Fei *et al. EURASIP Journal on Wireless Communications and Networking* (2020) 2020:91
https://doi.org/10.1186/s13638-020-01706-4

EURASIP Journal on Wireless Communications and Networking

RESEARCH **Open Access**

A new deep sparse autoencoder for community detection in complex networks

Rong Fei[1]*, Jingyuan Sha[1], Qingzheng Xu[2], Bo Hu[3], Kan Wang[1] and Shasha Li[1]

*Correspondence:
annyfei@xaut.edu.cn
[1]Xi'an University of Technology, JinHua Road No.5, Xi'an, China
Full list of author information is available at the end of the article

Abstract
Feature dimension reduction in the community detection is an important research topic in complex networks and has attracted many research efforts in recent years. However, most of existing algorithms developed for this purpose take advantage of classical mechanisms, which may be long experimental, time-consuming, and ineffective for complex networks. To this purpose, a novel deep sparse autoencoder for community detection, named DSACD, is proposed in this paper. In DSACD, a similarity matrix is constructed to reveal the indirect connections between nodes and a deep sparse automatic encoder based on unsupervised learning is designed to reduce the dimension and extract the feature structure of complex networks. During the process of back propagation, L-BFGS avoid the calculation of Hessian matrix which can increase the calculation speed. The performance of DSACD is validated on synthetic and real-world networks. Experimental results demonstrate the effectiveness of DSACD and the systematic comparisons with four algorithms confirm a significant improvement in terms of three index F_{same}, *NMI*, and modularity *Q*. Finally, these achieved received signal strength indication (RSSI) data set can be aggregated into 64 correct communities, which further confirms its usability in indoor location systems.

Keywords: Community detection, Deep sparse autoencoder, RSSI

1 Introduction

Community detection has a great significance to the study of complicated systems and our daily life; meanwhile, it is also one of the important methods for understanding many network structures in the real world. In the network, community structure implies some nodes in the network that are closely connected with each other, but sparsely connected with other nodes. Community detection divides nodes into a module in the graph, so that the inner side numbers of the module is larger than the edge numbers between modules with the topological structure of graphs as the source of information [1, 2].

At present, a variety of community detection algorithms have been proposed explore the community structure of complex networks.

The community mining method LPA [3] based on label propagation was proposed in 2007. It counts the labels of adjacent nodes of each node, and the highest frequency label is used as the new label of the node. The LPA is still a classic algorithm because it can process large networks for its linear running time to network size, and the propagation of labels also avoids predefine the community number. In addition, the LPA allows a vertex to carry multiple labels [4] .The LPA is particularly suitable for large social networks with complex and overlapping communities [5]. There are many improved LPA that appears, such as a new LPA parallelization scheme from a different perspective [5], the local affinity propagation (LAP) algorithm with near-linear time, and space complexities [6].

Classical clustering algorithms such as K-means are basic methods to solve the community discovery problem [7]. The speed of classical algorithms is fast enough, but their accuracy and stability need to be improved. When a similarity matrix is generated, only neighbor nodes are taken into account, and the related nodes are not included in the scope of consideration. The large scale of social networks poses challenges from the viewpoint of clustering methods. The high-dimensional similarity matrix in the network will aggravate the decline of accuracy.

Deep learning is about learning multiple levels of representation and abstraction that help to make sense of data such as images, sound, and text. Deep learning has shown how hierarchies of features can be learned in an unsupervised manner directly from data. The idea is to learn the representation of data at different levels or aspects through a computational model with multi-layer networks [8–10].

The application of deep learning algorithm has developed rapidly, such as driverless, biological information, and community detection, which displays excellent adaptability and practicability.

Chi-Hua Chen [11] proposed a deep learning model with generalization performance to obtain probability density function from the cumulative distribution function-based real data, which can be produced to improve further analyses with game theory or queueing theory. He [12] also proposed a cell probe-based method to analyze the cellular network signals and trained regression models for vehicle speed estimation, which is effective for cellular floating vehicle data.

In 2017, Shang et al. proposed the community detection algorithm based on deep sparse autoencoder (CoDDA) algorithm [13] that reduced the dimension of the network similarity matrix by establishing a deep sparse autoencoder. The lower dimension matrix with more obvious community structure was obtained. Finally, the K-means algorithm was used to cluster, and results with higher accuracy were obtained.

A recursive form is common in back propagation (BP) algorithm. The essence of the BP process is to minimize the reconstruction error, which can be classified as optimization problem [14–17]. The problem is generally solved by using nonlinear optimization methods which include gradient descent method [18], nonlinear conjugate gradient method [19], and quasi-Newton method. Each step of the calculation process of quasi Newton method only involves the calculation of function value and function gradient value; in this way, Hessian matrix calculation problem in other gradient descent method is effectively avoided [20].

Location-based services (LBS) make the wireless content industry develop to be close with public market application. WiFi is the common positioning signals in location -based services (LBS) [21, 22], and the signal intensity or spacetime attributes are combined with positioning algorithms to form the position technology [23]. Recently, Wi-Fi finger-printing positioning method often combines deep learning methods to improve the accuracy of classical KNN for indoor positioning [24].

Based on the deep sparse autoencoder and quasi-Newton method, we construct a community detection architecture, which improves the information loss in high-dimensional reduction. The proven architecture-DSACD first reduces the dimension of the high-dimensional matrix and then realizes community discovery by a deep sparse autoencoder. DSACD also improves the accuracy of K-means algorithm. In the application, the real community data sets for LBS are trained by the deep sparse autoencoder and optimized by the quasi-Newton method. The time loss in the process of community discovery is reduced to improve the efficiency of the algorithm; meanwhile, the accuracy is guaranteed.

The rest of this paper is organized as follows: Section 2 explains the proper nouns and algorithms appearing. In Section 3, the experimental process are introduced. DSACD performs confirmatory experiments through multiple experimental sets, including several parameter experiments to optimize the algorithm. The results of the experiment were evaluated by citing several evaluation criteria. Section 4 gives results discussion and an application. In Section 5, we consider the idea of future development based on these results.

- We construct a deep sparse autoencoder with L-BFGS method for community detection to accelerate the process of clustering community structures in large data set.
- In DSACD, a similarity matrix is constructed to show the indirect connections between nodes, a deep sparse automatic encoder is constructed with L-BFGS algorithm based on unsupervised learning to reduce the dimension and extract the feature structure of the network, and comparison experiments with four algorithms show a significant improvement in terms of three index F_{same}, NMI, and modularity Q of DSACD.
- We creatively design community detection experiments—a location-based service networks. The results show that DSACD can divide the partition into 64 communities which can prove its practicability.

2 Preliminaries

The definitions of DSACD studied in this paper are given as follows.

2.1 Matrix preprocessing

Let $G = (V, E)$ be the graph, where $V = v1, v2, ..., vn$ represents the set of nodes (vertices) in the graph and E represents the set of edges in the figure. Let $N(u)$ be the neighbor nodes set of node u. Let matrix $A = [a_{ij}]_{n\times n}$ be the adjacency matrix of graph G, and the corresponding elements of the matrix represent whether there are edges between two points in graph G. For example, a_{ij} equals 1 indicates that there exists an e_{ij}. If a_{ij} equals 0, the indication is that there is no e_{ij}.

In the small graph, the adjacency matrix A can directly calculate the community relationship in the graph using a clustering algorithm such as K-means, and the result is more accurate (see Section 3). However, the adjacency matrix records only the relationship between adjacent nodes, and does not express the relationship between the node and its neighbors, or even more distant nodes. For any two nodes in the community, even if they are not connected to each other, it is possible to have the same community. Therefore, if the adjacency matrix is directly used as the similarity matrix for community partitioning, the complete community relationship cannot be reflected. If the adjacency matrix is directly clustered, the information will be lost.

In this paper, on the premise of definition: the similarity matrix which can express the non-adjacent information matrix is calculated by transforming the adjacency . Based on this, the definitions are given as follows.

Definition 1 *Let a network graph be* $G = (V, E)$ *for* $\forall v \in V$. *If the number of the shortest path from the node* v_i *to another node* v_j *is s, then, the node* v_i *can jump to node* v_j *through s hops. That is, hop is the number of least traversed edges from the node* v_i *to* v_j.

As shown in the network of Fig. 1, node v_1 reaches v_2, v_3, or v_6 after one hop and arrives at v_4 or v_5 after two hops. For an instance, from $v_1 \rightarrow v_2$, the number of least traversed edges is 1, so the hop count is 1; from $v_1 \rightarrow v_5$, the number of least traversed edges is 2, and the hop count is 2.

Definition 2 *In* $G = (V, E)$, *the* ***similarity between two points*** v_i *and* v_j *is defined as formula 1:*

$$Sim(i,j) = e^{\tau(1-s)}, s \geq 1, \tau \in (0,1) \tag{1}$$

where the hop number from v_i *to* v_j, *and* τ *is the attenuation factor. The node similarity decreases with the increase in the hop threshold s,* τ *controls the attenuation rate of the similarity, and the velocity of the node similarity relationship decays faster with the increase in* τ.

Definition 3 *In* $G = (V, E)$, *its (**similarity matrix**)* $S = [s_{ij}]_{n\times n}$ *is calculated by the node similarity between two points in G where* $s_{ij} = Sim(v_i, v_j), v_i, v_j \in V$.

The similarity matrix obtained by processing the adjacency matrix by the hop count and the attenuation factor can better reflect the relationship between the distant nodes in the high-dimensional matrix, and the results of the community discovery are also improved.

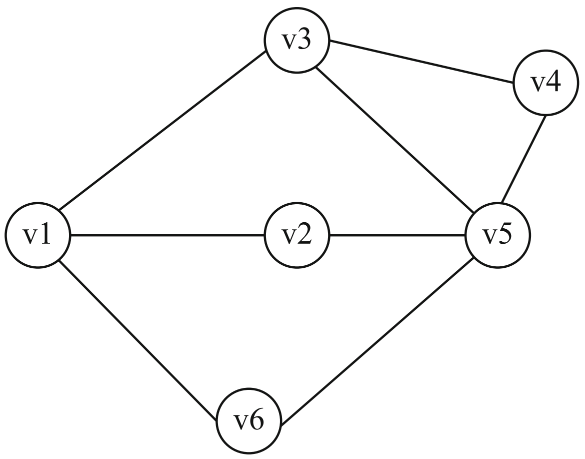

Fig. 1 Small network schematic

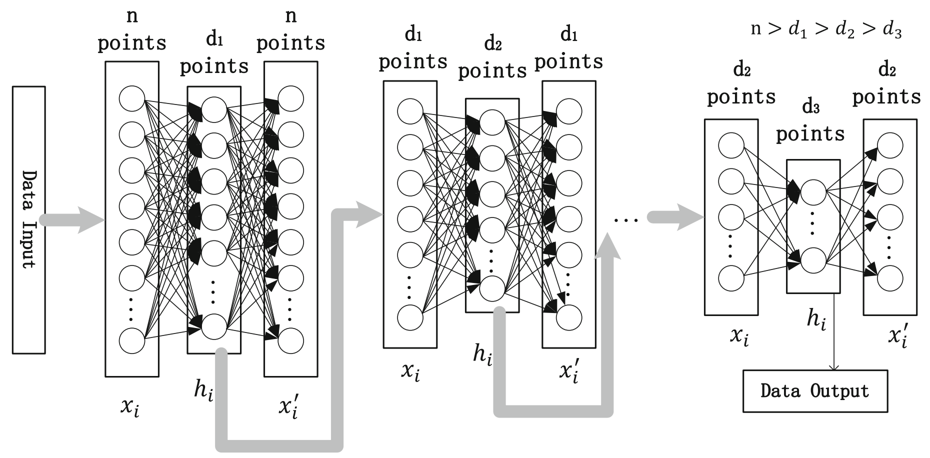

Fig. 2 Deep sparse autoencoder structure

Obviously, the selection of the hop count threshold and the attenuation factor will have an important impact on the similarity matrix. The selection of hop count is obtained from the parameter learning process, which is explained at Section 3.6. Section 3 of this paper will set up experiments on these two parameters to explore the impact of different parameters on the results.

2.2 Deep sparse autoencoder

Based on a sparse autoencoder, the structure of deep sparse autoencoder is shown in Fig. 2.

The output of the previous layer, that is, the code h after dimension reduction, is shown in Fig. 2, as the input of the next layer. Then, the dimensions are reduced one by one.

Autoencoder [25] is an unsupervised learning artificial neural network that can learn the efficient encoding of data to express the eigenvalues of the data. The typical usage of the AE is to reduce dimensionality.

Definition 4 *As shown in Figs. 3 and 4, given an unlabeled data set* $\{x^{(i)}\}_{i=1}^{m}$, *the* ***automatic encoder*** *learns the nonlinear code through a two-layer neural network (input layer*

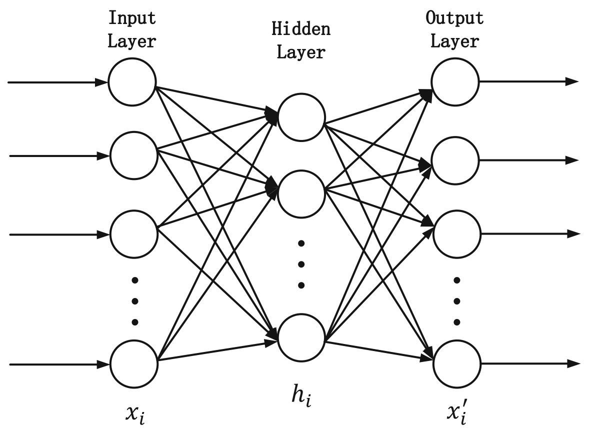

Fig. 3 Structure of the automatic encoder

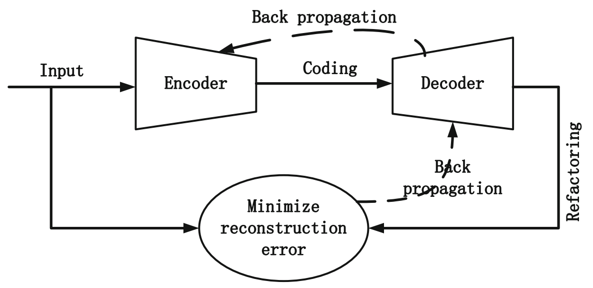

Fig. 4 Process of the automatic encoder

is not counted) to express the original data, the [25] *training process uses the back propagation algorithm, and the sign of the end of the training is that the difference between the learned nonlinear code and the original data is minimized.*

The automatic encoder is composed of two parts: the coder (encode) and the decoder (decode). The encoding process is from the input layer to the hidden layer. At this time, the input data are subjected to dimensionality reduction to form a code, which is encoded as the output of the encoder, and then, the code is used as an input of the decoder for decoding, and the decoded result has the same dimension as the input data, used as the output of the decoder. After the output result is obtained, the output result is compared with the input result, the reconstruction errors are calculated, and then, the back-propagation algorithm is used to adjust the weight matrix of the automatic encoder. The reconstruction errors are calculated again and iterated continuously until the number of iterations or the reconstruction errors are less than the specified range. The output is equal to or close to the input result. The process of training a neural network using a back-propagation algorithm is also referred to as the minimization of the reconstruction error. Finally, the output of the encoder, i.e., the encoding, is taken as the output of the automatic encoder.

Specific steps are as follows:

Let X be the network graph G similarity matrix with dimension n. As the input matrix, where $x_i \in (0,1)$, $X \in R^(n \times n)$. $x_i \in R^(n \times 1)$ represents the ith column vector in X, $W_1 \in R^(d \times n)$ is the weight matrix of the input layer [26], and $W_2 \in R^(n \times d)$ is the weight matrix of the hidden layer [27].

$b \in R^(d \times 1)$ is the offset column vector of the hidden layer [27].

$c \in R^(n \times 1)$ is the offset column vector of the input layer [27].

The output h of the coding layer is obtained by formula 2:

$$h_i = \tau(W_1 x_i + b) \tag{2}$$

where $h_i \in R^(d \times 1)$ is the encoded ith column vector. τ is the activation function, and the sigmoid function [28] is chosen as the activation function τ, which is shown by formula 3.

$$f(z) = \frac{1}{1+e^{-z}} \tag{3}$$

In formula 3, $z = W^T X$.

The matrix h obtained at this time is a matrix after dimensionality reduction.

The output z of the decoding layer is obtained by formula 4:

$$z_i = \tau(W_2 h_i + c) \tag{4}$$

where $z_i \in R(n \times 1)$ is the decoded ith column vector. τ is the activation function. The resulting matrix z is the same as the X dimension of the input matrix.

Combining the formula 2 with the formula 4, the reconstruction error is obtained:

$$\text{error} = \sum_{i=1}^{n} \|\tau(W_2\tau(W_1 x_i + b) + c) - x_i\|_2^2 \tag{5}$$

When the activation function is sigmoid, the mapping range of the neurons is $(0, 1)$. When the output is close to 1, it is called active, and when the output is close to 0, it is called inactive [29]. In a sparse autoencoder, sparseness restrictions are added to the hidden layer. A sparse restriction means that neurons are suppressed most of the time, that is, the output is close to 0. A sparse expression has been successfully applied to many applications, such as target recognition [30, 31], speech recognition [32], and behavior recognition [33]. The sparsity calculation method is as follows:

First, the average value of the output of the coding layer $\widehat{\rho_j}$ is calculated and $h_j(x)$ denotes the output value of the neuron for the jth neuron (h_j) of the hidden layer when the input is x [34]. The average value of the neuron output in the hidden layer is:

$$\widehat{\rho_j} = \frac{1}{m}\sum_{i=1}^{m}[h_j(x_i)] \tag{6}$$

To achieve sparsity, it is necessary to add a sparsity limit, which is achieved by:

$$\widehat{\rho_j} = \rho \tag{7}$$

where ρ is the sparsity parameter, generally, $\rho \ll 1$, such as 0.05. When formula 7 is satisfied, the activation value of the hidden layer neurons is mostly close to 0.

A sparsity limit is added to the reconstruction error, that is, a penalty term is added to the reconstruction error, and $\widehat{\rho_j}$ deviating from ρ is punished. The penalty function is as follows:

$$\sum_{j=1}^{d} \rho \log\frac{\rho}{\widehat{\rho_j}} + (1-\rho)\log\frac{1-\rho}{1-\widehat{\rho_j}} \tag{8}$$

where d represents the number of hidden layer neurons. This formula is based on Kullback-Leibler divergence (KL [35]), so it can also be written as formula 9:

$$\sum_{j=1}^{d} KL(\rho \parallel \widehat{\rho_j}) \tag{9}$$

In summary, formula 8 and formula 9 are combined to obtain the following:

$$KL(\rho \parallel \widehat{\rho_j}) = \rho \log\frac{\rho}{\widehat{\rho_j}} + (1-\rho)\log\frac{1-\rho}{1-\widehat{\rho_j}} \tag{10}$$

When $\widehat{\rho_j} = \rho$, the penalty function $KL(\rho \parallel \widehat{\rho_j})$ is 0. When $\widehat{\rho_j} = \rho$ is far from ρ, the function monotonically increases and tends to infinity, as shown in Fig. 5:

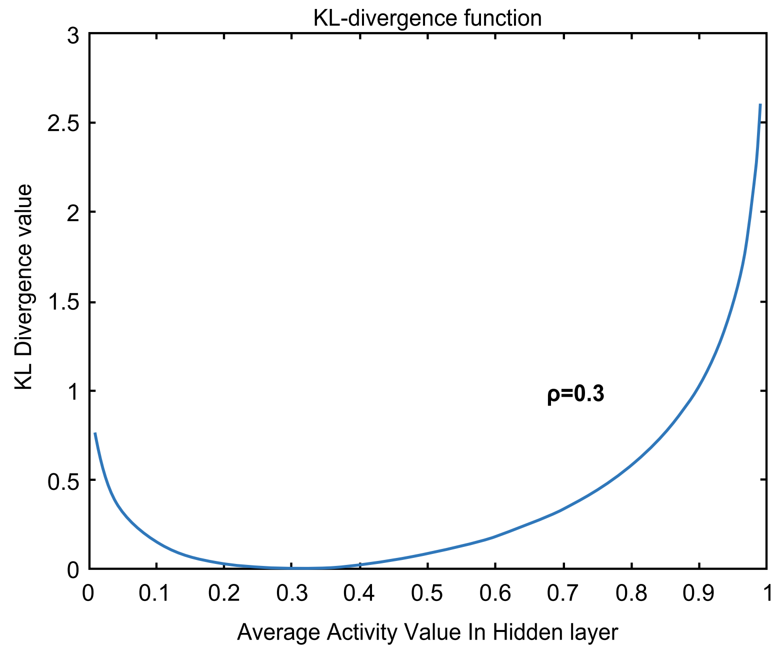

Fig. 5 KL divergence function

Therefore, by minimizing the sparse term penalty factor, that is, formula 10, $\widehat{\rho_j}$ is closed to ρ. At this point, the reconstruction error is updated to formula 11:

$$\text{error} = \sum_{i=1}^{n} \|\tau(W_2\tau(W_1x_i + b) + c) - x_i\|_2^2 + \beta \sum_{j=1}^{d} KL(\rho \parallel \widehat{\rho_j}) \tag{11}$$

where β is the weight of the sparse penalty factor.

The training sparse autoencoder minimizes the reconstruction error by the back-propagation algorithm, that is, formula 11.

2.3 The deep sparse autoencoder for community detection

Based on the deep sparse autoencoder shown in Fig. 2, the data are preprocessed first, and the similarity matrix $S_0 \in R^{(n\times n)}$ is obtained by formula 1. The similarity matrix is used as the input of the deep sparse autoencoder, then, the number of layers T of the deep sparse autoencoder is set and the number of nodes per layer $\{d_0, d_1, d_2, \cdots, d_T \mid d_0 = n, d_0 > d_1 > d_2 > \cdots > d_T\}$. The similarity matrix $S_0 \in R^{(n\times n)}$ is input into the sparse autoencoder with the hidden layer as d_1 as the input data of the first layer. After the first layer of training, the dimensioned matrix $S_1 \in R^{(n\times d_1)}$ is obtained, then, S is input into the second layer of the deep sparse autoencoder, and then, the dimension is reduced to obtain $S_2 \in R^{(n\times d_2)}$, etc., until the last layer. The low-dimensional feature matrix $S_T \in R^{(n\times d_T)}$ is obtained, and finally, the community is obtained by K-means clustering. See algorithms 1 and 2 for the detailed process.

Algorithm 1 hop count threshold S, attenuation factor σ, and formula 1 are used to compute the similar degree matrix *sim* of $A \in R^{(n\times n)}$. Algorithm 1 is used to obtain the similarity matrix by computing the similarity of x with other nodes in V.

Algorithm 2 uses the deep sparse autoencoder with L-BFGS in which the layer number is T to reduce the dimension for a similar degree matrix, and then, the feature is extracted,

Algorithm 1

```
Input: adjacency matrix $A$ of the network graph
  $k$ - the number of communities
  $S$ - the threshold of hops
  $\sigma$ - attenuation factor
  $T$ - the layer number of the deep sparse autoencoder
  $d$ (vector) - the number of nodes in each layer
Output: similarity matrix
  give an adjacency matrix
  perform preprocessing of adjacency matrix based on hops
  initialize the number of hops between any two endpoints as 1000
  initialize a $m * m$ similarity matrix sim
  for all $x \in v$ do
    initialize the status used to store the status of each node as the state of not visit
    initialize the queue
    initialize hop number set $D$
    initialize each hop number of hop number set (matrix) hop as 1000
    flag node $x$ as visiting
    initialize the hop number of node $x$ itself as 0
    write node $x$ and hop number 0 into the hop number set $D$
    push node $x$ into queue
  end for
  while $queue = 0$ do
    get node $u$ from queue
    for all node $v$ in queue $U$ do
      if $u$ is in the $(S - 1)$ hop of node $x$, and the state of $v$ is unvisited then
        let $v$ as the state of visiting
        write the hop number of $v$ and $x$ to $v$ into the hop number set $D$
        hops from $v$ to $v$ are 0
        hops $x$ to $v$ are equal to hops $x$ to $u$ plus 1
        push $v$ into queue
      end if
    end for
    set u as the state of end visit
  end while
  for all $v \in V$ do
    if $hop(x, y) < 1000$ then
      $Sim(x, y) = exp(\sigma * (1 - hop(x, y)))$
    else
      $Sim(x, y) = 0$
    end if
  end for
  get the similarity matrix $X$ based on the hop number
```

```
Algorithm 2
Input: sim matrixes
Output: reduced sim matrixs
  $S_1 := Sim$
  for $t \leftarrow 1, T$ do
    choose the back-propagation algorithm (CoDDA/ DSACD)
    input the characteristic matrix $S_t$
    minimum the Reconstruction error of formula 11 by the back-propagation algorithm

    get the output $h_t$ of hidden layer
    $S_{t+1} = S_t$
  end for
```

and the low-dimensional characteristic matrix $S_T \in R^{(n \times d_T)}$ is obtained. Algorithm2 is used to reduce the similarity matrix and obtain the characteristics.

The K-means algorithm is used in S_T to obtain the cluster result $Coms = \{C_1, C_2, \cdots, C_k\}$ and then return it.

After the K-means algorithm is used, the communities $\{C_1, C_2, \cdots, C_k\}$ are obtained, and then, the result is returned.

In the proposed algorithm, the inputs include the adjacent matrix $A \in R^{(n \times n)}$ of the $G = (V, E)$, k—the number of communities, S—the hop count threshold, σ—the attenuation factor, and T—the layer number of deep sparse autoencoder and nodes in every layer $d_t = \{t = 0, 1, 2, \cdots, T \mid d_0 > d_1 > \cdots > d_T\}$.

3 Experiments and analyses

3.1 Experimental design

Since this experiment is a test of the community detection algorithm, the ground-truth communities are selected for verification, so that the accuracy of the algorithm can be analyzed and verified accurately.

The hardware environment of our experiment is as follows: processor Intel Xeon 2.10 GHz E5-2683 v4, memory 64GB 2400 MHz, operating system Windows Server 2012 R2 standard, IDE: MATLAB 2015 (R2015b).

This experiment used four real data sets: Strike [36], Football [37], LiveJournal [38], and Orkut [39]. Among them, Strike is a 24-striker relationship table on wood processing projects. The frequency of discussion for strike topics between two people is the rules, which are added. If the frequency is high (there are specific criteria for evaluation during the investigation, no detailed explanation will be given here), then, a connection is established. Football is the timetable for the American Football Cup (FBS) held by the American College Sports Association (NCAA) in 2006. In the NCAA relationship network, if two teams played games, the connection is established.

LiveJournal is a free online blogging community where users can add friends. LiveJournal can create groups. When collecting community information, the software classifies it according to cultural background, entertainment preferences, sports, games, lifestyle, technology, etc.

Table 1 Data set information

Data set	Number of nodes	Number of edges	Number of communities	Average community size
Strike	24	38	3	8
Football	180	788	12	15
LiveJournal	6368	90599	8	796
com-orkut	8929	138690	12	744

Orkut is a social service network launched by Google. Friend relationship and group of friends can be constructed.

On the social network, nearly 4 million points and 30 million edges were extracted, and 8 communities with the largest number of nodes were selected to conduct experiments as data sets. Detailed information on each experimental set is shown in Tables 1 and 2.

3.2 Evaluation index

To determine whether the clustering result is accurate, it is necessary to evaluate the clustering results $Coms = \{C_1, C_2, \cdots, C_k\}$. The evaluation method selects F_{same} [3] and NMI. Both methods are evaluated according to the real community $GroundTruth = \{C'_1, C'_2, \cdots, C'_l\}$, in where l is the true number for communities. Moreover, Q [39, 40] was used to evaluate the quality of the community.

Evaluation standard F_{same}:

The community evaluation standard F_{same} is obtained by calculating the intersection of each real community and each cluster community and averaging these values. The formula is as follows:

$$F_{\text{same}} = \frac{1}{2n}\left(\sum_{i=1}^{k} \max_j \left|C_i \cap C'_j\right| + \sum_{j=1}^{t} \max_i \left|C_i \cap C'_j\right|\right) \tag{12}$$

where in the graph G, the number of nodes is n.

Evaluation standard *NMI*:

The NMI is the normalized mutual information. The formula is as follows:

$$NMI(C, C') = \frac{-2\sum_{i=1}^{C}\sum_{j=1}^{C'} N_{ij}\log\left(\frac{N_{ij}N}{N_{i.}N_{.j}}\right)}{\sum_{i=1}^{C} N_{i.}\log\left(\frac{N_{i.}}{N}\right) + \sum_{j=1}^{C'} N_{.j}\log\left(\frac{N_{.j}}{N}\right)} \tag{13}$$

where $N \in R^{(n\times n)}$ is the confusion matrix, the rows represent the real community, and the columns represent the communities found. N_ij represents the point of overlap between the real community C_i and the discovery community C'_j. $N_{.j}$ represents the sum of all the elements in column j, and N_i represents the sum of all the elements in row i. If the community is found to be in full agreement with the real community [41], the *NMI* value is 1. If the community is found to be completely different from the real community, the *NMI* value is 0.

Table 2 Number of layers of deep sparse autoencoder in different data sets

Data set	Number of nodes	Number of layers
Strike	24-16	2
Football	180-128	2
LiveJournal	6368-4096-2048-1024	4
com-orkut	8929-4096-2048-1024-512	5

Evaluation standard *Q*:

Modularity Q is a measure of how well a community is found. The formula is as follows:

$$Q = \sum_{i=1}^{k} \left(\frac{E_i^{in}}{m} - \left(\frac{2E_i^{in} + E_i^{out}}{2m} \right)^2 \right) \tag{14}$$

where E_i^{in} is the inner edges number of the community C_i, E_i^{out} is the outer edge number of the community C_i, and m is the total edge number in the graph G.

3.3 Analysis experiments

The experiment consists of four parts, which are the volatility exploration experiment based on the DSACD, the comparison experiment with other algorithms, the parameter experiment, and the visualization experiment. The volatility exploration experiment is to show the fluctuation of our algorithm on different data sets, which can explain the stability for our algorithm on large data set. We compare DSACD with CoDDA and K-means based on three performance evaluation standards as F_{same}, NMI, and modularity Q. For explaining the result of parameter selection, the parameter experiment is given. At last, we use the visualization experiment to show the clustering results.

3.4 Volatility exploration analysis

According to Algorithm 1, community discovery was performed on the four data sets, and the results were evaluated using F_{same}, *NMI*, and Q. However, since the selection of the center point of the K-means algorithm is random, where the weight matrix of neurons in the hidden layer and the output layer for the depth sparse autoencoder is also initialized by random numbers, the proposed algorithm is random. To be able to react to the smoothing of the data, this paper investigated the fluctuation of the data, and the results are displayed in Fig. 6.

The variance of each data set is shown in Table 3.

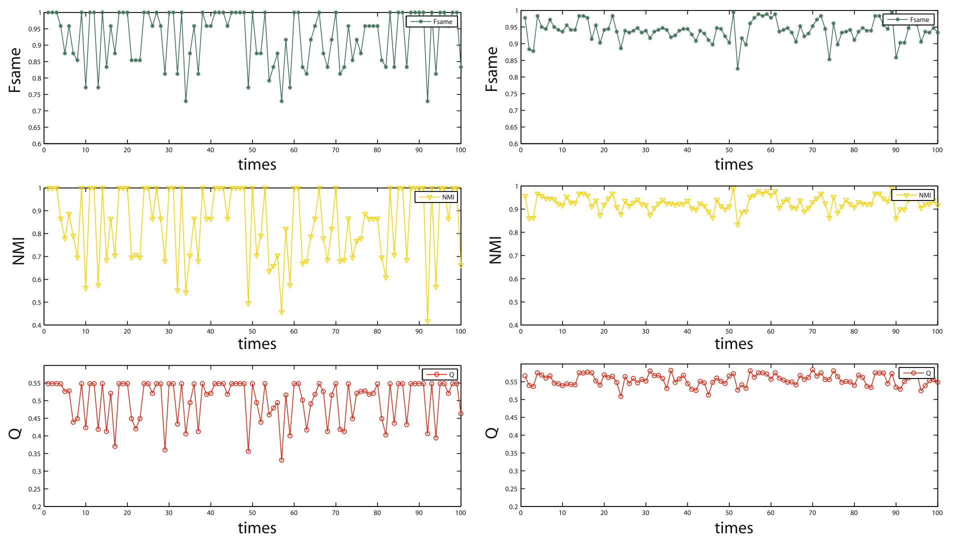

Fig. 6 Clustering results for the Strike dataset (left) and the Football dataset (right)

Table 3 Fluctuation variance of the proposed algorithm

	Strike			Football		
Evaluation standard	F_{same}	NMI	Q	F_{same}	NMI	Q
variance (unit: 10^{-3})	6.93	26.46	3.54	0.98	0.96	0.26

By performing 100 experiments on the Strike data set and the Football data set, Fig. 6 and Table 3 show that the clustering results have volatility. Taking the *NMI* value as an example, the small data set [36] has a variance of 26.46, and the larger data set [37] has a variance of 0.96, showing that multiple experiments are needed in small data sets to reduce the impact of fluctuations. In addition, the variance will decrease with the increase in the data set. This result also indicates that the algorithm has higher stability on the large data set, and the repetition number of the different experimental sets can be flexibly changed.

Table 4 describes the comparison of parametric experimental cluster results. The table shows that proved deep sparse autoencoder for community detection can significantly improve the cluster results and quality.

3.5 Algorithm comparison

In this experiment, the K-means algorithm and the similarity matrix were directly clustered, the CoDDA algorithm and the DSACD were compared, and the *NMI* value was used for evaluation. The hop count threshold of the CoDDA algorithm, the attenuation factor, and the value of the deep sparse autoencoder use the optimal values in Table 5. The Table 6 shows experimental results.

Note: the number of iterations of the Football and Strike datasets is 100, and the number of iterations of the LiveJournal dataset is 5.

As Table 6 shows, the DSACD has higher cluster accuracy and cluster quality, among which the selection of back-propagation algorithm, the CoDDA algorithm, and DSACD algorithm results achieve higher precision, which is consistent with the results of the paper [25]. To compare the differences, Table 7 lists several error values for the last iteration of the two back-propagation algorithms. The results show that the CoDDA algorithm reduced the error to 17 in the process of minimizing the reconstruction error, but the DSACD finally decreases to 7.9. Both algorithms provide better performance.

Finally, the DSACD is compared with the LPA [3] in Table 8. Then, the DSACD is significantly better than the LPA.

However, due to the characteristics of deep learning, it requires more time during training, as shown in Table 9. In large data sets, computing time needs to grow exponentially.

Table 4 The comparison of parametric experimental cluster results

	Repeat times	F_{same}	NMI	Q
Strike (front)	100	0.71	0.14	– 0.01
Strike (after)	100	**0.96**	**0.92**	**0.51**
Football (front)	100	0.67	0.15	0.66
Football (after)	100	**0.94**	**0.92**	**0.55**
LiveJournal (front)	5	0.61	0.01	0.00
LiveJournal (after)	5	**0.88**	**0.82**	**0.79**
com-orkut (front)	5	0.25	0.10	0.17
com-orkut (after)	5	**0.94**	**0.92**	**0.85**

Table 5 Parameter comparison before and after in parameter experiments

	Repeat times	*S*	σ	*T*
Strike (front)	100	1	0.1	1
Strike (after)	100	**2**	**0.7**	**1**
Football (front)	100	1	0.1	1
Football (after)	100	**2**	**1**	**2**
LiveJournal (front)	5	1	0.1	1
LiveJournal (after)	5	**6**	**0.4**	**4**
com-orkut (front)	5	6	0.1	2
com-orkut (after)	5	**7**	**0.2**	**5**

Table 6 The analysis of community detection results

Algorithm	Strike *NMI*	Football *NMI*	LiveJournal *NMI*	com-orkut *NMI*
K-means	0.82	0.67	0.35	0.3
Hop	0.89	0.90	0.66	0.58
CoDDA	0.91	0.93	0.82	0.77
DSACD	0.93	0.93	0.82	0.92

Table 7 Reconstruction error table of CoDDA and DSACD

CoDDA	DSACD
...	...
$2.28428e+01$	$8.41571e+00$
$1.85591e+01$	$8.13560e+00$
$1.71101e+01$	$8.43235e+00$
$1.71101e+01$	$7.90684e+00$

Table 8 Comparison with other community discovery algorithm cluster results

Algorithm		LPA	DSACD
Strike	F_{same}	**0.96**	**0.96**
	NMI	0.87	**0.93**
	Q	**0.55**	0.52
Football	F_{same}	0.26	**0.94**
	NMI	0.87	**0.93**
	Q	**0.75**	0.56
LiveJournal	F_{same}	0.49	**0.85**
	NMI	0.76	**0.85**
	Q	0.79	**0.80**
com-orkut	F_{same}	0.57	**0.94**
	NMI	0.61	**0.92**
	Q	0.74	**0.85**

Table 9 Time comparison with other community detection algorithms

Algorithm	Strike(s)	Football(s)	LiveJournal(h)	com-orkut(h)
LPA	0.008	0.020	0.01	0.01
CoDDA	0.011	2.561	4.59	6.21
DSACD	0.009	1.091	3.35	5.37

The weakness of the CoDDA is also shown, and the calculation time is much longer than the DSACD calculation time. The CoDDA requires up to a week or weeks to calculate a larger matrix [13]. In addition, although the calculation time of the DSACD is small, it requires a large amount of memory, and it requires at least 128 G of memory for the network of tens of thousands of nodes, but the CoDDA can normally be calculated on a normal configuration computer.

3.6 Parameter analysis

The deep sparse autoencoder for community detection(DSACD) contains three important parameters: the hop threshold (S) in the similarity matrix, the attenuation factor (σ), and the number of layers in the deep sparse autoencoder (T). These three parameters have a direct impact on the clustering results. This section sets up experiments to find the optimal parameters. The experimental procedure is shown as follows. First, a value is preselected for each parameter. Then, the experiment is performed according to the hop count threshold, the attenuation factor, and the layers order of the deep sparse autoencoder, and each experiment is repeated 5 or 100 times. Then, the optimal results are used as the peremeters in the next experiments. After the first experiments, the three optimal values obtained are reused as the experimental input values applied to adjust the obtained parameters. After the end of the second round, the optimal parameters obtained are output.

Each parameter value is initialized as s=1, sigma=0.1, and T=1. The selection of each parameter is random during initialization, and the minimum value is selected as the starting parameter.

The first round of results from the above table is shown in Fig. 7.

Figure 7 shows that the proved deep sparse autoencoder for community detection has a better clustering effect than the K-means clustering algorithm, but the significance of

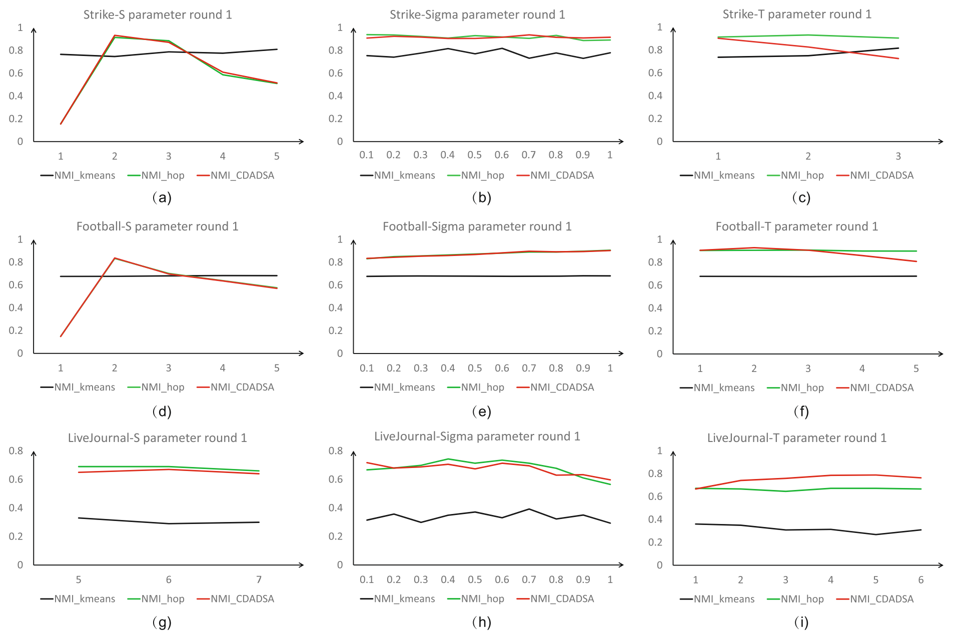

Fig. 7 The first-round experimental results of the parameter experiment

deep learning in the Strike and Football datasets does not seem obvious. The clustering results of the proved deep sparse autoencoder for community detection and the clustering results of the similarity matrix are similar, and the gap is not obvious in the parameter experiment of the attenuation threshold or the layer number. However, after a round of experiments on big data sets, the advantages are already evident.

The second-round results are shown in Fig. 8.

After the second-round parameter experiment, we find that the similarity matrix clustering result of the Strike dataset is the best. The proved deep sparse autoencoder for community detection does not improve the clustering result in the process of deep learning, but decreases the result, as shown in Fig. 8c). Therefore, in small datasets, the similarity matrix is utilized to process the adjacency matrix of the graph.

As shown in the Football dataset, the proved DSACD slightly improves the clustering results in the process of deep learning, and the highest value is obtained by the proved deep sparse autoencoder for community detection. The second round of results is significantly better than the first round of clustering quality.

Meanwhile, as shown in the LiveJournal dataset, the accuracy of the proved DSACD is significantly improved on the big data set. After deep learning, the NMI value is increased from 0.7111 to 0.8171, and the degree of improvement is approximately 13%. On the other hand, the NMI top value gradually increases during the course of parameter experiment and reflects the necessity and superiority of deep learning.

In Table 5, the parameter value of S, σ, and T before and after experiments are tested in four data sets. In Strike and Football, the average value of every parameter are detemined with the repeat experimental times are 100, but in LiveJournal, the detemined average value of every parameter only needs 5 repeat experimental times. The repeat experimental times are for comparing with the CoDDA in the same parameter standard.

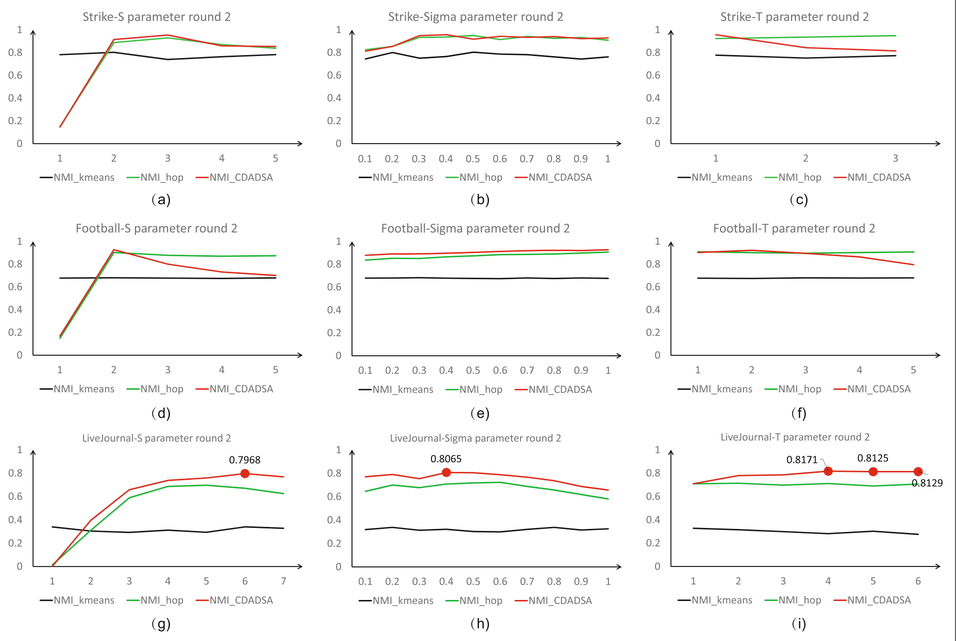

Fig. 8 The second-round experimental results of the parameter experiment

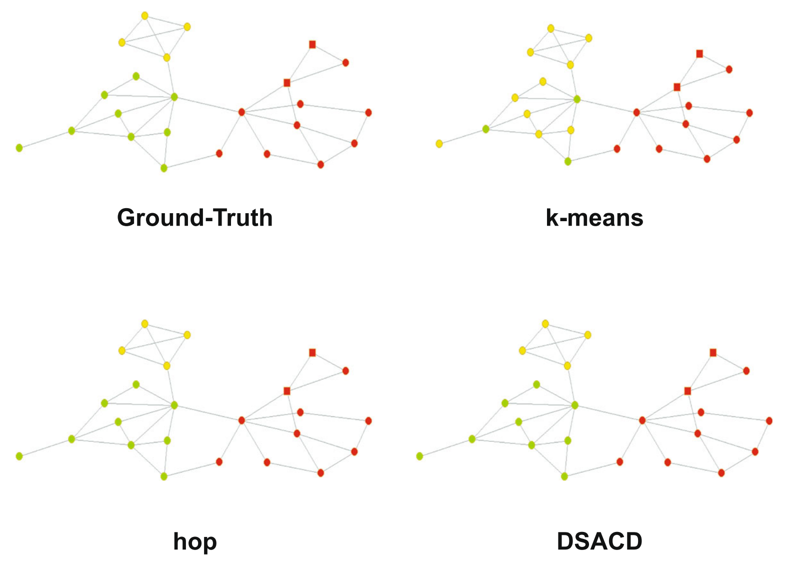

Fig. 9 Comparison of the actual results of the strike dataset and different cluster results

4 Result and discussion

4.1 Visualization results

This experiment is based on the real dataset (Ground-Truth), K-means algorithm, the hop-based clustering method (hop), CoDDA, and the DSACD, which are visual comparison. Intuitively, the cluster information between different communities is observed and evaluated. Figures 9 and 10 show the results. The same color represents the same community, and different colors represent different communities.

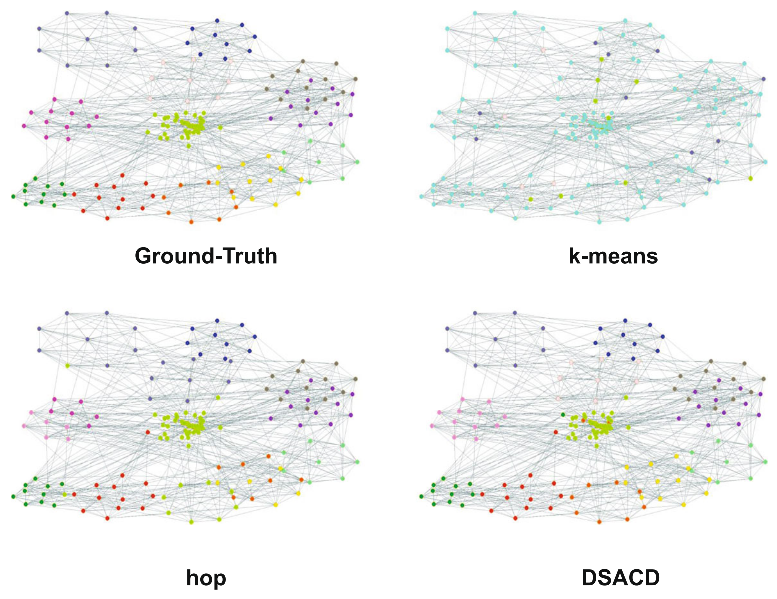

Fig. 10 Comparison of the true results of the football dataset and different clustering results

As seen from Fig. 9, the cluster results of the K-means algorithm are not accurate, and the green communities are basically clustered into yellow communities, which obviously does not conform to the real situation. Both the hop-based and the deep sparse autoencoder-based algorithms (DSACD) cluster accurate results, which are in good agreement with the real-world results [38], indicating the accuracy of the algorithm. Furthermore, because the hop-based algorithm (hop) and the deep sparse autoencoder algorithm (DSACD) are consistent, it shows that the algorithm has little meaning in deep learning of small data sets. The parameters are not properly selected or even lost information. In the data set, the hop count threshold and the attenuation factor should be focused on improving the cluster effect.

As Fig. 10 shows, the K-means algorithm cannot be accurately clustered in the 180-node dataset. Note that the K-means algorithm can be used in small maps but cannot be clustered in a slightly complex network because the adjacency matrix still carries much information that leads to unclear community boundaries. After adding the similarity matrix processing, the nearby nodes are connected, so the clustering quality is significantly improved. For the dataset, because the community structure is very obvious, the similarity matrix can obtain good clustering results, and the community structure is initially calculated but will still be confused in the community with close distance. After deep learning, the community clustering with obvious community structure is almost all successful, and there are still fewer nodes with cluster failure for clusters without community structure.

According to Fig. 11, there are 8 communities in the LiveJournal dataset, and a monster community appears for the K-means algorithm clustering. The hop-based processing [13] can successfully cluster nodes with obvious community structure, but for two communities with higher similarity, that is, the situation with numerous edges between communities cannot be correctly handled, and the connection between two communities with a close relationship can easily be clustered into a third community. As shown in

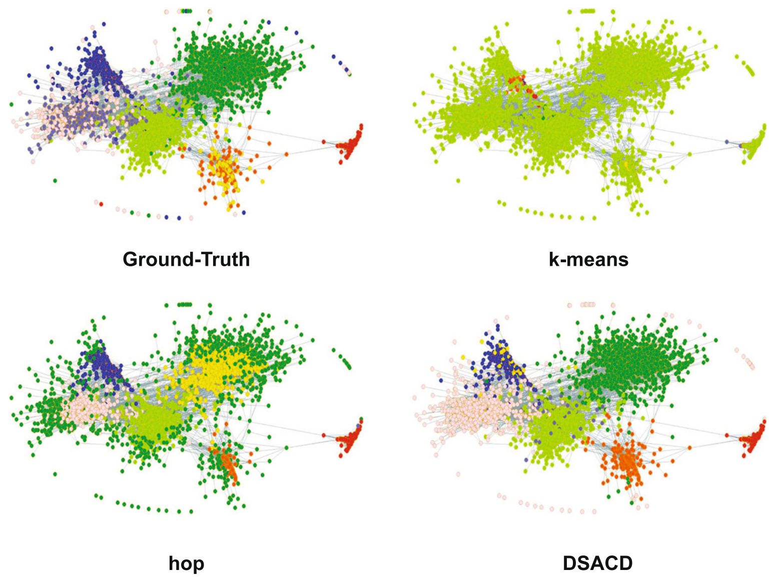

Fig. 11 The comparison of real results about LiveJournal datasets and different cluster results

Fig. 12, the yellow community section should be green. In addition, for nodes with fewer neighbors, the green node group on the left side of the figure should be pink, but there is no successful cluster. After learning the network features, the left green node group and the pink node group are successfully merged in the DSACD diagram. The clustering accuracy is further improved for the green node community, and finally, the community with the obvious community structure is clustered.

The experiments in Section 3.5, which has compared the time and results of DSACD and CODDA. To visualize the results of the two algorithms, the LiveJournal dataset is extracted, and the results of the two algorithms are visually compared. For the deep sparse autoencoder for community detection, except for the four communities with a high coupling degree in the upper right and upper left, the other four communities are clustered accurately. The same situation appears in the CoDDA. Four communities are obviously clustered successfully. At the same time, some mistakes emerge in the course of clustering such as one community clustered into two communities and two communities clustered into one. Compared comprehensively, the effect of the two algorithms is close.

The clustering method based on the K-means algorithm is random, especially in small datasets because the boundary between two communities is not easy to judge. The clustering results will change due to the selection of initial points, especially the time selected of the influential points will directly affect the clustering results. For small data sets, the results need to be averaged many times, and the final results will converge at a certain value. With the increase in datasets, the size of community groups also expands, and the proportion of nodes in the border area relative to the entire map also decreases, so the clustering results tend to be stable.

In the community detection algorithm based on the deep sparse autoencoder and L-BFGS, parameters need to be selected. Among these parameters, the hop threshold of the small-scale network is smaller. Because the size of the community is small, the influence of the relationship between nodes is smaller. The appropriate hop threshold can be taken from 2 to 3, and the calculation requires less time. For a large-scale network, the threshold of hops will increase correspondingly. At this time, the community structure is obvious, and the scale is large, and the relationship between nodes is complex, so the corresponding influence will increase. The threshold of hops can be taken from 6 to 8. At the same

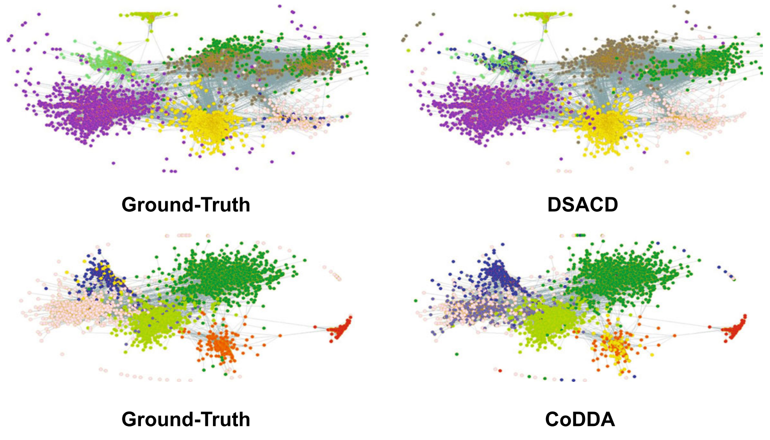

Fig. 12 Comparison of the CoDDA and the DSACD about the LiveJournal dataset

time, large-scale datasets need to be reduced several times to improve the accuracy of the community feature extraction. However, regardless of any parameter, it should not be too large or too small; otherwise, it will lead to data redundancy or missing data. This situation also demonstrates the necessity of the parameter experiment and also illustrates the necessity of conducting a parametric experiment.

When compared with other algorithms or with itself, the DSACD has higher accuracy. The similarity matrix plays a dominant role in small data sets. On the large dataset, a further dimension reduction operation based on the deep sparse autoencoder is needed for feature extraction. In the process of training, the CoDDA or the DSACD can be used for back-propagation. The CoDDA has the characteristics that it does not need to calculate the Hessian matrix and saves memory, but it takes a long time. The DSACD is characterized by more accurate calculation, but it requires more memory. In large data sets, the program may crash due to insufficient memory. It is necessary to determine in advance whether the hardware configuration meets the requirements. The accuracy of the two algorithms is slightly higher than the accuracy of the CoDDA algorithm.

Through the visualization software [42], the results of the K-means clustering can hardly be separated from the community, resulting in the emergence of a giant community that is the monster community. After calculating the similarity matrix, the community structure appears. Finally, dimension reduction by the deep sparse automatic encoder can separate the similar communities more accurately and further improve the clustering accuracy.

4.2 Application on the indoor positioning system

In this section, an application test with the benchmarking data of our indoor positioning systems is designed. Figure 13 depicts a public place in our library, in which the total area is $100m^2$ and area measurement is $64m^2$. Four APs are installed in four corners, and 64 points are set in the place. There are 9379 records of RSSI collected by smartphone—MI note2. Our DSACD can be used to gather 64 communities, and then, the distance between every points and each AP can be obtained by the log-distance path loss model [43]. Formula 15 is a logarithmic distance ranging model for indoor wireless signal transmission.

$$RSSI(d)_{dB} = RSSI(d_0) - 10\beta \lg\left(\frac{d}{d_0}\right) + \epsilon \tag{15}$$

In formula 15, $RSSI(d_0)$ represents the signal intensity when distance is between AP and signal source, and ϵ is a random variable that obeys normal distribution $(\epsilon; N(0, \sigma dB^2))$. β represents the path loss factor, and the indoor environment is usually set to 3 or 4.

Suppose the distance between the two APs and the signal source is d_1 and d_2, and the signal intensity difference among them is ΔdB.

$$RSSI(d_1) = RSSI(d_0) - 10\beta \lg\left(\frac{d_1}{d_0}\right) + \epsilon_1 \tag{16}$$

$$RSSI(d_1) + \Delta dB = RSSI(d_0) - 10\beta \lg\left(\frac{d_2}{d_0}\right) + \epsilon_2 \tag{17}$$

Since two APs are in the same localization area, let $\epsilon_1 = \epsilon_2$. Formula 16 is subtracted from formula 17, and the result is as follows:

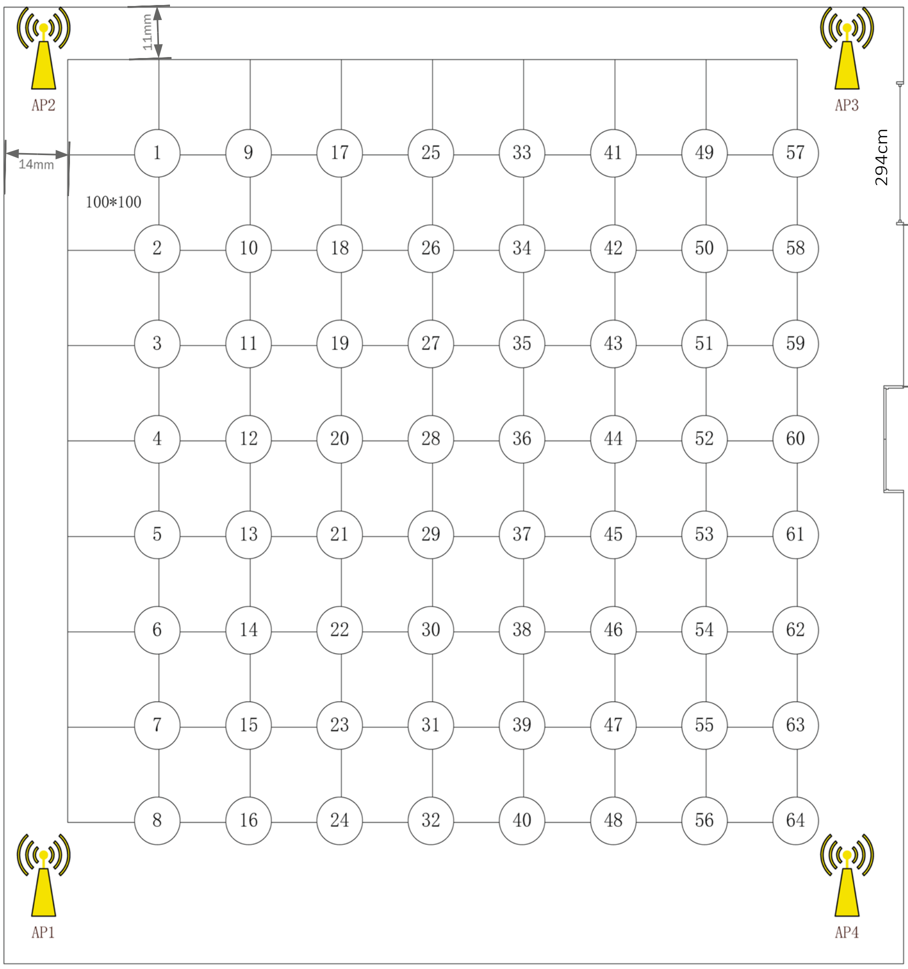

Fig. 13 The test scenario in our library

$$\Delta dB = 10\beta \lg \left(\frac{d_2}{d_1}\right) \tag{18}$$

Convert formula 18 to formula 19:

$$d_2 = 10^{\frac{\Delta dB}{10\beta}} d_1 \tag{19}$$

As shown in formula 19, ΔdB describes the distance relationship between the two APs. As shown in formula 20, $FingerPrint_i$ represents the signal intensity between the ith AP and m signal source. $MinFingerPrint_i$ represents the weakest signal intensity between the ith AP and jth signal source.

$$FingerPrint_i = \{RSSI_0, RSSI_1, \cdots, RSSI_i, \cdots, RSSI_m\} \tag{20}$$

$$MinFingerPrint_i = \min(FingerPrint_i) \tag{21}$$

The difference between $FingerPrint_i$ and $MinFingerPrint_i$

$$FingerPrint'_i = \left\{RSSI'_0, RSSI'_1, \cdots, 0, \cdots, RSSI'_m\right\} \tag{22}$$

According to formulas 19 and 22, formula 23 is as follows:

$$FingerPrint_i'' = \left\{ 10^{\frac{RSSI_0'}{10\beta}} d_1, 10^{\frac{RSSI_1'}{10\beta}} d_1, \cdots, d_1, \cdots, 10^{\frac{RSSI_m'}{10\beta}} d_1 \right\} \tag{23}$$

The form of formula 23 after normalization is as follows:

$$FingerPrint_i''' = \left\{ 10^{\frac{RSSI_0'}{10\beta}}, 10^{\frac{RSSI_1'}{10\beta}}, \cdots, 1, \cdots, 10^{\frac{RSSI_m'}{10\beta}} \right\} \tag{24}$$

Formula 24 represents the distance fingerprint of *i*th AP, and these distance fingerprints constitute the "distance fingerprint map" of the location area.

The geometric meaning of fingerprint is the distance between the reference point and AP. Fingerprint localization model is divided into two stages: offline and online. During the offline stage, the localization area is divided into different clusters by DSACD, using the technique proposed in this article and the binary classification for APs by K-means algorithm in each subarea to select available APs. During the online stage, subareas where the object points exist are selected using NN algorithm. Then, the coordinate of the target point is calculated.

The DSACD are used in offline stage. The selected 64 test points as shown in Fig. 13, while the fingerprint database are divided 64 subareas with 64 reference points as the centroids. The operation process is as follows: the collected signals in the whole region are transformed into fingerprints, and then, the fingerprints are as inputs of DSACD to realize regional classification. These effective fingerprint components are extracted from all the fingerprints in the subregion, then the fingerprint is transformed into distance fingerprint according to the fingerprint transformation model, and finally, the fingerprint database of the subregion is formed.

Figure 14 indicates the average errors of the distance between every point and each AP. The average distance error between 64 points and 4 AP shows normal distribution that is according with the laws of nature; meanwhile, a loop occurs at every eight points that is according with the law of collection. It is shown in Fig. 13 that every column has 8 points. In addition, the 64 average errors show that nodes with distance error less than 0.5 from AP1 account for 26.6% of the total number of nodes, nodes with distance error between 1.5 and 2 from AP1 account for 15.63% of the total number of nodes, nodes with distance error less than 0.5 from AP2 account for 21.88%, nodes with distance error between 1.5 and 2 from AP2 account for 10.94%, nodes with distance error less than 0.5 from AP3 account for 21.88%, nodes with distance error between 1.5 and 2 from AP3 account for 15.63%, nodes with distance error less than 0.5 from AP4 account for 10%, and nodes with distance error between 1.5 and 2 from AP4 account for 21.88%. For the 4 APs, nodes with higher distance error accuracy are reaching 21.88% which are from AP2/AP3, nodes with lower distance error are reaching 21.88% which is from AP4. During the process of measuring, there is voltage signal interference near AP4, whose reasonableness is confirmed by the calculation results.

In the collection environment, factors that have strong impacts for measured data are the temperature, angle, humidity, and crowd density. Sixty-four communities are gathered by DSACD, and then, the log-distance path loss model is used in every community to obtain the distance between every point and each AP. The achieved average errors can satisfy the necessary of location, which has a certain reference significance for the real-time of future research intelligent navigation positioning.

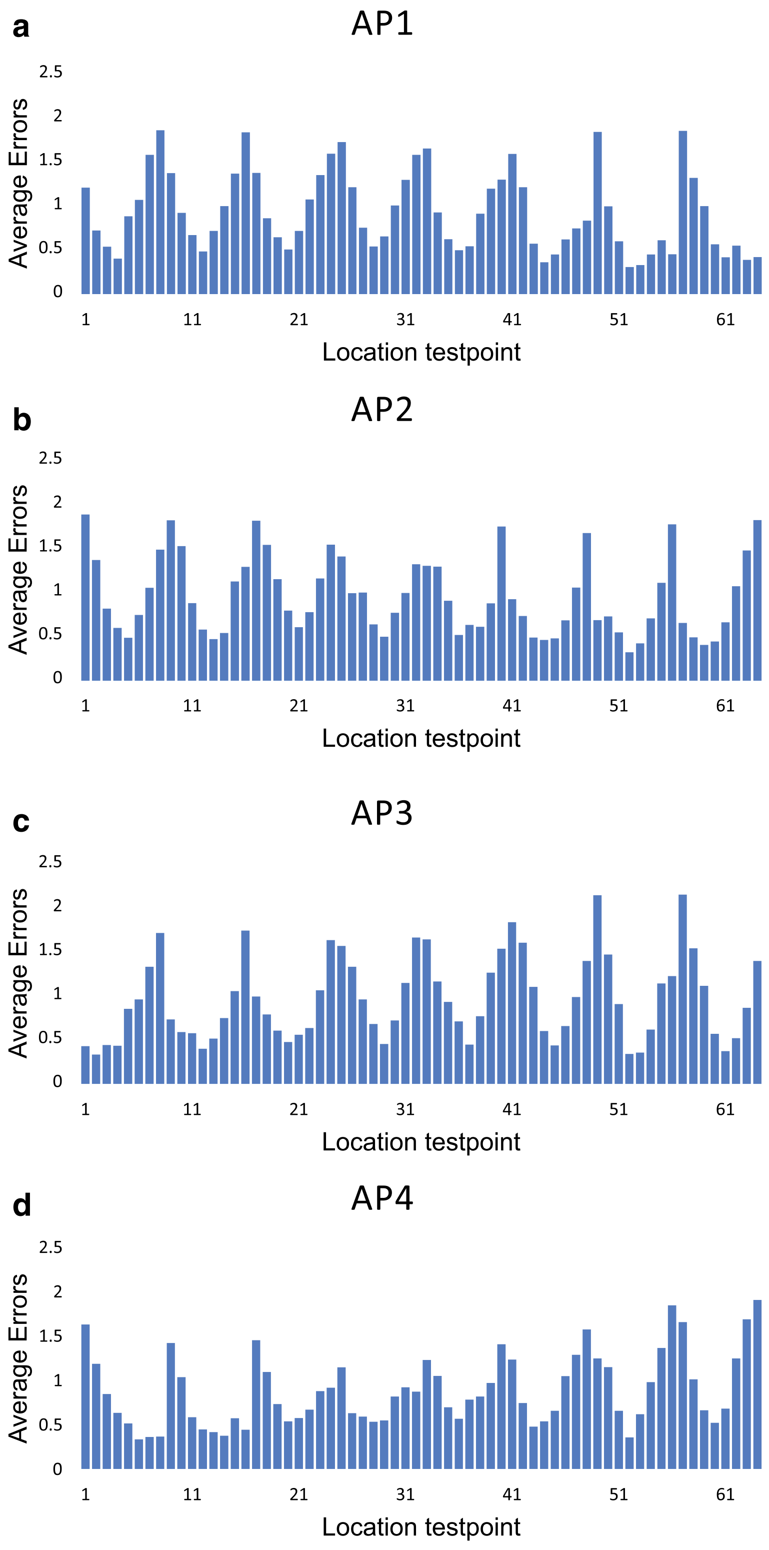

Fig. 14 The average errors of location test points

5 Conclusion

This paper proved a novel deep sparse autoencoder-based community detection (DSACD) and compares it with K-means, Hop, CoDDA, and LPA algorithm. Experiments show that for complex network graphs, dimensionality reduction by similarity matrix and deep sparse autoencoder can significantly improve clustering results.

Several issues persist and require further research. The similarity matrix calculation increases with the matrix size, which lead to a large memory consumption and high requirements for experimental equipment. Too many temporary variables in the back-propagation algorithm will also consume memory. Decomposition strategy for large matrix in similarity calculation should be expected in future studies.

Acknowledgements
The authors are extending their kind gratitude for the new ideas in our discussion from Fangying Song.

Authors' contributions
Rong Fei and Jingyuan Sha proposed the main idea about the deep sparse autoencoder on community detection, completed the simulation, and analyzed the result. Qingzheng Xu carried out the indoor location studies. Bo Hu participated in the compared community detection algorithms. Kan Wang and Shasha Li helped to modify the manuscript. The authors read and approved the final manuscript.

Authors' information
RONG FEI is an Associate Professor in Xi'an University of Technology. Her main research interests are Community detection, stochastic opposition algorithm, and location-based services.

Funding
This research was partially supported by the National Key Research and Development Program of China (2018YFB1201500), National Natural Science Foundation of China (under Grants nos. 61773313), and the Project of Technological Planning Project of Beilin District of Xi'an (GX1819).

Availability of data and materials
The experiment used four real data sets are Strike [36], Football [37], LiveJournal [38], and Orkut [39]. Strike is a 24-striker relationship table on wood processing projects. The frequency of discussion for strike topics between two people is the rules, which are added. If the frequency is high (there are specific criteria for evaluation during the investigation, no detailed explanation will be given here), then, a connection is established. Football is the timetable for the American Football Cup (FBS) held by the American College Sports Association (NCAA) in 2006. In the NCAA relationship network, if two teams played games, the connection is established. Orkut is a social service network launched by Google.

Competing interests
The authors declare that there is no conflict of interests regarding the publication of this paper.

Author details
[1]Xi'an University of Technology, JinHua Road No.5, Xi'an, China. [2]College of Information and Communication, National University of Defense, Dü8 east zhangba road, Xi'an, China. [3]Beijing Huadian Youkong Technology Co., Ltd., TianXiu Road No.10, Beijing, China.

Received: 3 January 2020 Accepted: 14 April 2020
Published online: 11 May 2020

References
1. S. Fortunato, Community detection in graphs. Phys. Rep. **486**(3-5), 75–174 (2010)
2. M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. **15**(6), 1373–1396 (2003)
3. U. N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks. Phys. Rev. E. **76**(3), 036106 (2007)
4. S. Gregory, Finding overlapping communities in networks by label propagation. New J. Phys. **12**(10), 103018 (2010)
5. K. Guo, W. Guo, Y. Chen, Q. Qiu, Q. Zhang, Community discovery by propagating local and global information based on the mapreduce model. Inf. Sci. **323**, 73–93 (2015)
6. Q. Zhang, Q. Qiu, W. Guo, K. Guo, N. Xiong, A social community detection algorithm based on parallel grey label propagation. Comput. Netw. **107**, 133–143 (2016)
7. S. E. Schaeffer, Graph clustering. Comput. Sci. Rev. **1**(1), 27–64 (2007)
8. Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives. IEEE Trans. Pattern. Anal. Mach. Intell. **35**(8), 1798–1828 (2013)
9. J. Schmidhuber, Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
10. Y. Lecun, Y. Bengio, G. Hinton, Deep learning. Nature. **521**(7553), 436 (2015)
11. C.-H. Chen, F. Song, F.-J. Hwang, L. Wu, A probability density function generator based on neural networks. Phys. A Stat. Mech. Appl. **541**, 123344 (2019)

12. C.-H. CHEN, IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **103**(1), 265–267 (2020)
13. S. Jing-Wen, W. Chao-Kun, X. Xin, Y. Xiang, Community detection algorithm based on deep sparse autoencoder. J. Softw. **28**(3), 648–662 (2017)
14. Z. Wang, Q. Zhang, A. Zhou, M. Gong, L. Jiao, Adaptive replacement strategies for moea/d. IEEE Trans. Cybern. **46**(2), 474–486 (2015)
15. Z. Wang, Q. Zhang, H. Li, H. Ishibuchi, L. Jiao, On the use of two reference points in decomposition based multiobjective evolutionary algorithms. Swarm Evol. Comput. **34**, 89–102 (2017)
16. Z. Wang, Y.-S. Ong, H. Ishibuchi, On scalable multiobjective test problems with hardly dominated boundaries. IEEE Trans. Evol. Comput. **23**(2), 217–231 (2018)
17. Z. Wang, Y.-S. Ong, J. Sun, A. Gupta, Q. Zhang, A generator for multiobjective test problems with difficult-to-approximate pareto front boundaries. IEEE Trans. Evol. Comput. **23**(4), 556–571 (2018)
18. A. Cauchy, Méthode générale pour la résolution des systemes d'équations simultanées. Comp. Rend. Sci. Paris. **25**(1847), 536–538 (1847)
19. R. Fletcher, C. M. Reeves, Function minimization by conjugate gradients. Comput. J. **7**(2), 149–154 (1964)
20. J. Vlček, L. Lukšan, Generalizations of the limited-memory BFGS method based on the quasi-product form of update. J. Comput. Appl. Math. **241**, 116–129 (2013)
21. B. Molina, E. Olivares, C. E. Palau, M. Esteve, A multimodal fingerprint-based indoor positioning system for airports. IEEE Access. **6**, 10092–10106 (2018)
22. H. Wang, L. Dong, W. Wei, W.-S. Zhao, K. Xu, G. Wang, The WSN monitoring system for large outdoor advertising boards based on zigbee and mems sensor. IEEE Sensors J. **18**(3), 1314–1323 (2017)
23. S. Xia, Y. Liu, G. Yuan, M. Zhu, Z. Wang, Indoor fingerprint positioning based on Wi-Fi: an overview. ISPRS Int. J. Geo-Inf. **6**(5), 135 (2017)
24. P. Dai, Y. Yang, M. Wang, R. Yan, Combination of DNN and improved KNN for indoor location fingerprinting. Wirel. Commun. Mob. Comput. **2019**, 1–9 (2019)
25. Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, A. Y. Ng, in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, On optimization methods for deep learning (Omnipress, Bellevue, 2011), pp. 265–272
26. J. Xu, D. W. Ho, A new training and pruning algorithm based on node dependence and Jacobian rank deficiency. Neurocomputing. **70**(1-3), 544–558 (2006)
27. L. T. Duarte, F. J. V. Zuben, R. R. F. Attux, R. Ferrari, C. M. Panazio, L. N. D. Castro, J. M. T. Romano, in *IEEE Workshop on Machine Learning for Signal Processing*, Mlp-based equalization and predistortion using an artificial immune network (IEEE, Mystic, 2005)
28. J. Yang, G. Xie, Y. Yang, X. Li, in *2018 International Conference on Control, Automation and Information Sciences (ICCAIS)*, A rotating machinery fault diagnosis method for high-speed trains based on improved deep learning network (IEEE, Hangzhou, 2018), pp. 440–444
29. M. Xu, Y. Dong, Z. Li, M. Han, T. Xing, in *2018 37th Chinese Control Conference (CCC)*, A novel time series prediction model based on deep sparse autoencoder (IEEE, Wuhan, 2018), pp. 1678–1682
30. R. Raina, A. Battle, H. Lee, B. Packer, A. Y. Ng, in *Proceedings of the 24th International Conference on Machine Learning*, Self-taught learning: transfer learning from unlabeled data (ACM, New York, 2007), pp. 759–766
31. H. Lee, R. Grosse, R. Ranganath, A. Y. Ng, in *Proceedings of the 26th Annual International Conference on Machine Learning*, Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations (ACM, Montreal, 2009), pp. 609–616
32. H. Lee, P. Pham, Y. Largman, A. Y. Ng, in *Advances in Neural Information Processing Systems*, Unsupervised feature learning for audio classification using convolutional deep belief networks (Neural Information Processing Systems Foundation, Inc. (NIPS), Vancouver, 2009), pp. 1096–1104
33. G. W. Taylor, R. Fergus, Y. LeCun, C. Bregler, in *European Conference on Computer Vision*, Convolutional learning of spatio-temporal features (Springer, Heraklion, 2010), pp. 140–153
34. X. Xue, M. Yao, Z. Wu, A novel ensemble-based wrapper method for feature selection using extreme learning machine and genetic algorithm. Knowl. Inf. Syst. **57**(2), 389–412 (2018)
35. S. Kullback, R. A. Leibler, On information and sufficiency. The annals of mathematical statistics. **22**(1), 79–86 (1951)
36. B. S. Dohleman, Exploratory social network analysis with Pajek. Psychometrika. **71**(3), 605 (2006)
37. L. Sz, Implementation of various criterias for datamining algorithms. https://github.com/luks91/datamining-criteria/tree/master/dataset/. Accessed 16 Nov 2014
38. J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth. Knowl. Inf. Syst. **42**(1), 181–213 (2015)
39. A. Clauset, M. E. Newman, C. Moore, Finding community structure in very large networks. Phys. Rev. E. **70**(6), 066111 (2004)
40. Z. Liu, B. Xiang, W. Guo, Y. Chen, K. Guo, J. Zheng, Overlapping community detection algorithm based on coarsening and local overlapping modularity. IEEE Access. **7**, 57943–57955 (2019)
41. Q. Li, J. Zhong, Q. Li, C. Wang, Z. Cao, A community merger of optimization algorithm to extract overlapping communities in networks. IEEE Access. **7**, 3994–4005 (2019)
42. Program for Large Network Analysis. http://vlado.fmf.uni-lj.si/pub/networks/Pajek/. Accessed 21 Oct 2018
43. J. Li, J. Tian, R. Fei, Z. Wang, H. Wang, Indoor localization based on subarea division with fuzzy C-means. Int. J. Distrib. Sensor Netw. **12**(8), 1550147716661932 (2016)