Knowledge-Based Systems 183 (2019) 104874

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Logic-driven autoencoders☆

Rami Al-Hmouz [a,*], Witold Pedrycz [a,b,c], Abdullah Balamash [a], Ali Morfeq [a]

[a] Electrical and Computer Engineering Department, Faculty of Engineering, King Abdulaziz University, Jeddah, Saudi Arabia
[b] Department of Electrical & Computer Engineering, University of Alberta, Edmonton, Alberta T6R 2V4, Canada
[c] Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland

ARTICLE INFO

Article history:
Received 24 December 2018
Received in revised form 23 July 2019
Accepted 25 July 2019
Available online 28 July 2019

Keywords:
Autoencoder
Logic processing
Deep learning
Fuzzy neurons
AND neurons
OR neurons
Learning
Knowledge representation

ABSTRACT

Autoencoders are computing architectures encountered in various schemes of deep learning and realizing an efficient way of representing data in a compact way by forming a set of features. In this study, a concept, architecture, and algorithmic developments of logic-driven autoencoders are presented. In such structures, encoding and the decoding processes realized at the consecutive layers of the autoencoder are completed with the aid of some fuzzy logic operators (namely, OR, AND, NOT operations) and the ensuing encoding and decoding processing is carried out with the aid of fuzzy logic processing. The optimization of the autoencoder is completed through a gradient-based learning. The transparent knowledge representation delivered by autoencoders is facilitated by the involvement of logic processing, which implies that the encoding mechanism comes with the generalization abilities delivered by OR neurons while the specialization mechanism is achieved by the AND-like neurons forming the decoding layer. A series of illustrative examples is also presented.

1. Introductory comments

Autoencoders arise as one of the fundamental constructs of deep learning which are aimed at the algorithmic formation of a reduced and representative feature space [1,2]. As the essential prerequisite of ensuing processing (classification or prediction) autoencoders generate a high-level feature space in which such processing can be accomplished more efficiently. Autoencoders can be generally categorized into denoising autoencoders [3], contractive autoencoders [4], variational autoencoders [5], and *k*-sparse autoencoders [6], to recall some of the main categories of their architectures.

Commonly, autoencoders are arranged in a cascaded format, which leads to the successive formation of more abstract and condensed (compressed) features. There have been a number of studies focused on various architectural refinements of autoencoders [7,8], their optimization processes [9–11], and application studies [12,13]. Different types of features were fused in a Multimodal deep autoencoder (MDA) where a nonlinear mapping between 2D images and 3D poses were achieved [14]. Moreover, A prediction of energy demand in various situation was investigated by a deep learning model-based autoencoder [15].

A deep learning architecture of convolutional neural network (CNN) [16], was used to develop cross autoencoder in feature learning problems for two levels of social data to obtain a uniform representation across different modalities [17]. In [18], a combination of granular computing and deep learning concept investigated in the context of social manufacturing of service planning using stacked denoising auto-encoder has been introduced. The efficiency of planning tasks of multi granularity services can be improved through the autoencoder in class-imbalanced data. Also, categorical information to build user profiles was exploited by a model built of semantics aware autoencoders for a recommender system [19]. Various autoencoder applications have been investigated by researchers that include for example image binarization [20], feature extraction for speech [21], denoising reduction [22] and image recognition [23].

The main objective of this study is to introduce a concept of logic-based autoencoders along with their architectures and design approaches. A logic-oriented autoencoder realizes knowledge representation and a formation of new features (feature spaces) by logically transforming highly dimensional data yielding some logic expressions articulated through the calculus of fuzzy logic. In spite of the existing diversity of studies on autoencoders being quite often encountered in the setting of deep learning, logic-based constructs have not been studied so far. In comparison with the existing approaches, the underlying logic

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.104874.

* Corresponding author.
E-mail addresses: ralhmouz@kau.edu.sa (R. Al-Hmouz), wpedrycz@ualberta.ca (W. Pedrycz), asbalamesh@kau.edu.sa (A. Balamash), morfeq@kau.edu.sa (A. Morfeq).

https://doi.org/10.1016/j.knosys.2019.104874
0950-7051/

expressions deliver highly attractive aspect of interpretability of the structure of features and the logic relationships among the features. This offers a significant level of originality as the idea of logic-oriented processing has not been pursued in the past although it delivers a highly desirable facet of interpretability and transparency of the obtained results. In this case one gains a useful insight into the buildup of new feature spaces by identifying their origin as a *or*-wise aggregated original features present at the lower level of abstraction (encoding). Similarly, the logic aspect manifests when interpreting the results of decoding through a collection of and aggregation where the specialization (decomposition) of high-level features is completed. The study dwells upon the constructs and practices of neurofuzzy systems, especially fuzzy neurons present throughout the entire architecture. With the gradient-based learning of the connections of the neurons, the logic autoencoder becomes optimized. To enhance the readability of the logic expressions, detailed pruning of the weakest (less essential) relationships expressed by the weights of the autoencoder could be easily completed.

The contributions of the paper are summarized in the following manner:

1- The structure of the autoencoder is represented by its underlying logical relationships.
2- Formation of new feature spaces that are formed from the logic expressions.
3- The process of the construction of the autoencoder is realized in a concise and abstract fashion.

The exposure of the material is arranged in a top-down format. A structure of the logic autoencoder is presented in Section 2. The detailed gradient-based learning scheme is outlined in Section 3. A discussion on the enhancements of the interpretability of the logic expression through pruning is presented in Section 4. Section 5 is devoted to experimental studies. Concluding comments are covered in Section 6.

In the study, we adhere to the standard notation; $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ … are vectors with the entries located in the [0,1] multidimensional hyper cubes. Matrices of connections are denoted by capital letters, say W, V, etc. The logic operations (AND and OR) are implemented with the use of triangular norms (t-norms and t-conorms, respectively) [24,25].

2. Logic autoencoders: main functionality and processing

The inherent advantage of auto-encoders lies in their ability to compress the data and reveal its essential information content. The compression effect could be quantified by considering the ratio h/n, where n stands for the number of features encountered in the original data while h stands for the number of the features being the reduced feature space. Obviously, the lower this ratio is, the higher the achieved compression becomes. Here we introduce, analyze, and design logic-based autoencoders. In contrast to the existing architectures [7,8], the processing realized here is inherently logic-driven, viz. it is exclusively based on *and or*-like mappings. The logic operations commonly encountered in fuzzy sets are implemented with the aid of t-norms and t-conorms, respectively. t-norms are commonly encountered models of *and* logic operations. t-conorms model *or* operations.

An overall structure of a single-layer logic autoencoder is visualized in Fig. 1. Note that the main processing elements building the consecutive layers of the architecture are composed of logic operations.

The two-phase processing is realized as follows.

Encoding Input vector $\boldsymbol{x}$ located in $[0,1]^n$ hypercube is encoded using OR logic expression (realizing OR compression) and yielding

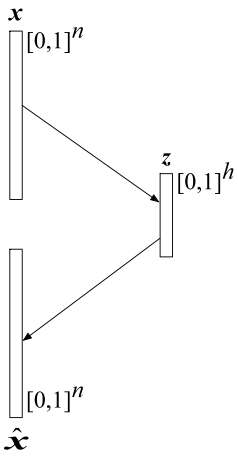

Fig. 1. A generic topology of the logic autoencoder.

a new vector $\boldsymbol{z}$ positioned in $[0,1]^h$ hypercube where $h < n$

$$\boldsymbol{z} = \boldsymbol{x} \circ \boldsymbol{W} \tag{1}$$

viz.

$$z_j = \underset{i=1}{\overset{n}{S}}(w_{ji} t x_i) \tag{2}$$

$j = 1, 2, \ldots, h$. In essence, the processing described by (1) is completed by means of fuzzy OR neurons [26] where $W = [w_{ij}]$ stands for the matrix of adjustable weights (connections) of the neurons. The composition of the inputs with the connections of the neuron is carried out in the form of the s-t composition involving t-norms and t-conorms. It is worth noting that in virtue of the use of the *or* operation, the inputs are combined or wise, which entails that the original inputs are combined together to create a more abstract counterparts than the original features contributing to the realization of the aggregation. The weights describe an impact of the individual features on the result; the higher the value of the weight, the more relevant the corresponding feature in the calculations of the output. In a limit case, the zero weight eliminates the contribution of the corresponding variable stating that it could be dropped.

Decoding The decoding is realized with the use of AND expression (fuzzy AND neuron) returning the reconstructed version of the original input x yielding the output $\hat{\boldsymbol{x}}$

$$\hat{x} = z \bullet V \tag{3}$$

namely

$$\hat{x}_i = \underset{j=1}{\overset{h}{T}}(v_{ij} s z_j) \tag{4}$$

$i = 1, 2, \ldots, n$. Here the t-s composition forming the AND neurons uses t-conorms and t-norms in a inverted order in comparison with the composition forming the OR neurons. In virtue of the t-norms behind the AND neurons, they realize a specialization (refinement) of the previous results of aggregation thus returning original inputs. The two main features of the above architecture can be highlighted here. First, the weights (W and V) are adjustable which makes the system adaptive, in this way, an efficient learning process can be completed. Second, the logic processing makes the structure of the autoencoder interpretable and the results transparent; we focus on this aspect of the topology in Section 4.

Table 1
Selected examples t-norms and dual t-conorms.

t-norms		t-conorms	
Minimum	$a\ t_m b\ = \min(a, b)$	Maximum	$a\ s_m b = \max(a, b)$
Product	$a\ t_p b = ab$	Probabilistic sum	$a\ s_p b = a + b - ab$
Lukasiewicz	$a\ t_l b = \max(a + b - 1, 0)$	Lukasiewicz	$a\ s_l b = \min(a + b, 1)$
Drastic product	$a\ t_d b = \begin{cases} a, & \text{if } b = 1 \\ b, & \text{if } a = 1 \\ 0, & \text{otherwise} \end{cases}$	Drastic sum	$a\ s_d b = \begin{cases} a, & \text{if } b = 0 \\ b, & \text{if } a = 0 \\ 0, & \text{otherwise} \end{cases}$

3. Gradient-based learning

The optimization is completed by adjusting the values of the weight matrices $\boldsymbol{W}$ and $\boldsymbol{V}$ assuming values in the [0,1] interval, say with the aid of the gradient based learning by minimizing the distance between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$. More specifically, for the training purposes, we consider a data set coming in a form of a family of vectors $\boldsymbol{x}(1)$, $\boldsymbol{x}(2)$, …, $\boldsymbol{x}(N)$ while the minimized performance index reads as follows

$$Q = \sum_{k=1}^{N} |\left|\boldsymbol{x}\,(k) - \hat{\boldsymbol{x}}\,(k)\right|\,|^2 \tag{5}$$

where the distance function $\|.\|$ is the Euclidean one. In other words, the explicit detailed formula (5) reads as

$$Q = \sum_{k=1}^{N}\sum_{i=1}^{n}(x_i\,(k) - \hat{x}_i(k))^2 \tag{6}$$

Proceeding with the detailed computing of the gradient-based learning, in which the optimization of Q is realized with regard to the individual entries of W and V, viz. $\min_{W,V} Q$ one arrives at the following well-known update expressions

$$W\ (\text{iter} + 1) = W\ (iter) - \alpha\nabla_w Q$$
$$V\ (\text{iter} + 1) = V\ (iter) - \alpha\nabla_v Q \tag{7}$$

where α is a positive learning rate whose value is adjusted experimentally by achieving a sound compromise between the speed of learning and the stability of the learning process itself. Proceeding with the successive iterations of the learning process we have

$$\frac{\partial Q}{\partial W} = -2\sum_{k=1}^{N}(x\,(k) - \hat{x}(k))\frac{\partial \hat{x}}{\partial W}\nabla_W Q$$

$$\frac{\partial \hat{x}}{\partial W} = \frac{\partial \hat{x}}{\partial z}\frac{\partial z}{\partial W}$$
$$\frac{\partial Q}{\partial V} = -2\sum_{k=1}^{N}(x\,(k) - \hat{x}(k))\frac{\partial \hat{x}}{\partial V} \tag{8}$$

To proceed with further detailed calculations, the corresponding derivatives can be computed once some specific t- norms and t-conorms have been specified. In what follows, we present the results for several commonly used t-norms (minimum, product, Lukasiewicz and the drastic product) and t-conorms (maximum, probabilistic sum, Lukasiewicz and the drastic sum); see Table 1. The computing details of the gradient-based optimization involving t-norms and t-conorms are shown in Table 2

Likewise, the results of derivatives for the dual t-norms and t-conorms of (8) are collected in Table 3.

4. Interpretation of structure revealed through autoencoders

The underlying logic fabric of the constructed autoencoder is beneficial when it comes to the interpretation of relationships between the encoded inputs and the decoding of the following results. In light of the logic operations, we can talk about the logic generalization of some input variables (through the encoding process) and the specialization of the results of aggregation formed through the AND operations used during the decoding process. To facilitate interpretation, it is helpful to concentrate on the most representative (strong) dependencies developed during the design of the autoencoder. This facilitation is achieved by the retaining the most essential weights. In case of OR neurons, the higher the weight, the more important it is. For AND neurons, lower weights are more relevant. To reflect this when revealing the essential structural dependencies of the autoencoder, we introduce two thresholding operations applied to the weights

– for OR neurons

$$w_{ij}^{\lambda} = \begin{cases} w_{ij} & \text{if } w_{ij} \geq \lambda \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

– for AND neurons

$$v_{ij}^{\mu} = \begin{cases} v_{ij} & \text{if } v_{ij} \leq \mu \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

The thresholding operations eliminate the least relevant weights. This improves interpretation but at the same time the reconstruction error increases. Denote this error produced by the trimmed autoencoder by $Q^*(\lambda, \mu)$ with λ and μ standing for the threshold values positioned in the unit interval. Obviously $Q^*(0, 1) = Q$. Higher values of λ and lower values of μ yield higher values of Q^* thus a sound compromise is required. For instance, one can admit such threshold values for which Q^*/Q does not exceed a certain ratio greater than 1. The reduced structure of the autoencoder (after thresholding) can be displayed and structural relationships are easily visualized. In terms of OR neurons, one shows which inputs are generalized to a single entry of $\boldsymbol{z}$. For the AND neuron, one reveals which entries of $\boldsymbol{z}$ are specialized to a given entry of the reconstructed vector.

5. Experimental studies

The experimental studies involve some synthetic data and a collection of publicly available data coming from the Machine Learning repository (archive.ics.uci.edu/m). Our intent is to illustrate the detailed design process and show the performance and interpretation of the results.

Synthetic data

The synthetic data set is composed of a family of $N = 200$ 6-dimensional data generated by the following expressions

$$x_1 = |\sin(0.05k)|$$
$$x_2 = k/N$$
$$x_3 = \exp(-0.02k)$$
$$x_4 = \frac{1}{1 + \exp(-(k - 100))}$$
$$x_5 = \text{trunc}\begin{cases} 0.2 + \text{N}(0, 0.01) \text{ if } k < 120 \\ 0.9 - \text{N}(0, 1) \quad \text{otherwise} \end{cases}$$
$$x_6 = |0.3 + \text{N}(0, 0.1)|$$

$k = 1, 2, \ldots, 200$. The data are presented in Fig. 2.

The inputs of the autoencoder are 12-dimensional as both the original variables as well as their complements are considered.

Table 2
Derivatives of selected *t-norms* and *t*-conorms.

t-norms		*t*-conorms	
Minimum	$\partial(a\ t_m b)/\partial a = \begin{cases} 1 & a < b \\ 0 & a > b \end{cases}$	Maximum	$\partial(a\ s_m b)/\partial a = \begin{cases} 0 & a < b \\ 1 & a > b \end{cases}$
	$\partial(a\ t_m b)/\partial b = \begin{cases} 0 & a < b \\ 1 & a > b \end{cases}$		$\partial(a\ s_m b)/\partial b = \begin{cases} 1 & a < b \\ 0 & a > b \end{cases}$
Product	$\partial(a\ t_p b)/\partial a = b$ $\partial(a\ t_p b)/\partial b = a$	Probabilistic sum	$\partial(a\ s_p b)/\partial a = 1 - b$ $\partial(a\ s_p b)/\partial b = 1 - a$
Lukasiewicz	$\partial(a\ t_l b)/\partial a = \begin{cases} 0 & a + b - 1 < 0 \\ 1 & a + b - 1 > 0 \end{cases}$	Lukasiewicz	$\partial(a\ t_l b)/\partial a = \begin{cases} 1 & a + b < 1 \\ 0 & a + b > 1 \end{cases}$
	$\partial(a\ t_i b)/\partial b = \begin{cases} 0 & a + b - 1 < 0 \\ 1 & a + b - 1 > 0 \end{cases}$		$\partial(a\ t_i b)/\partial b = \begin{cases} 1 & a + b < 1 \\ 0 & a + b > 1 \end{cases}$
Drastic product	$\partial(a\ t_d b)/\partial a = \begin{cases} 1, \text{ if } b = 1 \\ 0, \text{ if } a = 1 \\ 0, \text{ othersie} \end{cases}$	Drastic sum	$\partial(a\ s_d b)/\partial a = \begin{cases} 1, \text{ if } b = 0 \\ 0, \text{ if } a = 0 \\ 0, \text{ othersie} \end{cases}$
	$\partial(a\ t_d b)/\partial b = \begin{cases} 0, \text{ if } b = 1 \\ 1, \text{ if } a = 1 \\ 0, \text{ othersie} \end{cases}$		$\partial(a\ s_d b)/\partial b = \begin{cases} 0, \text{ if } b = 0 \\ 1, \text{ if } a = 0 \\ 0, \text{ othersie} \end{cases}$

Table 3
Derivatives of *t*-norms and *t*-conorms used in the learning scheme.

t-norms		*t*-conorms	
Minimum operation	$\frac{\partial z_j}{\partial w_{ij}} = \begin{cases} 1 & x_i > w_{ij} \\ 0 & x_i < w_{ij} \end{cases}$	Maximum	$\frac{\partial \hat{x}_i}{\partial z_j} = \begin{cases} 1 & z_j > v_{ij} \\ 0 & z_j < v_{ij} \end{cases}$ $\frac{\partial \hat{x}_i}{\partial v_{ij}} = \begin{cases} 0 & z_j > v_{ij} \\ 1 & z_j < v_{ij} \end{cases}$
Product	$\frac{\partial z_j}{\partial w_{ij}} = x_i$	Probabilistic sum	$\frac{\partial \hat{x}_i}{\partial z_j} = 1 - v_{ij}$ $\frac{\partial \hat{x}_i}{\partial v_{ij}} = 1 - z_j$
Lukasiewicz	$\frac{\partial z_j}{\partial w_{ij}} = \begin{cases} 1 & x_i + w_{ij} - 1 > 0 \\ 0 & x_i + w_{ij} - 1 < 0 \end{cases}$	Lukasiewicz	$\frac{\partial \hat{x}_i}{\partial z_j} = \begin{cases} 1 & z_j + v_{ij} < 1 \\ 0 & z_j + v_{ij} > 1 \end{cases}$ $\frac{\partial \hat{x}_i}{\partial v_{ij}} = \begin{cases} 0 & z_j + v_{ij} > 1 \\ 1 & z_j + v_{ij} < 1 \end{cases}$
Drastic product	$\frac{\partial z_j}{\partial w_{ij}} = \begin{cases} 1 & x_i = 1 \\ 0 & \text{othersiwse} \end{cases}$	Drastic sum	$\frac{\partial \hat{x}_i}{\partial z_j} = \begin{cases} 1 & v_{ij} = 0 \\ 0 & \text{otherwise} \end{cases}$ $\frac{\partial \hat{x}_i}{\partial v_{ij}} = \begin{cases} 0 & \text{otherwise} \\ 1 & z_j = 0 \end{cases}$

The learning is carried out with the use of the gradient-based method presented in Section 4; the learning rate α was set to 0.1 (the experiments revealed that higher values of a resulted in some instability of the learning process while too low values of the learning rate produced a very slow convergence of the method). The experiments were carried out for some combinations of *t*-norms and *t*-conorms, namely
case 1: t-nom: product, t-conorms: probabilistic sum
case 2: t-nom: Lukasiewicz, t-conorms: Lukasiewicz
case 3: t-nom: Drastic product, t-conorms: drastic sum

The learning rate α influences the convergence of the optimization process as it can be shown in Fig. 3. Increasing the learning rate (say, $\alpha = 0.5$) might reach the optimal value faster than observed for lower values of α, however it might result in some oscillations during the learning process while decreasing the learning rate (say, $\alpha = 0.01$) requires more iterations to reach the optimal solution. The number of iterations was set to 1000 and we selected $\alpha = 0.1$ for the first 500 iterations and then lower values, say $\alpha = 0.01$ for the remaining of the optimization process to ensure that it is fast enough and no oscillation around the solution occur.

Furthermore, we varied the size of the hidden layer h by starting from $h = 2$ and running up to $h = 15$. The data were split into a learning (70%) and testing (30%) subset. A plot of the learning process ($\alpha = 0.1$) for $h = 4$ is displayed in Fig. 4.

One can conclude that the gradient-based learning has delivered a sound learning mechanisms. There is a visible impact of the values of h on the resulting learning. Fig. 5 shows the performance index Q for different combination of t-norms and t-conorms and different values of h . Also, the error of a traditional autoencoder has been provided and labeled as AE.

Considering the value of h = 4 (which comes as a reasonable choice when a visible drop in the values of the performance index is observed) and choosing the pair of t-norms and t-conorms, the obtained matrices of connections W and V are shown below

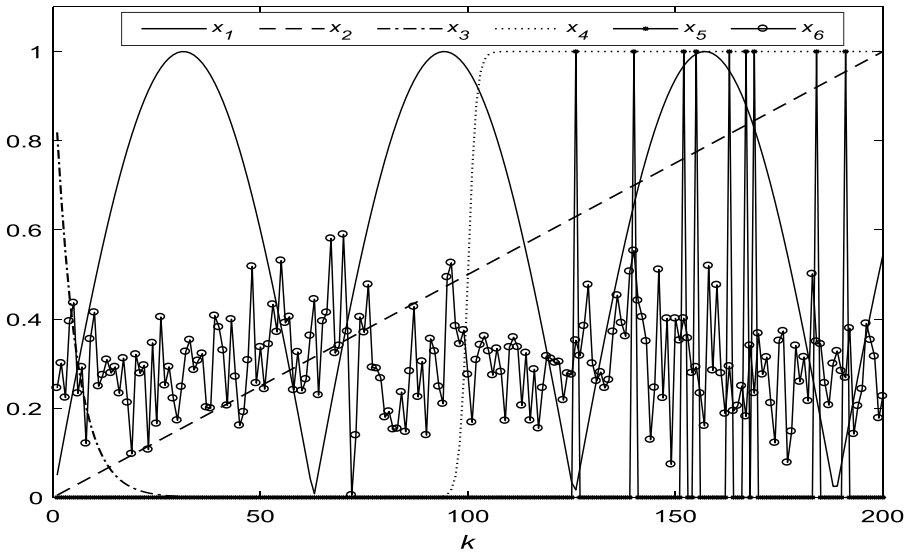

Fig. 2. Data x_1–x_6 used in the design of the autoencoder.

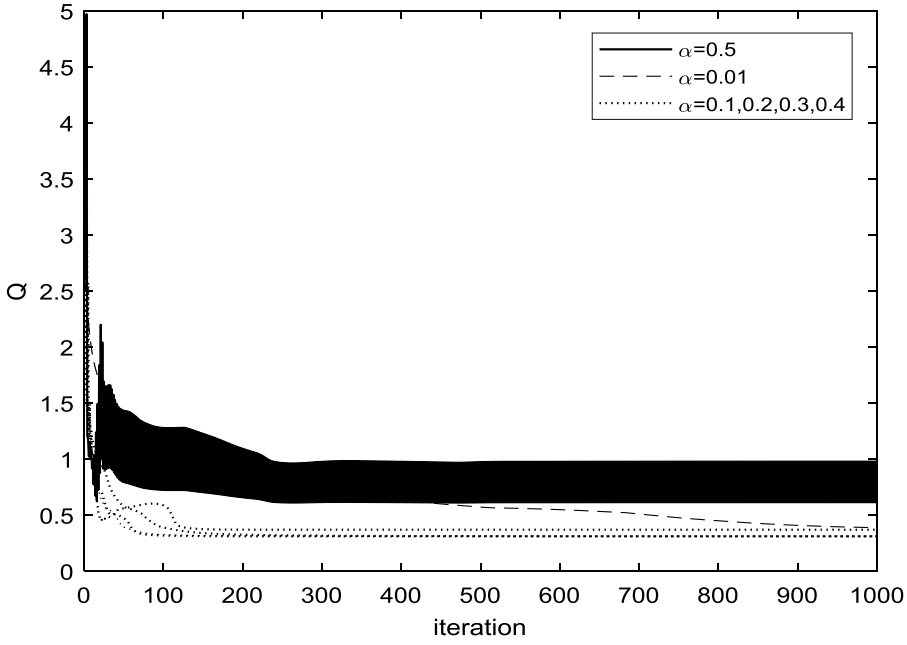

Fig. 3. Performance index Q for different values of learning rate (α).

Case 1:

$$W = \begin{pmatrix} 0.00 & 0.00 & 0.00 & 0.00 \\ \mathbf{0.59} & 0.00 & 0.00 & 0.34 \\ 0.34 & \mathbf{0.45} & \mathbf{1.00} & \mathbf{0.88} \\ \mathbf{0.82} & 0.00 & 0.00 & 0.26 \\ \mathbf{0.93} & \mathbf{0.84} & 0.21 & \mathbf{0.68} \\ 0.00 & \mathbf{0.38} & 0.18 & 0.00 \\ \mathbf{0.35} & 0.00 & \mathbf{0.40} & \mathbf{0.54} \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & \mathbf{0.78} & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 \end{pmatrix}$$

The matrix of $V =$ is given in Box I.

Case 2:

$$W = \begin{pmatrix} 0.00 & 0.00 & 0.10 & 0.07 \\ 0.05 & 0.06 & \mathbf{0.27} & \mathbf{0.74} \\ \mathbf{0.64} & 0.13 & \mathbf{0.42} & \mathbf{0.61} \\ 0.00 & \mathbf{0.62} & 0.08 & \mathbf{0.47} \\ \mathbf{0.49} & \mathbf{0.66} & 0.20 & \mathbf{1.00} \\ 0.07 & \mathbf{0.34} & \mathbf{0.33} & \mathbf{0.54} \\ \mathbf{0.26} & 0.02 & 0.02 & \mathbf{0.76} \\ 0.15 & 0.03 & 0.07 & 0.02 \\ 0.03 & 0.00 & 0.19 & \mathbf{0.27} \\ \mathbf{0.31} & 0.00 & 0.00 & 0.00 \\ 0.00 & \mathbf{0.28} & 0.03 & 0.08 \\ 0.01 & 0.00 & 0.05 & 0.00 \end{pmatrix}$$

The matrix of $V =$ is given in Box II.

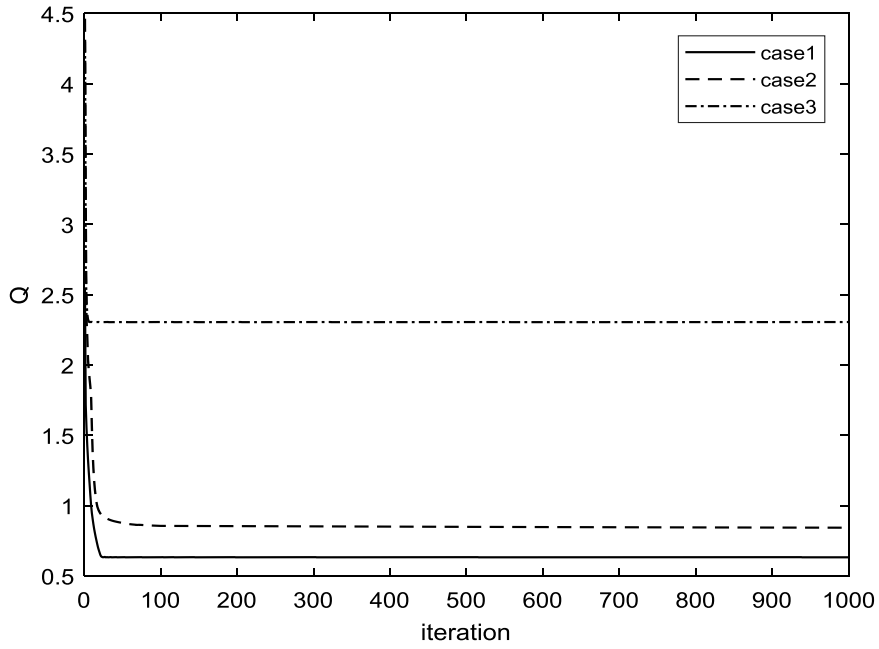

Fig. 4. Performance index Q (training data) obtained in successive iterations of the gradient-based learning.

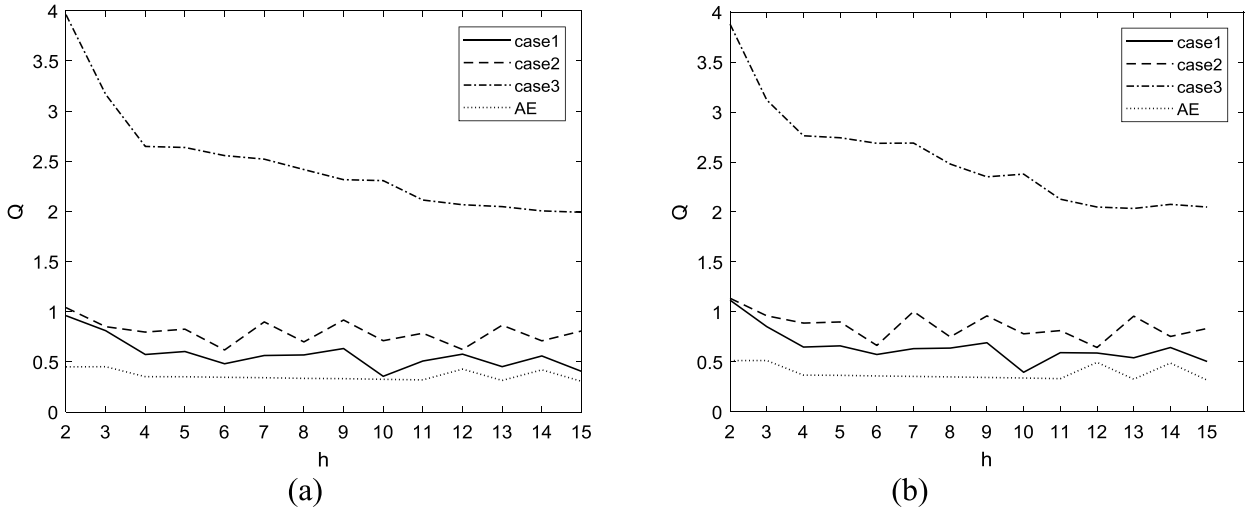

Fig. 5. Performance index using different values of h for (a) training set (b) testing set.

$$V = \begin{pmatrix} 1.00 & \mathbf{0.07} & \mathbf{0.01} & \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} & 0.72 & 1.00 & 1.00 & 0.99 & 1.00 & 0.72 \\ \mathbf{0.57} & 0.62 & \mathbf{0.00} & 0.72 & \mathbf{0.00} & \mathbf{0.00} & 0.66 & 0.98 & 1.00 & 0.81 & 1.00 & 0.70 \\ 0.94 & 1.00 & \mathbf{0.00} & 1.00 & \mathbf{0.00} & 1.00 & \mathbf{0.29} & \mathbf{0.20} & 1.00 & \mathbf{0.00} & 1.00 & 0.64 \\ 1.00 & 0.90 & \mathbf{0.00} & \mathbf{0.33} & \mathbf{0.00} & 0.90 & \mathbf{0.12} & \mathbf{0.55} & 0.97 & 0.90 & 0.95 & 1.00 \end{pmatrix}$$

Box I.

$$V = \begin{pmatrix} 0.98 & 0.51 & 0.70 & 0.63 & \mathbf{0.00} & \mathbf{0.16} & \mathbf{0.32} & \mathbf{0.30} & 0.97 & \mathbf{0.42} & 0.97 & 0.74 \\ 0.73 & 0.96 & 0.74 & \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} & 0.98 & 0.73 & 0.87 & 0.96 & 0.72 & \mathbf{0.48} \\ \mathbf{0.43} & \mathbf{0.40} & \mathbf{0.00} & 0.79 & 0.37 & 0.83 & \mathbf{0.31} & 0.97 & 0.79 & \mathbf{0.39} & 0.90 & 0.52 \\ 0.79 & \mathbf{0.10} & \mathbf{0.00} & 0.69 & 0.78 & 0.59 & \mathbf{0.09} & 0.81 & 0.74 & 0.91 & 0.76 & 0.88 \end{pmatrix}$$

Box II.

Case 3:

$$W = \begin{pmatrix} \mathbf{1.00} & 0.00 & \mathbf{1.00} & 0.00 \\ \mathbf{1.00} & 0.00 & \mathbf{1.00} & 0.00 \\ \mathbf{1.00} & \mathbf{1.00} & \mathbf{1.00} & 0.00 \\ \mathbf{0.81} & 0.00 & \mathbf{0.86} & 0.00 \\ 0.00 & 0.00 & \mathbf{0.97} & \mathbf{0.96} \\ 0.00 & \mathbf{1.00} & \mathbf{1.00} & \mathbf{1.00} \\ \mathbf{1.00} & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & \mathbf{1.00} & 0.00 \\ 0.00 & \mathbf{0.98} & 0.00 & 0.00 \\ 0.00 & 0.39 & 0.06 & 0.42 \\ 0.17 & 0.49 & 0.00 & 0.00 \\ 0.00 & 0.00 & \mathbf{1.00} & \mathbf{1.00} \end{pmatrix}$$

The matrix of $V =$ is given in Box III.

The most significant entries of the matrices of the connections are marked in boldface. To facilitate the readability of the relationships and highlight the most essential linkages, the Boolean version of the autoencoder is formed. The contour plot of values of Q when varying the thresholds λ and μ is shown in Fig. 6.

This leads to the following Boolean expressions governing the dimensionality reduction of the data for $Q^*/Q = 1.1$:

Case 1: ($\lambda = 0.35$ and $\mu = 0.6$)

$z1 = \text{OR}(x_2, x_4, x_5, \bar{x}_1)$
$z_2 = \text{OR}(x_3, x_5, x_6, \bar{x}_4)$
$z_3 = \text{OR}(x_3, \bar{x}_1, \bar{x}_4)$
$z_4 = \text{OR}(x_1, x_5, \bar{x}_1)$

$\hat{x}_1 = z_2$
$\hat{x}_2 = z_1$
$\hat{x}_3 = \text{AND}(z_1, z_2, z_3, z_4)$
$\hat{x}_4 = \text{AND}(z_1, z_4)$
$\hat{x}_5 = \text{AND}(z_1, z_2, z_3, z_4)$
$\hat{x}_6 = \text{AND}(z_1, z_2)$

Case 2: ($\lambda = 0.2$ and $\mu = 0.5$)

$z_1 = \text{OR}(x_3, x_5, \bar{x}_1, \bar{x}_3)$
$z_2 = \text{OR}(x_4, x_5, x_6, \bar{x}_5)$
$z_3 = \text{OR}(x_2, x_3, x_6, \bar{x}_5)$
$z_4 = \text{OR}(x_2, x_3, x_4, x_5, x_6, \bar{x}_1, \bar{x}_3)$

$\hat{x}_1 = z_3$
$\hat{x}_2 = \text{AND}(z_3, z_4)$
$\hat{x}_3 = \text{AND}(z_3, z_4)$
$\hat{x}_4 = z_2$
$\hat{x}_5 = \text{AND}(z_1, z_2)$
$\hat{x}_6 = \text{AND}(z_1, z_2)$

Case 3: ($\lambda = 0.5$ and $\mu = 1$)

$z_1 = \text{OR}(x_1, x_2, x_3, x_4, \bar{x}_1)$
$z_2 = \text{OR}(x_3, x_6, \bar{x}_3)$
$z_3 = \text{OR}(x_1, x_2, x_3, x_4, x_5, x_5, \overline{x_2}, \bar{x}_6)$
$z_4 = \text{OR}(x_5, x_6, \bar{x}_6)$

$\hat{x}_1 = z_4$
$\hat{x}_2 = \text{AND}(z_1, z_4)$
$\hat{x}_3 = \text{AND}(z_1, z_3)$
$\hat{x}_4 = \text{AND}(z_1, z_2, z_3)$

Table 4
Summary of data sets used in the experiments.

Dataset	Size (number of data and dimensionality)	Number of clusters (c)
Iris	150 × 4	3
Car small	100 × 5	7
Wine	178 × 14	4
leaf	340 × 15	6
Ecoli	336 × 8	8

$\hat{x}_5 = \text{AND}(z_1, z_2, z_3, z_4)$
$\hat{x}_6 = \text{AND}(z_1, z_2, z_3, z_4)$

where overbar denotes a complement operation.

With regard to computational time, it was computed across the training process for the synthetic set and for $h = 2$:

case 1: 4.50 s
case 2: 7.67 s
case 3: 13.62 s
AE: 4.26 s

It should be noted for the logic driven autoencoder that some weights will be set to 0 or 1 in the optimization process. This leads to a lower number of computations during the test phase.

Publicly available data sets

The inputs of the logic autoencoder are elements of the [0,1] hypercube. As the data are usually located in R^n, we complete clustering of the data with the use of the FCM algorithm [27] (with the fuzzification coefficient set to 2.0). We consider the data presented in Table 4 (composed of N n-dimensional data). The FCM clustering was completed for c clusters yielding $N * 2c$-dimensional data for both data (u) and their complements ($\bar{u}$) were obtained by (11). The value of c selected according to the available classes in the set. The training set is composed of 70% of data. Selected pairs of t-norms and t-conorms were used along with the varying values of the size of the layer of the OR processing unit. The corresponding plots of Q are displayed in Fig. 7.

$$u_{st} = \frac{1}{\sum_{j=1}^{c} \left(\frac{\|x_t - v_s\|}{\|x_t - v_j\|} \right)^{\frac{2}{m-1}}} \tag{11}$$

$s = 1,2, \ldots, c$; $t = 1, 2, \ldots, N$.

The optimization of the prototypes $\mathbf{v}_i$ is carried out assuming the Euclidean distance between the data and the prototypes that is $\|\boldsymbol{x}_k - \boldsymbol{v}_i\|^2$.

$$v_{st} = \frac{\sum_{k=1}^{N} u_{sk}^m x_{kt}}{\sum_{k=1}^{N} u_{sk}^m} \tag{12}$$

In Fig. 7, a decreasing trend can be noticed for the values of Q as h increases in the three cases. The best performance was achieved for case 1. A slight improvement or no improvement of Q after stabilization is achieved. In terms of *t-norms* and *t-conorms*, different combinations achieve better performance for different data sets.

The plots of Q treated as a function of the two parameters (λ and μ) for different combinations of t-norms and t-conorms for selected values of the thresholds are shown in Fig. 8.

A concise representation of the results coming in the form of the logical topology constructed by the logic autoencoder is shown in Fig. 9 (for the selected data set). The direct input is represented by a solid line while the dotted line represents the complement of the inputs. The relationships are displayed between the input layer and hidden layer are realized with the

$$V = \begin{pmatrix} 1.00 & \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} & 1.00 & \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} \\ 1.00 & 1.00 & 1.00 & \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} & 1.00 & 1.00 & 1.00 & \mathbf{0.00} & \mathbf{0.00} \\ 1.00 & 1.00 & \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & \mathbf{0.00} \\ \mathbf{0.00} & \mathbf{0.00} & 1.00 & 1.00 & \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} & 1.00 & \mathbf{0.00} & 1.00 & \mathbf{0.00} & 1.00 \end{pmatrix}$$

Box III.

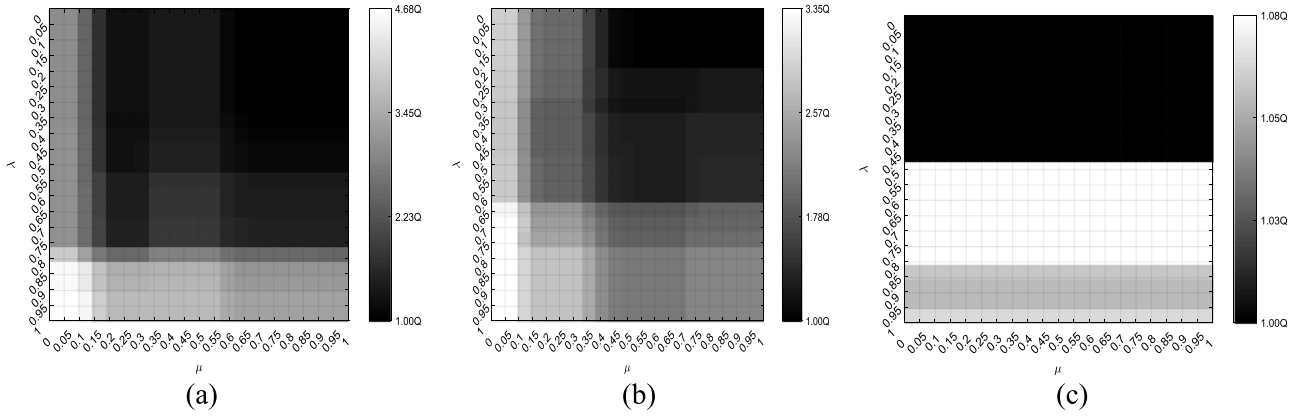

Fig. 6. Contour plots of Q regarded as a function of thresholds λ and μ ($h = 4$) for (a) case 1 (b) case 2 (c) case 3.

aid of some t-norm while the dependencies between the hidden layer and the output layer are captured in terms of the corresponding t-conorms. The input–output logical representation of the autoencoder for the Wine ($h = 3$, c = 4) and Iris ($h = 2$, c = 3) data sets for the three cases is described as follows:

Wine data set:	Iris data set:
Case 1: ($\lambda = 0.35$ and $\mu = 0.6$)	Case 1: ($\lambda = 0.4$ and $\mu = 0.7$)
$z_1 = \text{OR}(x_2, x_3, \overline{x}_3)$	$z_1 = \text{OR}(x_1, \overline{x_2}, \overline{x}_3)$
$z_2 = x_2$	$z_2 = \text{OR}(x_2, x_2)$
$z_3 = x_4$	
$\hat{x}_1 = \text{AND}(z_1,\ z_2,\ z_3)$	
$\hat{x}_2 = \text{AND}(z_1,\ z_2)$	$\hat{x}_1 = \text{AND}(z_1,\ z_2)$
$\hat{x}_3 = \text{AND}(z_1,\ z_2,\ z_3)$	$\hat{x}_2 = z_2$
$\hat{x}_4 = z_3$	$\hat{x}_3 = z_2$
Case 2: ($\lambda = 0.2$ and $\mu = 0.5$)	Case 2: ($\lambda = 0.35$ and $\mu = 0.6$)
$z_1 = \text{OR}(x_2, x_3, \overline{x}_3)$	$z_1 = \text{OR}(x_1, x_3)$
$z_2 = x_2$	$z_2 = \text{OR}(x_2, x_3)$
$z_3 = x_4$	
$\hat{x}_1 = \text{AND}(z_2,\ z_3)$	$\hat{x}_1 = \text{AND}(z_1,\ z_2)$
$\hat{x}_2 = z_1$	$\hat{x}_2 = z_2$
$\hat{x}_3 = z_1$	$\hat{x}_3 = z_1$
$\hat{x}_4 = \text{AND}(z_1,\ z_2)$	
Case 3: ($\lambda = 0.65$ and μ=.7)	Case 3: ($\lambda = 0.45$ and $\mu = .6$)
$z_1 = \text{OR}(x_2, \overline{x_2})$	$z_1 = \text{OR}(x_1, \overline{x_2}, \overline{x}_3)$
$z_2 = \text{OR}(x_1, x_3, \overline{x_2}, \overline{x_4})$	$z_2 = x_2$
$z_3 =$ $\text{OR}(x_2, x_3, x_4, \overline{x_2}, \overline{x_2}, \overline{x_4})$	
$\hat{x}_1 = z_3$	
$\hat{x}_2 = \text{AND}(z_1, z_3)$	$\hat{x}_1 = z_1$
$\hat{x}_3 = \text{AND}(z_2, z_3)$	$\hat{x}_2 = z_2$
$\hat{x}_4 = \text{AND}(z_1, z_3)$	$\hat{x}_3 = \text{AND}(z_1, z_2)$

where overbar denotes a complement operation.

It should be emphasized that the main objective of this paper is the realization of the transparent representation of autoencoder and its weights. As shown in Fig. 9, the autoencoder can be represented with the use of a lower number of variables when selecting some threshold values of λ and μ. Although the performance index becomes higher compared to the performance index produced by the "traditional" autoencoder, the differences are not substantial while the benefits stemming from the emerging logic and transparent architecture are tangible.

To investigate an impact of h on the performance of the constructed reduced space, we consider the MNIST database of handwritten digits (http://yann.lecun.com/exdb/mnist/), we have resized the images to become of size 15 by 15 pixels. The dimensionality of the reduced feature space was selected as $h = 10$, 20, 30, 40, and 50. For these values, the resulting values of Q for the training and testing data are displayed in Fig. 10. The noticeable tendency is visible: as anticipated, the lower the values of h becomes, the higher the values of Q are obtained. The best results were reported for the pair of t-norms and t-conorms specified by case 1.

6. Conclusions

The study has been devoted to the logic-based autoencoders. In comparison with the existing architectures, the proposed construct comes with inherent transparency of the dependencies involved in the building of the reduced space. The logic character of processing helps achieve insight into the way in which original features are aggregated and contribute to the buildup of more abstract aggregates. The design processes along with the analysis and interpretation of the developed structure have been provided and illustrated through a series of experimental studies. The proposal put forward in this study offers a direction of logic-oriented and transparent architectures of building a compact and

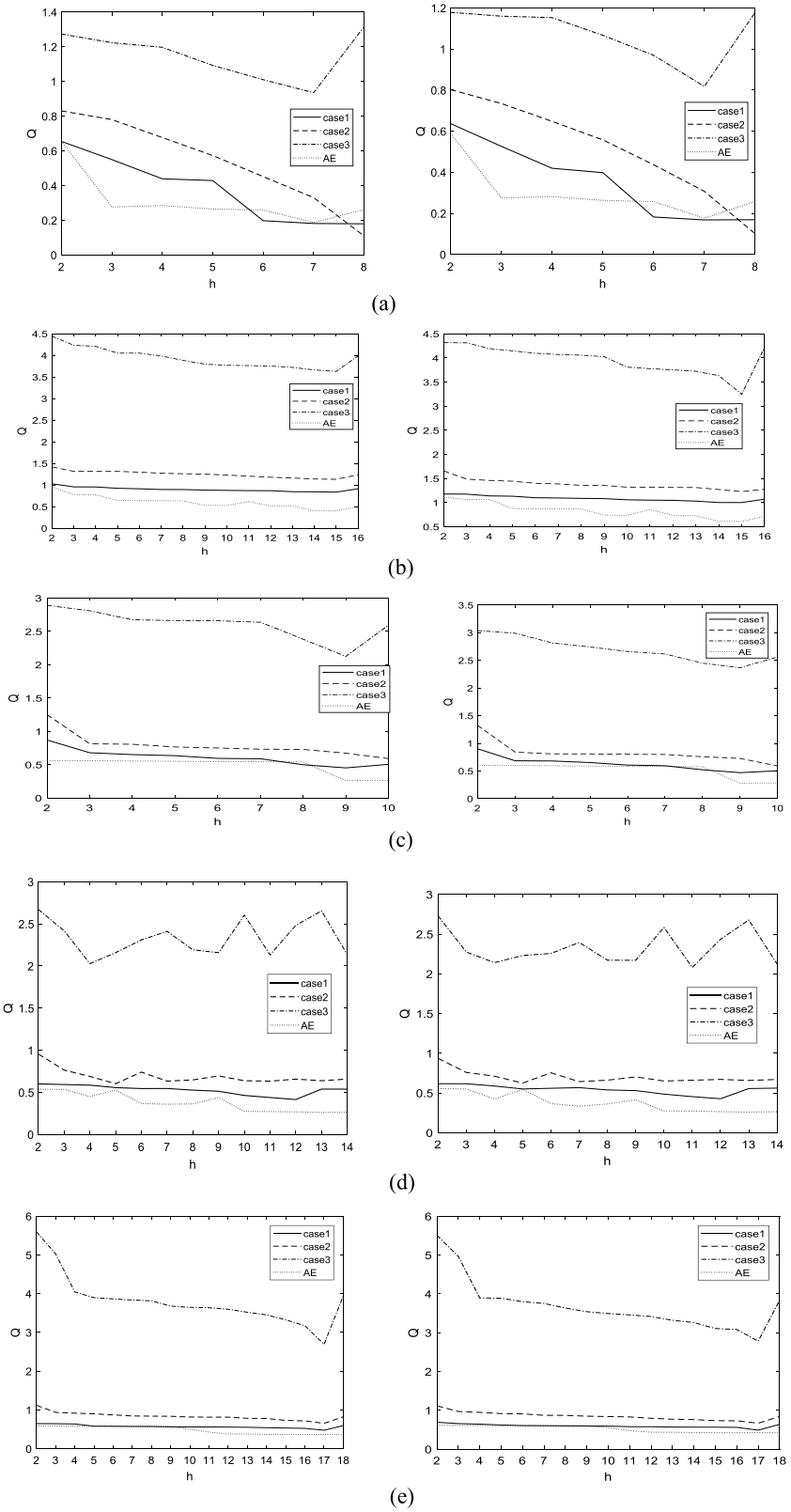

Fig. 7. Performance index Q of training and testing (left to right) for different values of h for (a) iris (b) carsmall (c) wine (d) leaf (e) ecoli.

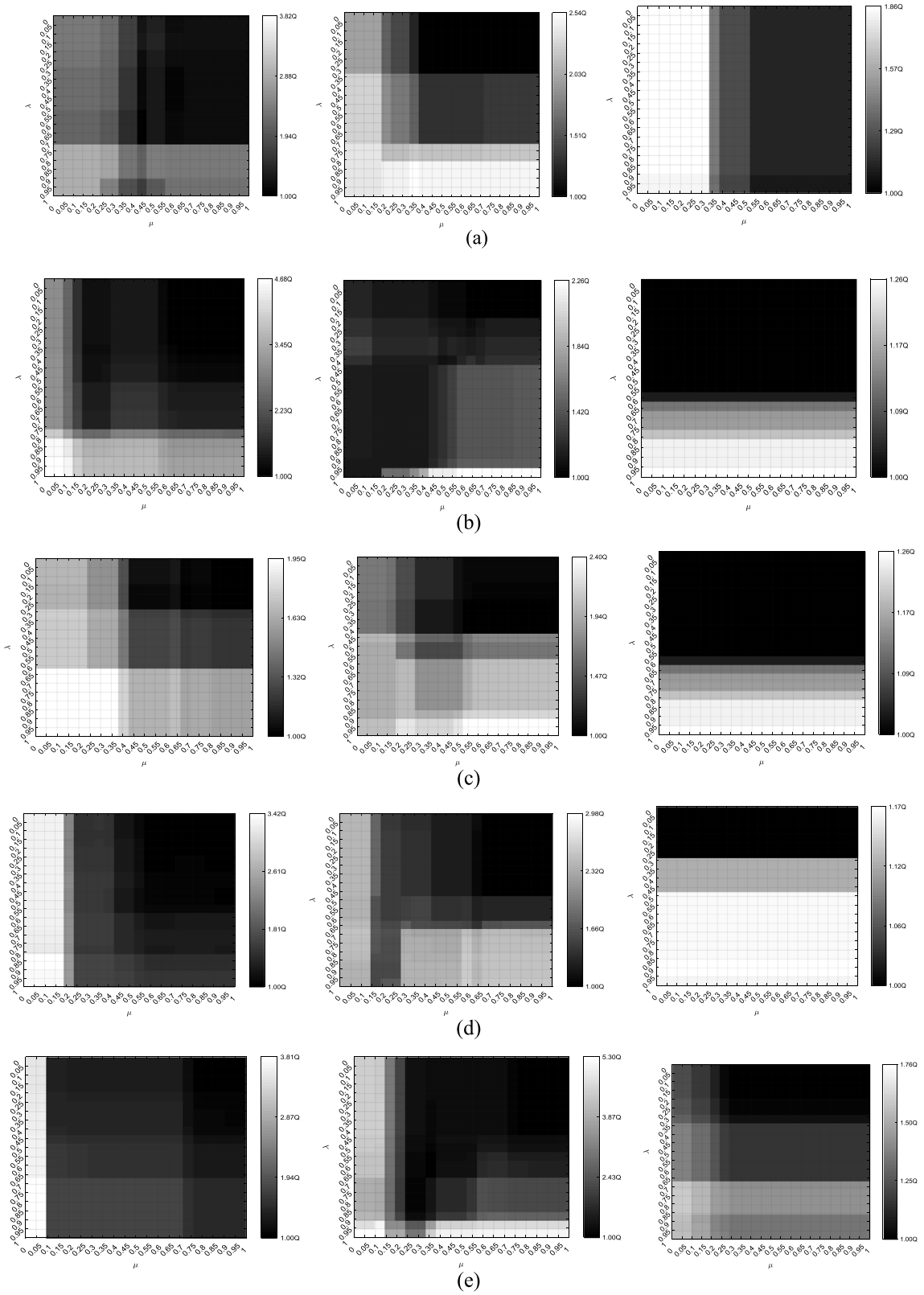

Fig. 8. From left to right case1, case 2, and case 3 contour plot of Q for (a) iris ($h = 3$) (b) carsmall ($h = 5$) (c) wine ($h = 3$) (d) leaf ($h = 4$) (e) ecoli ($h = 5$).

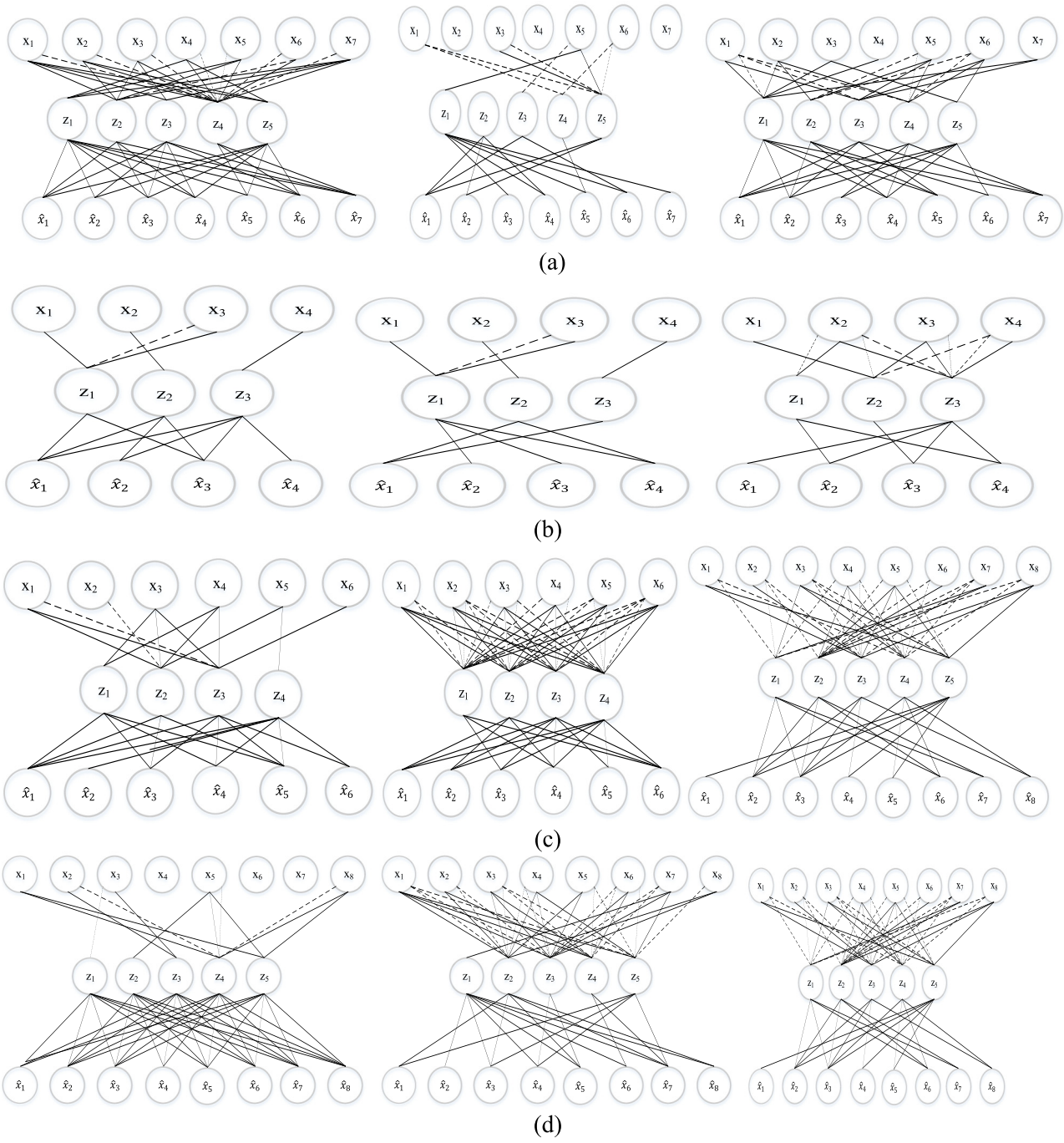

Fig. 9. From left to right case 1, case 2, and case 3 logic structure of autoencoder for $Q^*/Q = 1.1$ for (a) carsmall ($h = 5$) (b) wine ($h = 3$) (c) leaf ($h = 4$) (d) ecoli ($h = 5$).

abstract view of data through an ensemble of logic constructs of features.

There are several future interesting and practically relevant studies along the line of investigation elaborated here. More learning scenarios can be involved in the formation of the learning environment; especially one could focus on multilayer (cascaded) structure with a particular attention paid to the associated interpretation of results, a selection of triangular norms (which comes as an item on the agenda of structural design). To cope with a lack of ideal compression processes and quantify the quality of the constructed logic autoencoders, bringing mechanisms of Granular Computing could constitute another direction of studies worth investigating.

Acknowledgments

This project was funded by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, Saudi Arabia under grant no. (KEP-5-135-39). The authors, therefore, acknowledge with thanks DSR technical and financial support.

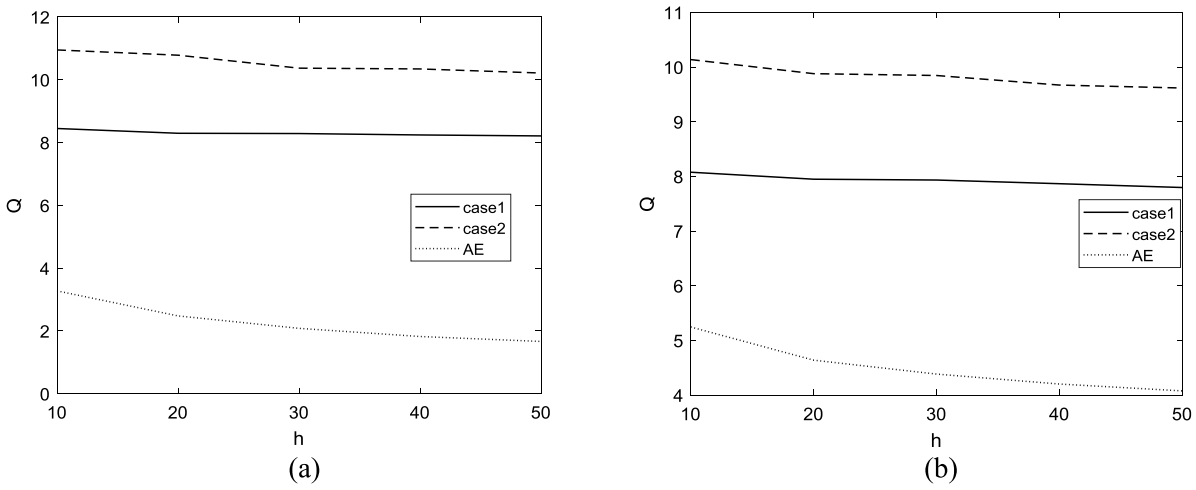

Fig. 10. Performance index Q as a function of h; shown are the results for (a) the training and (b) testing data.

References

[1] R. Salakhutdinov, G. Hinton, An efficient learning procedure for deep Boltzmann machines, Neural Comput. 24 (2012) 1967–2006.

[2] J. Deng, Z. Zhang, F. Eyben, B. Schuller, Autoencoder-based unsupervised domain adaptation for speech emotion recognition, IEEE Signal Process. Lett. 21 (2014) 1068–1072.

[3] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.A. Manzagol, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion, J. Mach. Learn. 11 (12) (2010) 3371–3408.

[4] S. Rifai, P. Vincent, X. Müller, X. Glorot, Y. Bengio, Contractive auto-encoders: Explicit invariance during feature extraction, in: Proc. 28th Int. Conf. Int. Conf. Mach. Learn. (ICML), 2011, pp. 833–840.

[5] C. Wu, F. Wu, S. Wu, Z. Yuan, J. Liu, Y. Huang, Semi-supervised dimensional sentiment analysis with variational autoencoder, Knowl.-Based Syst. 165 (2019) 30–39.

[6] M. Ranzato, C. Poultney, S. Chopra, Y.L. Cun, Efficient learning of sparse representations with an energy-based model, Adv. Neural Inf. Process. Syst. 19 (2007) 1137–1144.

[7] S. Gao, Y. Zhang, K. Jia, J. Lu, Y. Zhang, Single sample face recognition via learning deep supervised autoencoders, IEEE Trans. Inf. Forensics Secur. 10 (2015) 2108–2118.

[8] C. Xia, F. Qi, G. Shi, Bottom–up visual saliency estimation with deep autoencoder-based sparse reconstruction, IEEE Trans. Neural Netw. Learn. Syst. 27 (2016) 1227–1240.

[9] K.G. Dizaji, A. Herandi, C. Deng, W. Cai, H. Huang, Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization, in: Proc. I EEE Int. Conf. Comput. Vis. (ICCV), 2017, pp. 5747–5756.

[10] J. Deng, Z. Zhang, E. Marchi, B. Schuller, Sparse autoencoder based feature transfer learning for speech emotion recognition, in: Proc. Hum. Assoc. Conf. Affect. Comput. Intell. Interact. (ACII), 2013, pp. 511–516.

[11] Z. Chen, C.K. Yeo, B.S. Lee, C.T. Lau, Y. Jin, Evolutionary multi-objective optimization based ensemble autoencoders for image outlier detection, Neurocomputing 309 (2018) 192–200.

[12] A. Janowczyk, A. Basavanhally, A. Madabhushi, Stain normalization using sparse autoencoders (StaNoSA): Application to digital pathology, Comput. Med. Imaging Graph. 57 (2017) 50–61.

[13] Đ.T. Grozdić, S.T. Jovičić, M. Subotić, Whispered speech recognition using deep denoising autoencoder, Eng. Appl. Artif. Intell. 59 (2017) 15–22.

[14] C. Hong, J. Yu, J. Wan, D. Tao, M. Wang, Multimodal deep autoencoder for human pose recovery, IEEE Trans. Image Process. 24 (12) (2015) 5659–5670.

[15] J.Y. Kim, S.B. Cho, Electric energy consumption prediction by deep learning with state explainable autoencoder, Energies 12 (4) (2019) 739.

[16] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Netw. 61 (2015) 85–117.

[17] Q. Guo, J. Jia, G. Shen, L. Zhang, L. Cai, Z. Yi, Learning robust uniform features for cross-media social data by using cross autoencoders, Knowl. Based Syst. 102 (2016) 64–75.

[18] J. Leng, Q. Chen, N. Mao, P. Jiang, Combining granular computing technique with deep learning for service planning under social manufacturing contexts, Knowl.-Based Syst. 143 (2018) 295–306.

[19] V. Bellini, A. Schiavone, T. Di Noia, A. Ragone, E.D. Sciascio, Knowledge-aware autoencoders for explainable recommender systems, in: Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems, 2018, pp. 24–31.

[20] J. Calvo-Zaragoza, A. Gallego, A selectional auto-encoder approach for document image binarization, Pattern Recognit. 86 (2019) 37–47.

[21] J. Deng, Z. Zhang, F. Eyben, B. Schuller, Autoencoder-based unsupervised domain adaptation for speech emotion recognition, IEEE Signal Process. Lett. 21 (9) (2014) 1068–1072.

[22] J. Zabalza, J. Ren, J. Zheng, H. Zhao, C. Qing, Z. Yang, P. Du, S. Marshall, Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging, Neurocomputing 185 (2016) 1–10.

[23] G. Hinton, R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.

[24] L.A. Zadeh, Towards a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic, Fuzzy Sets and Systems 90 (2) (1997) 111–117.

[25] W. Pedrycz, Granular Computing: Analysis and Design of Intelligent Systems, CRC Press/Francis Taylor, Boca Raton, 2013.

[26] K. Hirota, W. Pedrycz, OR/AND neuron in modeling fuzzy connectives, IEEE Trans. Fuzzy Syst. 2 (2) (1994) 15l–161.

[27] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, 1981.