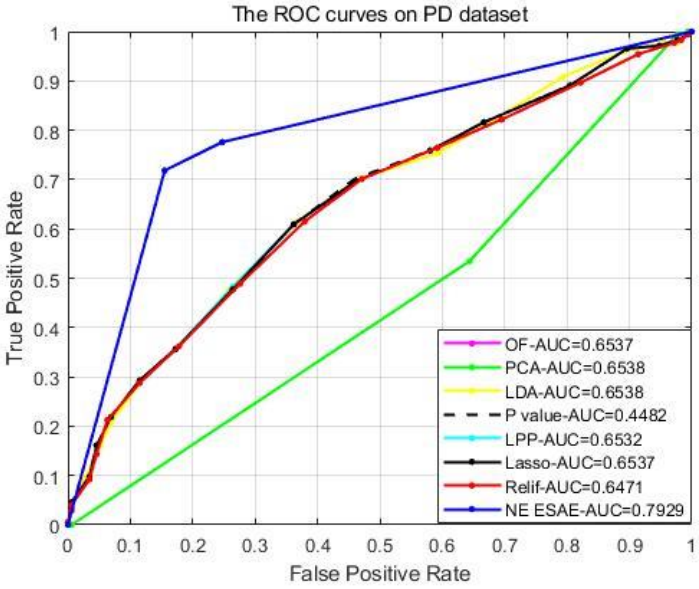
4.3.1 Comparison with classical feature-learning algorithms

To evaluate the performance of the NE_ESAE algorithm, its performance was compared to those representative feature-learning algorithms, including feature selection algorithms (PCA [64], LDA [65], LPP[66], Relief [67], Lasso [68] and P_value [69]). The results are listed in Table 8. In addition to accuracy, receiver operating characteristic (ROC) curves were also obtained in this experiment to better compare the sample recognition ability of the proposed algorithm and the comparison algorithm.

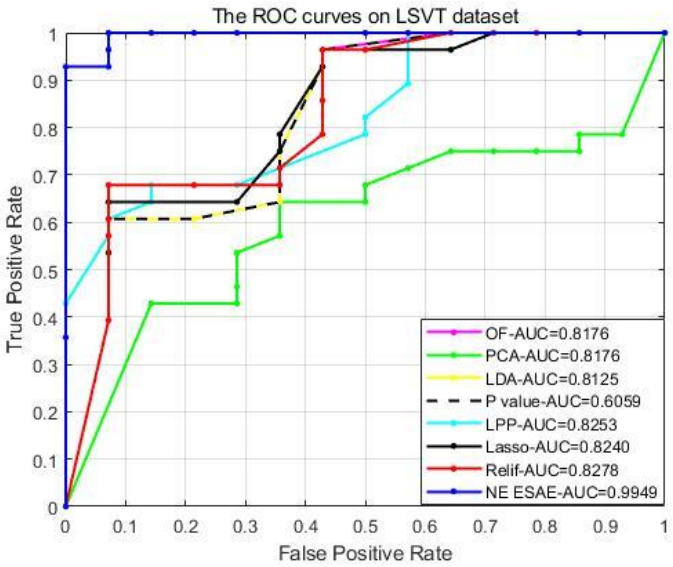
Table 8 Comparison of different feature-learning algorithms

Dataset	OF (%)	PCA (%)	LDA (%)	LPP (%)	Relief (%)	Lasso (%)	P_value (%)	**NE_ESAE** (%)
AD	54.00	60.00	62.67	60.67	54.00	54.00	48.67	**90.67**
	±9.55	±8.50	±4.94	±7.60	±10.90	±9.55	±7.67	**±6.41**
LSVT	80.48	90.48	87.14	90.48	88.57	76.67	75.71	**96.67**
	±6.39	±3.37	±3.61	±3.76	±3.10	±5.16	±5.93	**±3.61**
PD	62.70	64.94	64.20	65.29	64.08	62.70	56.61	**78.74**
	±1.86	±1.95	±1.74	±1.47	±1.13	±1.86	±3.92	**±4.22**
Vehicle	80.35	82.34	82.55	81.77	78.16	80.78	75.39	**89.53**
	±1.31	±1.05	±0.92	±0.89	±0.19	1.05	±0.89	**±1.96**
Pendigits	**98.13**	98.07	97.87	97.97	97.82	98.13	92.87	84.21
	±0.05	±0.13	±0.23	±0.11	±0.23	±0.05	±1.46	±0.83
Statlog	86.13	87.23	87.09	87.27	86.10	86.15	86.13	**87.93**
	±0.53	±0.54	±0.63	±0.48	±0.40	±0.51	±0.68	**±0.93**
Urban	79.91	82.40	83.38	82.93	80.18	79.91	80.44	**84.29**
	±3.87	±2.80	±2.10	±2.53	±2.53	±3.84	±2.67	**±3.38**
heart	78.89	85.33	84.67	84.22	81.78	78.29	62.00	**99.44**
	±3.42	±2.53	±2.14	±3.37	±3.00	±3.42	±4.67	**±1.11**
WDBC	95.66	97.88	97.46	97.35	97.04	95.78	88.78	**100.00**
	±1.52	±0.84	±0.69	±0.53	±1.43	±1.54	±2.19	**±0.00**
Maxlittle	84.62	88.62	88.62	86.77	85.85	85.23	76.62	**96.92**
	±3.92	±2.79	±4.16	±4.43	±5.03	±4.16	±2.75	**±1.09**
Wisconsin	96.30	97.18	96.83	97.00	96.74	96.30	93.66	**99.89**
	±1.72	±1.19	±1.26	±1.18	±1.83	±1.72	±1.67	**±0.21**
PID	70.40	72.34	75.78	69.67	73.29	70.39	73.91	**89.38**
	±2.74	±1.98	±3.49	±5.54	±4.51	±2.74	±3.55	**±1.88**

The experimental results in Table 8 further demonstrate the advantages of the proposed NE_ESAE algorithm. For example, in dataset AD, the accuracy of our algorithm is 28% higher than the best LDA and 42% higher than the worst P_value among other classical feature-learning algorithms. In dataset PD, the accuracy of our algorithm is 13.8% higher than that of PCA and 22.13% higher than that of P_value. The results also show that simultaneous sample learning and feature learning are more effective than only feature learning in the original space. The multilayer sample spaces contain different structural information from the original space, which can provide more potential sample distribution information. In addition, it can be seen from the variance in the table that the proposed algorithm has better stability than other feature learning algorithms in most cases.



(a) Description of the ROC curves on PD



(b) Description of the ROC curves on LSVT

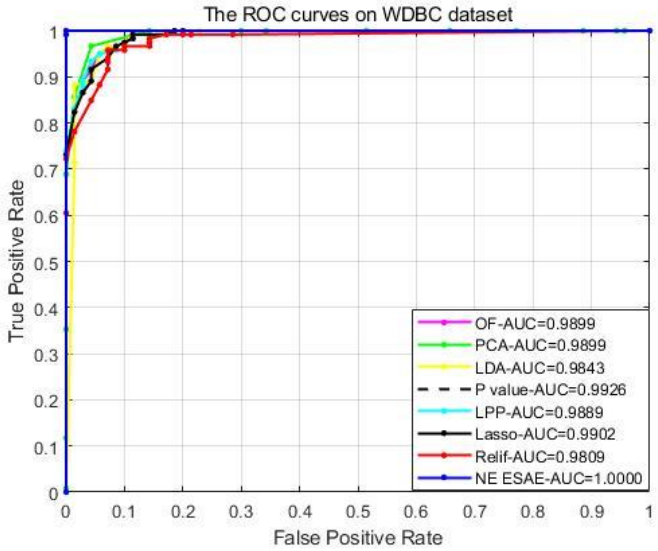


Fig. 8 (c) Description of the ROC curves on WDBC

Fig. 8 (a) to (c) describe the ROC curves of NE_ESAE and the comparison methods on the PD, LSVT, and WDBC datasets. The AUC (area under the curve) is the area enclosed by the ROC curve and the coordinate axes. It can be seen from the figure that the AUC of NE_ESAE is higher than that of the comparison method, which appears that the prediction model of the proposed method is optimal in comparison.

4.3.2 Comparison with the representative stacked autoencoders

To verify the advantage of the proposed NE_ESAE over the existing SAEs, some representative SAEs were considered for comparison. They included the stacked autoencoder (SAE), stacked sparse autoencoder (SSAE) [42], stacked denoising sparse autoencoder (SDSAE) [46], stacked pruning sparse autoencoder (SPSAE) [47], embedded stacked group sparse autoencoder ensemble with L1 regularization and manifold reduction (ESGSAE_FF) [70] and gated stacked target-related autoencoder (GSTAE) [48]. Since most of the SAEs had tested ton he seven datasets, including AD, LSVT, PD, Urban, Vehicle, Pendigits, and Statlog, these datasets were used for comparison here. The experimental results are presented in Table 9.

been

Table 9 Classification accuracy (mean ±variance) of different deep autoencoder classifiers

Dataset	SAE (%)	SSAE (%)	SDSAE (%)	SPSAE (%)	ESGSAE_FF (%)	GSTAE (%)	**NE_ESAE** (%)
AD	50.67±7.95	56.67±5.27	55.58±4.36	57.78±4.27	67.33±2.49	71.11±8.16	**90.67±6.41**
LSVT	83.80±5.16	83.33±5.83	76.62±5.29	84.33±5.36	92.76±0.62	84.66±4.32	**96.67±3.61**
PD	61.15±2.91	64.48±2.05	64.88±1.84	64.22±2.34	66.72±0.87	73.89±4.27	**78.74±4.22**
Urban	74.48±3.33	79.73±0.67	75.17±1.88	77.81±1.17	83.20±1.01	76.98±0.73	**84.29±3.38**
Vehicle	67.30±3.33	70.00±2.99	72.00±2.25	74.76±2.93	81.91±0.42	79.71±2.93	**89.53±1.96**
Pendigits	89.64± 1.44	93.80± 0.51	94.58 ± 0.53	91.60 ± 0.57	**98.00 ±0.12**	93.53±0.77	84.21±0.83
Statlog	83.67± 0.36	84.85± 0.84	83.65± 0.71	85.87± 0.86	87.28±0.12	85.42±0.38	**87.93±0.93**

As Table 9 shows, the proposed NE_ESAE has the best classification accuracy compared with other stacked autoencoders in most cases. For example, the classification accuracy of NE_ESAE in AD dataset is 40.00% higher than SAE, 34.00% higher than SSAE, 35.09% higher than SDSAE, 32.89% higher than SPSAE, 23.34% higher than ESGSAE_FF, and 19.56% higher than GSTAE, respectively. The experimental results prove that it is feasible and effective to consider the sample structure information and introduce the complementary ideas of original features and deep features in pretraining. For datasets with a small number of samples

and high features, such as LSVT and Urban, the NE_ESAE model still shows better performance. One of the possible reasons is that the NSELM considers the latent data structure information among samples so that the neglected intrinsic information among samples can be better learned. The experimental results demonstrate that the proposed NE_ESAE is an efficient new SAE that has better feature learning ability.

4.4 Parametric analysis

4.4.1 Effects of classifier type

The impact of different classifiers (extreme learning machines (ELM), SVM, and random forest (RF)) on the proposed algorithm's performance was experimentally studied. The results are presented in Table 10.

Table 10 Classification accuracy (mean ±variance) of the proposed algorithm with different classifiers

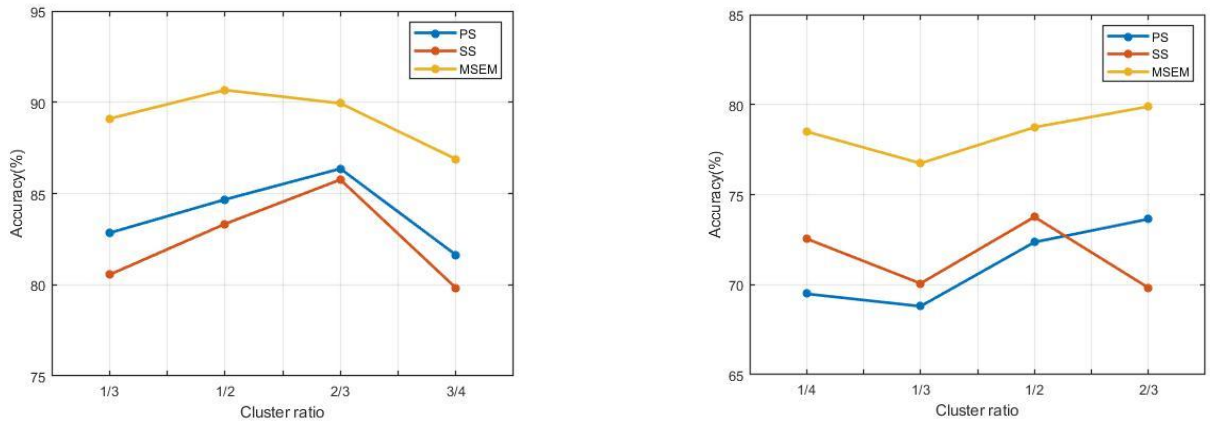
Dataset	SVM (%)	RF (%)	ELM (%)
AD	**90.67±6.41**	88.00±8.69	82.6714.61
LSVT	96.67±3.61	**97.62±2.38**	97.14±3.91
PD	**78.74±4.22**	77.67±5.25	76.67±6.30
Vehicle	**89.53±1.96**	80.00±8.11	80.99±7.34
Pendigits	84.21±0.83	82.78±1.79	**85.17±1.35**
Statlog	**87.93±0.93**	81.61±2.49	82.99±1.93
Urban	**84.29±3.38**	75.18±5.45	82.14±4.60
heart	**99.44±1.11**	98.89±1.28	98.33±2.13
WDBC	**100.00±0.00**	**100.00±0.00**	99.87±0.26
Maxlittle	96.92±1.09	**98.75±1.71**	**98.75±1.71**
Wisconsin	**99.89±0.21**	99.41±0.51	99.71±0.51
PID	89.38±1.88	**89.84±2.47**	89.38±2.25

Table 10 shows that half of the datasets have the highest classification accuracy when SVM is used as the classifier. For example, on the AD dataset, the classification accuracy obtained by using SVM is 90.67%, which is 2.67% and 8.00% higher than those of RF and ELM, respectively, and has better stability. For dataset PD, the classification accuracy of SVM is the highest, with an accuracy of 78.74%, which is 1.07% higher than that of RF and 2.07% higher than that of ELM. For datasets with a large number of samples, such as Pendigits, the ELM algorithm has better performance than SVM and RF. The possible reason is that ELM is based on a feedforward neural network, which requires a certain number of samples to optimize the weights. Hence it can show better performance for large sample datasets. Overall, the classification performance of RF is between that of SVM and ELM, but one disadvantage of RF is that as the number of decision trees increases, its training process requires more time and space. On the other hand, by comparing the variance of the classification accuracy obtained from the twelve experiments in Table 10, it can be found that the variance is always the smallest when SVM is used. Overall, the experimental results show that the proposed algorithm has the best stability when SVM is used as the classifier.

4.4.2 Effects of the cluster ratio

This section studies the influence of the cluster ratio in IMC on the performance of the proposed algorithm. where cluster ratio means the ratio of the number of samples in the sample space before and after clustering (i.e., the ratio of PS to OS and the ratio of

SS to PS). The experiment was carried out on the AD and PD datasets. Considering the different numbers of samples in the AD and PD datasets, the cluster ratio of dataset AD was set as 1/3, 1/2, 2/3, and 3/4, and that of dataset PD was set as 1/4, 1/3, 1/2, and 2/3. The results are presented in Fig. 9. In the figure, the PS curve represents the model's classification accuracy in the PS sample space, the SS curve represents the model's classification accuracy in the SS sample space, and the MSEM curve represents the classification accuracy after fusion of the multilayer sample space by MSEM.



(a)AD (b)PD

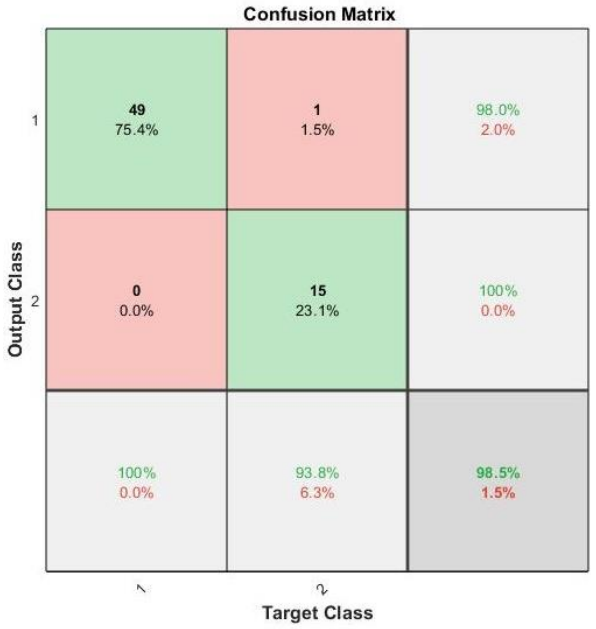
Fig. 9. The effect of the cluster ratio on the performance of the proposed algorithm

Fig. 9 shows that the cluster ratio has a significant impact on the experimental results. According to the MSEM curve, no matter how large is the cluster ratio selected, the classification accuracy after MSEM will be higher than that of PS and SS, indicating that the information contained in each sample space is complementary. As shown in Fig. 9 (a), the classification accuracy of PS and SS increases with increasing cluster ratio at the beginning. When the cluster ratio reaches 2/3, the classification accuracy is the best and then decreases. In addition, the classification accuracy of the MSEM increases with increasing cluster ratio at the beginning. When the cluster ratio reaches 1/2, the classification accuracy reaches the best value and then decreases. By comparing these three curves, we find that although the classification accuracy of PS and SS is the highest when the ratio is 2/3, the classification accuracy of MSEM is lower than that when the ratio is 1/2. We hypothesize that the reason is that the number of samples in the AD is small; when the cluster ratio is too large, more samples in the SS come directly from the PS, resulting in lower differences between the interlayer sample spaces.

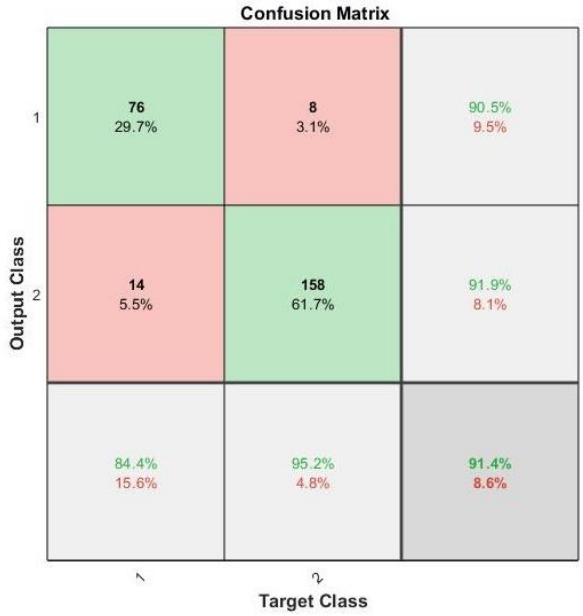
Different from dataset AD, in Fig. 9 (b), at the beginning, the classification accuracy of PS and SS has the same rule as that of MSEM. With the increase in the cluster ratio, the classification accuracy of these three curves decreases. When the cluster ratio reaches 1/3, the classification accuracy of PS and MSEM increases with increasing cluster ratio, and the classification accuracy of SS is the lowest at 2/3. We hypothesize that the reason is that the number of samples in dataset PD is greater than that in dataset AD. Although there are samples in the PS that are directly from the OS, the generated new samples are still sufficient to ensure the difference of each space, so the fusion performance is better. However, the cluster ratio should not be too large; otherwise, it will increase the time complexity of the algorithm. In summary, we find that the algorithm performs best when the cluster of the dataset is 1/2, which makes the algorithm have high classification accuracy but does not increase the algorithm complexity.

4.5 Confusion matrix

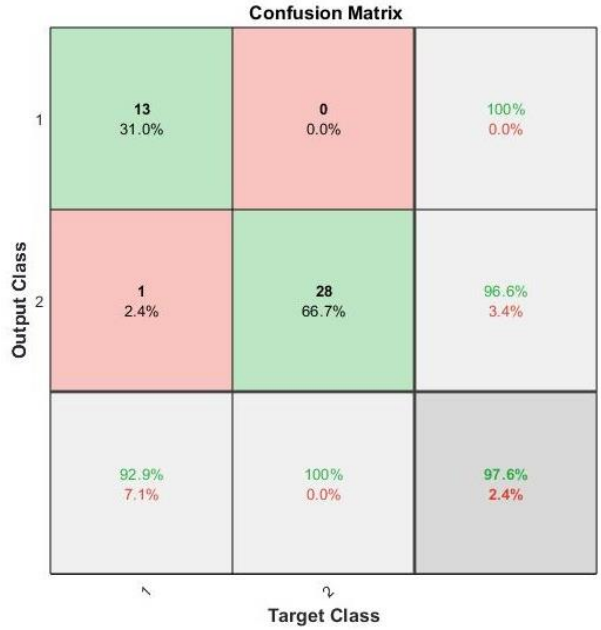
To better show the performance of the proposed NE_ESAE model, we introduced the confusion matrix. We selected six datasets with apparent imbalanced distributions: Maxlittle (size of each class: 48 and 147), PID (size of each class: 500 and 268), LSVT (sizes per class: 42 and 84), Statlog (sizes per class: 1533, 703, 1358, 626, 707 and 1508), Urban (sizes per class: 36, 116, 106, 122, 59, 112, 61, 34 and 29) and Wisconsin (sizes per class: 444 and 239). To better evaluate the classification performance of the proposed algorithm and to more intuitively reflect the prediction results for each class, the confusion matrix in each dataset is shown in Fig. 10.



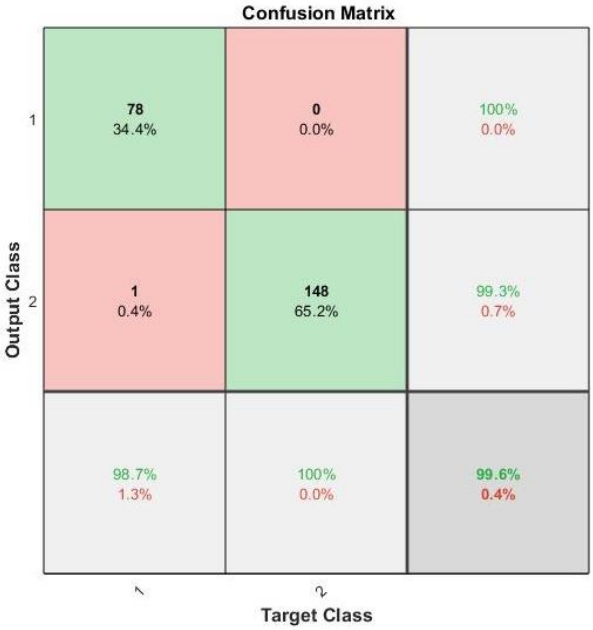
(a) Maxlittle



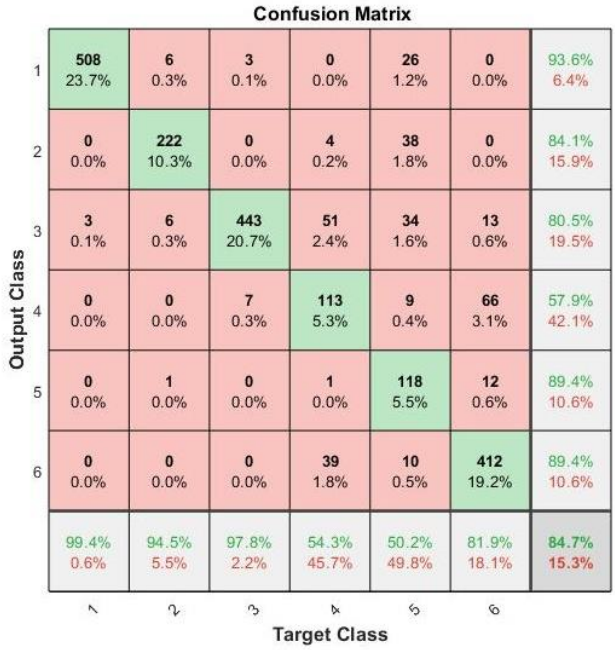
(b) PID



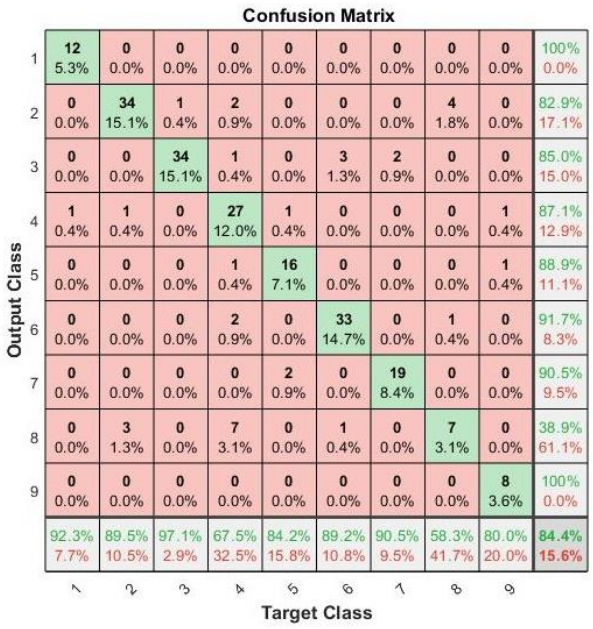
(c) LSVT



(d) Wisconsin



(e) Statlog



(f) Urban

Fig. 10. The confusion matrix under different datasets

As shown in Fig. 10 (a), the ratio of the two classes of samples in the Maxlittle datasetMaxlittle is 3:1. The prediction accuracy of the proposed model is 100% for the larger size class and 93.8% for the smaller size class. Wrong prediction only occurs in the second class of samples, and the main reason for the error is that the second class is predicted to be the first class. In Fig. 10 (b), the ratio of the two classes of samples in the PID dataset is 1:2, which is similar to the Maxlittle datasetMaxlittle. The prediction accuracy of the model for larger size classes is higher than that for smaller size classes. A similar pattern can be found in Fig. 10 (c) dataset LSVT and Fig. 10 (d) dataset Wisconsin. As seen from Fig. 10 (e), for the Statlog dataset, the overall accuracy is 84.7%, while the accuracy of a single class can reach 99.4%. The number of samples in the second and fifth classes is similar, but the classification accuracy is quite different, which are 94.5% and 50.2%, respectively. It can be seen that class 5 is mostly mistaken for class 2. A possible reason is that classes 2 and 5 are relatively close, and there are not enough samples for the classifier to learn better boundaries. Likewise, as seen from Fig. 10 (f), for the Urban dataset, the overall accuracy is 84.4%, while the recognition rate for a single class can reach 97.1%. By comparing the classification accuracy of each classification class with the overall accuracy, it can be seen that the difference between the classification accuracy of a single class and the overall classification accuracy is small, and the proposed model has certain advantages as to the class imbalance problem.

5. Conclusion

Deep learning is very important in machine learning algorithms. SAE is a very important and widely used algorithm among them. However, existing SAEs only focus on original samples without considering the hierarchical structure between samples. This limits the construction of more powerful samples and the improvement of accuracy. To solve this problem, the NE_ESAE model is proposed. First, the NSELM is proposed for preprocessing SAE. NSELM constructs a sample pair and a multilayer sample space, which considers the hierarchical structure between neighboring and similar samples and generates layers of envelope samples with better quality. Second, ESAE is proposed and trained in each layer of the sample space to consider the original features during training and in the network structure, thereby better finding the relationship between the samples with original features and deep features. Third, feature reduction and base classifiers are conducted on the layers of envelope samples and output classification results of layer samples. Finally, the classification results of the layers of envelope sample space are fused through the MSEM to realize the final results. In summary, the proposed NE_ESAE can consider the hierarchical structure between samples and obtain a multilayer of deep features, thereby overcoming the limitation of the existing SAEs.

To verify the effectiveness of NE_ESAE, we compared its performance with some representative feature learning methods and some representative SAEs. The results show that the proposed NE_ESAE has obvious advantages over most algorithms. Taking the AD dataset as an example, the classification accuracy of NE_ESAE is 40.00%, 34.00%, 35.09%, 32.89%, and 23.34% higher than that of the other SAEs. Finally, the influence of the classifier type and the cluster ratio in the IMC of the NE_ESAE model was analyzed, which provides a certain reference for parameter setup.

This proposed NSELM method not only deeply mines the structural information of the data but also learns the structural information of each layer of samples, which increases the complementarity of the learned features and enhances the representation ability of the learned features. The ESAE proposed in this paper is a lightweight deep network. The network introduces the original features into the structure and training process of the autoencoder, which improves the complementarity between the deep features and the original features, thereby achieving high-quality features. By combining the NSELM and ESAE, the NE_ESAE model is proposed to realize the sample-feature cooperative transformation. As seen from further analysis, the proposed NE_ESAE model has the significant advantages of better generalization ability, more stability, and higher classification accuracy. In summary, different from the existing SAEs, the proposed NE_ESAE realizes structured samples and the sample-feature cooperative transformation.

However, the proposed model in this paper also has some limitations. First, the NE_ESAE model mainly focuses on structural datasets and does not consider nonstructural datasets, such as image data and speech data. Second, the number of layers in the ESAE is not large (or not too deep). In the future, we plan to extend the proposed approach in two directions. One is to explore the network structure to enhance the feature learning ability, and the other is to incorporate more types of data.

Acknowledgments

The authors would like to thank the editor and reviewers for their valuable comments and suggestions. The authors would also like to thank those individuals or institutions that have provided data support for this research. This work was supported in part by the National Natural Science Foundation of China (NSFC) under grants 61771080 and U21A20448, the Key Project of Technology Innovation and Application Development in Chongqing (cstc2019jsxz-mbdxX0050), the Natural Science Foundation of Chongqing (cstc2020jscx-msxm0369, cstc2020jcyj-msxmX0100, cstc2020jscx-fyzx0212, cstc2020jcyj-msxmX0523, cstc2020jscx-gksbx0009, the Chongqing Social Science Planning Project (2018YBYY133), and the fund of Sichuan (21ZDYF3646).

Data availability

The data and codes can be found in https://github.com/CUTEZZZOE/ESSAE

Conflict of interest

The authors declare no conflicts of interest pertaining to this work.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

References

[1] Lee H, Kwon H. Going deeper with contextual CNN for hyperspectral image classification [J]. IEEE Transactions on Image Processing, 2017, 26(10):4843-4855.

[2] Uzair M, Shafait F, Ghanem B, et al. Representation learning with deep extreme learning machines for efficient image set classification[J]. Neural Computing and Applications, 2016, 30(4):1211-1223.

[3] Muhammad K, Khan S, Del Ser J, et al. Deep Learning for Multigrade Brain Tumor Classification in Smart Healthcare Systems: A Prospective Survey [J], IEEE Transactions on Neural Networks and Learning Systems, 2021, 32(2): 507-522.

[4] Roy S, Menapace W, Oei S, et al. Deep learning for classification and localization of COVID-19 markers in point-of-care lung ultrasound[J]. IEEE transactions on medical imaging, 2020, 39(8): 2676-2687.

[5] X. Liu et al., Deep Multiview Union Learning Network for Multisource Image Classification[J], IEEE Transactions on Cybernetics, 2022, 52(6):4534-4546.

[6] T. -W. Sun, End-to-End Speech Emotion Recognition With Gender Information[J], IEEE Access,2020, 8:152423-152438.

[7] B. Zhang, D. Xiong, J. Xie and J. Su, Neural Machine Translation With GRU-Gated Attention Model[J], IEEE Transactions on Neural Networks and Learning Systems, 2020, 31(11):4688-4698.

[8] Wickstrøm K, Mikalsen K Ø, Kampffmeyer M, et al. Uncertainty-aware deep ensembles for reliable and explainable predictions of clinical time series[J]. IEEE Journal of Biomedical and Health Informatics, 2020, 25(7): 2435-2444.

[9] R. Chandra, "Competition and Collaboration in Cooperative Coevolution of Elman Recurrent Neural Networks for Time-Series Prediction[J], IEEE Transactions on Neural Networks and Learning Systems, 2015, 26(12): 3123-3136.

[10] J. Yu, C. Hong, Y. Rui and D. Tao, Multitask Autoencoder Model for Recovering Human Poses [J], IEEE Transactions on Industrial Electronics, 2018, 65(6): 5060-5068.

[11] Wu F, Wang Z, Lu W, et al. Regularized Deep Belief Network for Image Attribute Detection[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2017, 27(7):1464-1477.

[12] J. Wang, J. Zhang and X. Wang, Bilateral LSTM: A Two-Dimensional Long Short-Term Memory Model With Multiply Memory Units for Short-Term Cycle Time Forecasting in Re-entrant Manufacturing Systems[J], IEEE Transactions on Industrial Informatics, 2018, 14(2): 748-758.

[13] Hinton G E, Salakhutdinov R R. Reducing the dimensionality of data with neural networks[J]. Science, 2006, 313(5786):504-507.

[14] Li R, Wang X, Quan W, et al. Stacked Fusion Supervised Auto-encoder with an Additional Classification Layer[J]. Neural Processing Letters, 2020, 51(3): 2649-2667.

[15] Yang S, Zhang Y, Zhu Y, et al. Representation learning via serial autoencoders for domain adaptation[J]. Neurocomputing, 2019, 351: 1-9.

[16] Farajian N, Adibi P. Minority manifold regularization by stacked auto-encoder for imbalanced learning[J]. Expert Systems with Applications, 2021, 169: 114317.

[17] Karim A M, Kaya H, Güzel M S, et al. A novel framework using deep auto-encoders based linear model for data classification[J]. Sensors, 2020, 20(21): 6378.

[18] Wang W, Du X, Shan D, et al. Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine[J]. IEEE Transactions on Cloud Computing, 2020.

[19] Liu W, Ma T, Tao D, et al. HSAE: A Hessian regularized sparse auto-encoders[J]. Neurocomputing, 2016, 187: 59-65.

[20] Wang Y, Yang H, Yuan X, et al. Deep learning for fault-relevant feature extraction and fault classification with stacked supervised auto-encoder[J]. Journal of Process Control, 2020, 92: 79-89.

[21] T. Su, Y. Liu, J. Zhao and J. Liu, Probabilistic Stacked Denoising Autoencoder for Power System Transient Stability Prediction With Wind Farms[J], IEEE Transactions on Power Systems, 2021, 36(4): 3786-3789.

[22] J. Yang et al., No Reference Quality Assessment for Screen Content Images Using Stacked Autoencoders in Pictorial and Textual Regions [J], IEEE Transactions on Cybernetics, 2022, 52(5): 2798-2810.

[23] Kingma D P, Welling M. Auto-Encoding Variational Bayes[C]. 2nd International Conference on Learning Representations, ICLR 2014, 2014.

[24] Wang X, Tan K, Du Q, et al. CVA2E: A Conditional Variational Autoencoder With an Adversarial Training Process for Hyperspectral Imagery Classification[J]. IEEE Transactions on Geoscience and Remote Sensing, 2020, 58(8):5676-5692.

[25] Xie X, Li Z, Zhang P, et al. Information structures and uncertainty measures in an incomplete probabilistic set-valued information system[J]. IEEE Access, 2019, 7: 27501-27514.

[26] Li F, Shang C, Li Y, et al. Interpolation with just two nearest neighboring weighted fuzzy rules[J]. IEEE Transactions on Fuzzy Systems, 2019, 28(9): 2255-2262.

[27] Rodriguez A, Laio A. Clustering by fast search and find of density peaks[J]. Science, 2014, 344(6191):1492-1496.

[28] Wang Y, Xia S, Tang Q, et al. A Novel Consistent Random Forest Framework: Bernoulli Random Forests[J]. IEEE Transactions on Neural Networks and Learning Systems, 2018, 29(8):3510-3523.

[29] Zeng L Z, Benatallah B, Ngu A H H, et al. QoS-aware middleware for Web services composition[J]. IEEE Transactions on Software Engineering, 2004, 30(5): 311-327.

[30] Cover T, Hart P. Nearest neighbor pattern classification[J]. IEEE transactions on information theory, 1967, 13(1): 21-27.

[31] Xu R, Wunsch D. Survey of clustering algorithms[J]. IEEE Transactions on neural networks, 2005, 16(3): 645-678.

[32] Cui S, Wang Y, Yin Y, et al. A cluster-based intelligence ensemble learning method for classification problems[J]. Information Sciences, 2021, 560: 386-409.

[33] Yang M, Sinaga K P. A Feature-Reduction Multi-View k-Means Clustering Algorithm[J]. IEEE Access, 2019, 7:114472-114486.

[34] Y. Lin and S. Chen, A Centroid Auto-Fused Hierarchical Fuzzy c-Means Clustering [J], IEEE Transactions on Fuzzy Systems, 2021, 29(7): 2006-2017.

[35] Shen J, Hao X, Liang Z, et al. Real-Time Superpixel Segmentation by DBSCAN Clustering Algorithm[J]. IEEE Transactions on Image Processing, 2016, 25(12):5933-5942.

[36] Zhang J, Feng X, Liu Z. A Grid-Based Clustering Algorithm via Load Analysis for Industrial Internet of Things[J]. IEEE Access, 2018, 6:13117-13128.

[37] Misbahulmunir S, Ramachandaramurthy V K, Thayoob Y H M. Improved Self-Organizing Map Clustering of Power Transformer Dissolved Gas Analysis Using Inputs Pre-Processing[J]. IEEE Access, 2020, 8:71798-71811.

[38] Li F, Zhang X, Wang P, et al. Deep instance envelope network-based imbalance learning algorithm with multilayer fuzzy C-means clustering and minimum interlayer discrepancy[J]. Applied Soft Computing, 2022, 123: 108846.

[39] Long M, Cao Y, Wang J, et al. Learning transferable features with deep adaptation networks[C]//International conference on machine learning. PMLR, 2015: 97-105.

[40] B. Yang, Y. Lei, F. Jia, N. Li and Z. Du, A Polynomial Kernel Induced Distance Metric to Improve Deep Transfer Learning for Fault Diagnosis of Machines [J], IEEE Transactions on Industrial Electronics, 2020, 67(11): 9747-9757.

[41] Lei Y, Yuan W, Wang H, et al. A skin segmentation algorithm based on stacked autoencoders[J]. IEEE Transactions on Multimedia, 2017, 19(4):740-749.

[42] Praveen G B, Agrawal A, Sundaram P, et al. Ischemic stroke lesion segmentation using stacked sparse autoencoder[J]. Computers in Biology and Medicine, 2018, 99:38-52.

[43] Fan J, Lin K, Han M. A novel joint change detection approach based on weight-clustering sparse autoencoders[J]. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2019, 12(2): 685-699.

[44] Shi Y, Lei J, Yin Y, et al. Discriminative feature learning with distance constrained stacked sparse autoencoder for hyperspectral target detection[J]. IEEE Geoscience and Remote Sensing Letters, 2019, 16(9):1462-1466.

[45] Liu L, Wang Y, Peng J, et al. Latent relationship guided stacked sparse autoencoder for hyperspectral imagery classification[J]. IEEE Transactions on Geoscience and Remote Sensing, 2020, 58(5): 3711-3725.

[46] Görgel P, Simsek A. Face recognition via deep stacked denoising sparse autoencoders (DSDSA)[J]. Applied Mathematics and Computation, 2019, 355:325-342.

[47] Zhu H, Cheng J, Zhang C, et al. Stacked pruning sparse denoising autoencoder based intelligent fault diagnosis of rolling bearings[J]. Applied Soft Computing, 2020, 88: 106060.

[48] Q. Sun and Z. Ge, Gated Stacked Target-Related Autoencoder: A Novel Deep Feature Extraction and Layerwise Ensemble Method for Industrial Soft Sensor Application [J], IEEE Transactions on Cybernetics, 2022, 52(5): 3457-3468.

[49] Martınez-Munoz G, Suárez A. Aggregation ordering in bagging[C]//Proc. of the IASTED International Conference on Artificial Intelligence and Applications. Citeseer, 2004: 258-263.

[50] Mika S, Ratsch G, Weston J, et al. Fisher discriminant analysis with kernels[C]//Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468). Ieee, 1999: 41-48.

[51] Qu Q, Zhang Y, Eldar Y C, et al. Convolutional phase retrieval via gradient descent[J]. IEEE Transactions on Information Theory, 2019, 66(3): 1785-1821.

[52] Cai J F, Candès E J, Shen Z. A singular value thresholding algorithm for matrix completion[J]. SIAM Journal on optimization, 2010, 20(4): 1956-1982.

[53] L. Zhang, S. Wang, G. -B. Huang, W. Zuo, J. Yang and D. Zhang, Manifold Criterion Guided Transfer Learning via Intermediate Domain Generation [J], IEEE Transactions on Neural Networks and Learning Systems, 2019, 30(12): 3759-3773.

[54] Tan X, Liu Y, Li Y, et al. Localized instance fusion of MRI data of Alzheimer's disease for classification based on instance transfer ensemble learning[J]. Biomedical engineering online, 2018, 17(1): 1-17.

[55] Tsanas A, Little M A, Fox C, et al. Objective automatic assessment of rehabilitative speech treatment in Parkinson's disease[J]. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2013, 22(1): 181-190.

[56] Sakar B E, Isenkul M E, Sakar C O, et al. Collection and analysis of a Parkinson speech dataset with multiple types of sound recordings[J]. IEEE Journal of Biomedical and Health Informatics, 2013, 17(4): 828-834.

[57] Hoti F, Holmström L. A semiparametric density estimation approach to pattern classification[J]. Pattern Recognition, 2004, 37(3): 409-419.

[58] Dua, D. Graff, C. UCI Machine Learning Repository, University of California, School of Information and Computer Science. Irvine, CA, 2019. [http://archive.ics.uci.edu/ml].

[59] M. A. Little, P. E. McSharry, E. J. Hunter, J. Spielman and L. O. Ramig, Suitability of Dysphonia Measurements for Telemonitoring of Parkinson's Disease [J], IEEE Transactions on Biomedical Engineering, 2009, 56(4): 1015-1022.

[60] Johnson B, Xie Z. Classifying a high resolution image of an urban area using super-object information[J]. ISPRS journal of photogrammetry and remote sensing, 2013, 83: 40-49.

[61] Wolberg W H, Street W N, Mangasarian O L. Machine learning techniques to diagnose breast cancer from image-processed nuclear features of fine needle aspirates[J]. Cancer letters, 1994, 77(2-3): 163-171.

[62] Wolberg W H, Mangasarian O L. Multisurface method of pattern separation for medical diagnosis applied to breast cytology[J]. Proceedings of the national academy of sciences, 1990, 87(23): 9193-9196.

[63] Smith J W, Everhart J E, Dickson W C, et al. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus[C]//Proceedings of the annual symposium on computer application in medical care. American Medical Informatics Association, 1988: 261.

[64] Seghouane A K, Shokouhi N, Koch I. Sparse principal component analysis with preserved sparsity pattern[J]. IEEE Transactions on Image Processing, 2019, 28(7): 3274-3285.

[65] Li, Cheng-Hsuan, Bor-Chen Kuo, and Chin-Teng Lin. LDA-based clustering algorithm and its application to an unsupervised feature extraction[J]. IEEE Transactions on Fuzzy systems, 2010, 19(1): 152-163.

[66] Wang R, Nie F, Hong R, et al. Fast and orthogonal locality preserving projections for dimensionality reduction[J]. IEEE Transactions on Image Processing, 2017, 26(10): 5019-5030.

[67] Deng Z, Chung F L, Wang S. Robust relief-feature weighting, margin maximization, and fuzzy optimization[J]. IEEE Transactions on Fuzzy Systems, 2010, 18(4): 726-744.

[68] Yamada M, Jitkrittum W, Sigal L, et al. High-dimensional feature selection by feature-wise kernelized lasso[J]. Neural computation, 2014, 26(1): 185-207.

[69] Donoho D, Jin J. Higher criticism thresholding: Optimal feature selection when useful features are rare and weak[J]. Proceedings of the National Academy of Sciences, 2008, 105(39): 14790-14795.

[70] Li Y, Lei Y, Wang P, et al. Embedded stacked group sparse autoencoder ensemble with L1 regularization and manifold reduction[J]. Applied Soft Computing, 2021, 101: 107003.