Postprint: Classification Algorithm for Sparse and Label-Constrained Semi-Supervised Autoencoder

Authors: Wang Huiling, Song Wei, Wang Chenni

Date: 2018-05-24T00:00:00+00:00

Abstract

Autoencoders can express semantic features of data through deep unsupervised learning, but due to the difficulty in effectively determining the number of hidden layer nodes, the processed data often leads to low classification accuracy and weak stability when further used for classification. To address these issues, a Semi-supervised Autoencoder with Sparsity and Label Constraints (SLRAE) is proposed to achieve an organic combination of unsupervised and supervised learning, thereby more accurately extracting the essential features of samples. The sparsity constraint term imposes constraints on the response of each hidden node, enabling the discovery of potential structures in data even when the number of hidden neurons is large; simultaneously, a label constraint term is introduced to compare actual labels with expected labels in a supervised learning manner, adjusting network parameters in a targeted fashion to further improve classification accuracy. To verify the effectiveness of the proposed method, extensive experiments were conducted on multiple datasets. The results demonstrate that, compared with traditional Autoencoders (AE), Sparse Autoencoders (SAE), and Extreme Learning Machines (ELM), the data processed by SLRAE and applied to the same classifier can significantly improve classification accuracy and stability.

Full Text

Preamble

Sparse and Label Regularized Semi-supervised Auto-encoder for Classification

Wang Huiling[1], Song Wei[1,2], Wang Chenni[1,2]

1. School of Internet of Things Engineering, Jiangnan University, Wuxi, Jiangsu 214122, China

2. Engineering Research Center of Internet of Things Technology Applications, Ministry of Education, Wuxi, Jiangsu 214122, China

Abstract: Auto-encoders can express semantic features of data through deep unsupervised learning, but determining the optimal number of hidden layer nodes is challenging. When the processed data is subsequently used for classification, this often leads to low classification accuracy and weak stability. To address these issues, this paper proposes a Sparse and Label Regularized Auto-Encoder (SLRAE) that achieves organic integration of unsupervised and supervised learning to more accurately extract essential features from samples. The sparse regularization term imposes constraints on the response of each hidden node, enabling the algorithm to discover potential structures in data even when the number of hidden neurons is large. Simultaneously, a label regularization term is introduced to compare actual labels with desired labels in a supervised manner, adjusting network parameters specifically to further improve classification accuracy. To verify the effectiveness of the proposed method, extensive experiments were conducted on multiple datasets. The results demonstrate that compared with traditional auto-encoders (AE), sparse auto-encoders (SAE), and extreme learning machines (ELM), SLRAE significantly improves classification accuracy and stability when the processed data is applied to the same classifier.

Keywords: classification; sparse regularization; label regularization; auto-encoder; extreme learning machine

0 Introduction

With the rapid development of information technology, data in digital databases is growing exponentially. Effectively extracting valuable features from massive information has become a key research focus in data mining and pattern recognition. Since 2006, auto-encoders (AE) [2], represented by feature learning [1], have achieved breakthrough progress in representation learning within machine learning. These advances are primarily used for initializing deep learning models [3]. However, conventional AE often suffers from decreased classification accuracy due to their complex network structures.

To address this problem, scholars have proposed several AE regularization methods: Sparse Auto-Encoder (SAE) [4,5] was introduced by Bengio et al. in 2007, requiring hidden layer neuron activations to satisfy certain sparsity constraints; Denoising Auto-Encoder (DAE) [6,7] was proposed by Vincent et al. in 2008, which adds noise to the input vector and trains the encoder to reconstruct the original input, making the network more robust; Contractive Auto-Encoder (CAE) [8,9] was proposed by Bengio et al. in 2011, which uses the squared Frobenius norm of the Jacobian matrix of the hidden layer output with respect to the

weights as a regularization term in the reconstruction function to obtain robust intermediate features.

The above regularized AEs are all unsupervised learning processes [10]. Unsupervised deep learning methods analyze data itself and can effectively extract deep abstract features, but due to the lack of prior label information, the extracted features cannot describe specific categories and are difficult to apply to classification [11]. Supervised learning can solve this problem well, but the learned features cannot represent the original data effectively, have weak generalization ability, and easily produce overfitting.

To better address the problems existing in unsupervised and supervised learning methods, this paper proposes a Sparse and Label Regularized Semi-supervised Auto-Encoder (SLRAE). This approach uses basic semantic features obtained from unsupervised learning and discriminative features describing labels from supervised learning to represent data characteristics, which are then further used for classification. To verify the effectiveness of our proposed method, we conducted extensive experiments on the USPS database [12] and six UCI datasets [13]. We compared the proposed SLRAE method with AE, SAE, ELM, LRAE (auto-encoder using only label regularization), DBN [14], and Adaptive DBN [15]. Experimental results demonstrate that SLRAE achieves better classification accuracy.

1.1 Auto-Encoder (AE)

The concept of auto-encoder was proposed in the late 1980s. A basic auto-encoder consists of a three-layer neural network including an input layer, a hidden layer, and an output layer, where the number of neurons in the input and output layers is identical. In the encoding stage, a function f_θ maps the input vector $\mathbf{x}$ to obtain the intermediate representation $\mathbf{y}$. In the decoding stage, a function g_θ maps the intermediate representation $\mathbf{y}$ to obtain the reconstructed data $\mathbf{z}$. The auto-encoder fine-tunes network parameters by minimizing the reconstruction error J_{AE}:

$$J_{AE} = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{z}_i\|^2$$

where $\mathbf{x}_i$ is the i-th training sample, $\mathbf{z}_i$ is the i-th output data, and n is the number of training samples.

In AE, the parameters to be adjusted are $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{W}', \mathbf{b}'\}$, where $\mathbf{b}$ and $\mathbf{b}'$ are encoding and decoding biases, $\mathbf{W}$ and $\mathbf{W}'$ are encoding and decoding weights, and $\mathbf{W}'$ is the transpose of $\mathbf{W}$. Typically, the classical gradient descent method is used to obtain optimal values for these parameters.

Compared with traditional BP neural networks [16] that require substantial manual feature extraction work, AE can improve the efficiency of feature extraction, reduce the dimensionality of original input data, and effectively learn compressed and distributed characteristics of given datasets.

1.2 Extreme Learning Machine (ELM)

Extreme Learning Machine (ELM) [17,18] is a single-hidden-layer feedforward neural network with fast learning characteristics proposed by Professor Huang Guangbin. The input weights and biases of the hidden layer are randomly generated; only the number of hidden layer neurons needs to be set to calculate the output weights of the network. A traditional ELM includes an input layer, a hidden layer, and an output layer. Assuming there are N arbitrary samples $\{(\mathbf{x}_j, \mathbf{t}_j)\}_{j=1}^{N}$, the hidden layer output matrix of this ELM network structure with input $\mathbf{x}_j$ is $\mathbf{H}$, then the entire network output layer can be expressed as:

$$\mathbf{T} = \mathbf{H}\beta$$

where β is the weight matrix between the hidden layer and output layer, and $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_N]_{m\times N}^{T}$ represents the target label values of samples. Thus, the training of the feedforward neural network can be transformed into solving a least squares problem for the output weight matrix. The output weights can be obtained by:

$$\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose (MP) generalized inverse of the hidden layer output matrix $\mathbf{H}$. Unlike traditional feedforward neural networks that require multiple iterative feedbacks, ELM uses the least squares method to directly calculate the weight matrix, resulting in fast training speed and strong generalization ability.

---

2 Sparse and Label Regularized Semi-supervised Auto-Encoder (SLRAE)

This paper proposes a semi-supervised auto-encoder using sparse and label regularizations, where the unsupervised learning process for unlabeled samples can improve the generalization ability of the semi-supervised auto-encoder, and the supervised learning process for labeled samples can enhance the classification accuracy of the semi-supervised auto-encoder model.

In the auto-encoder, a label term is added to reduce the error between actual output labels and desired output labels of training data, thereby compensating for the low classification accuracy of unsupervised learning. We also utilize sparse regularization to achieve sparse coding of input data, effectively extracting hidden structures of data by constraining the number of hidden layer nodes, which provides good capability for learning datasets. The learned feature representations are then input to a classifier, and the classification results verify the effectiveness of the proposed semi-supervised auto-encoder.

[Figure 1: see original paper] shows the classification framework of SLRAE. The cost function of SLRAE consists of J_{AE}, J_{wd}, J_{sparse}, and J_{label}. J_{AE} is the reconstruction error between input and output data. Minimizing J_{AE} makes the input data as close as possible to the output data, thereby more accurately reconstructing the output. J_{wd} is used to reduce the magnitude of weights and prevent overfitting. Based on this, our proposed SLRAE also includes a sparse constraint term J_{sparse} and a label error term J_{label} to constrain the hidden layer, making the representation sparser and enabling SLRAE to effectively extract potential essential features from large amounts of data.

In fact, our label constraint utilizes supervised learning to improve classification accuracy. This label term is calculated using the idea of Extreme Learning Machine (ELM). Since original input data is often large and redundant, making it difficult to directly obtain essential semantic features, the sparse constraint limits the number of hidden layer nodes to make the representation sparser. Adding sparse constraints can capture the main information of the input. Therefore, the SLRAE proposed in this paper can not only effectively extract potential essential features from large amounts of data but also improve classification accuracy.

In SLRAE, assuming n samples $\{\mathbf{x}_i\}_{i=1}^n$ are input to a d-dimensional space, each sample is represented as $\mathbf{x}_i \in \mathbb{R}^d$. The encoding function f_{θ_i} maps $\mathbf{x}_i$ to an l-dimensional hidden layer to obtain the intermediate representation $\mathbf{y}_i$, and the decoding function g_{θ_i} reconstructs $\mathbf{y}_i$ to obtain $\mathbf{z}_i$. They can be concisely expressed as:

$$\mathbf{y}_i = f_{\theta_i}(\mathbf{x}_i) = s(\mathbf{W}_1\mathbf{x}_i + \mathbf{b}_1)$$

$$\mathbf{z}_i = g_{\theta_i}(\mathbf{y}_i) = s(\mathbf{W}_2\mathbf{y}_i + \mathbf{b}_2)$$

where $s(x) = \frac{1}{1+e^{-x}}$ is the sigmoid activation function. $\mathbf{W}_1$ and $\mathbf{b}_1$ represent the weights and biases between the input layer and hidden layer, while $\mathbf{W}_2$ and $\mathbf{b}_2$ represent the weights and biases between the hidden layer and output layer. We use $\mathbf{X} \in \mathbb{R}^{d\times n}$ to denote the sample feature matrix of the input layer, $\mathbf{Y} \in \mathbb{R}^{l\times n}$ to denote the sample feature matrix of the hidden layer, and $\mathbf{Z} \in \mathbb{R}^{d\times n}$ to denote the sample feature matrix of the output layer for the input data.

In this paper, the number of hidden layer nodes is much larger than that of the input layer. We can add a sparsity constraint on the hidden layer to ensure learning more local structural information. In each iteration, SLRAE calculates the average activation degree $\widehat{\rho}_j = \frac{1}{n}\sum_{i=1}^{n} a_j(\mathbf{x}_i)$, where n is the number of samples, $\mathbf{x}_i$ is the i-th sample, and $a_j(\mathbf{x}_i)$ represents the activation degree of the j-th hidden neuron for the i-th sample. Under the sparsity constraint, ρ is the sparsity parameter with $\widehat{\rho}_j = \rho$, and typically ρ is a small value close to 0. Here, KL divergence is a method used to measure the difference between two distributions, calculated as:

$$J_{sparse} = \sum_{j=1}^{l} KL(\rho \| \widehat{\rho}_j) = \sum_{j=1}^{l} \left[\rho \log \frac{\rho}{\widehat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\widehat{\rho}_j} \right]$$

In this paper, we also use data labels. Let $\mathbf{t}_i$ represent the original label of the i-th input sample, $\mathbf{H}$ represent the output matrix of the hidden layer, and β represent the weights and biases between the hidden layer and output label layer. By using the formula $\hat{\mathbf{T}} = \mathbf{H}\beta$, the actual output label can be calculated. Based on the given labels $\mathbf{T}$ and the actual output labels $\hat{\mathbf{T}}$ calculated by ELM, the label error between actual and given labels is obtained and minimized, which can be expressed as:

$$J_{label} = \frac{1}{2n} \sum_{i=1}^{n} \|\mathbf{t}_i - \hat{\mathbf{t}}_i\|^2 = \frac{1}{2n} \sum_{i=1}^{n} \|\mathbf{t}_i - \mathbf{H}_i\beta\|^2$$

Thus, the objective function of SLRAE is:

$$J_{SLRAE} = J_{AE} + \lambda J_{wd} + \beta J_{sparse} + \gamma J_{label}$$

where λ, β, and γ are parameters controlling the relative importance of each term in the formula. The first term is the reconstruction error term, the second term J_{wd} is a weight decay term to prevent overfitting, the third term is the sparsity penalty term that allows discovering important structures in input data even when the number of hidden neurons is large, and the last term is the label error term representing the error between original and actual labels.

The gradient descent method is used to optimize the objective function and update $\mathbf{W}_1$, $\mathbf{b}_1$, $\mathbf{W}_2$, and $\mathbf{b}_2$ as follows:

$$\mathbf{W}_1 \leftarrow \mathbf{W}_1 - \alpha \frac{\partial J_{SLRAE}}{\partial \mathbf{W}_1}$$

$$\mathbf{b}_1 \leftarrow \mathbf{b}_1 - \alpha \frac{\partial J_{SLRAE}}{\partial \mathbf{b}_1}$$

where α is the learning rate.

For the reconstruction error term J_{AE}, the partial derivatives with respect to $\mathbf{Z}$ can be expressed as:

$$\frac{\partial J_{AE}}{\partial \mathbf{Z}} = \frac{1}{n}(\mathbf{Z} - \mathbf{X})$$

Since the sparsity constraint term only acts on the hidden layer, when using the backpropagation method, feedback only needs to be applied to the encoding stage. Therefore, the partial derivatives of J_{sparse} with respect to $\mathbf{W}_1$ and $\mathbf{b}_1$ are:

$$\frac{\partial J_{sparse}}{\partial \mathbf{W}_1} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial J_{sparse}}{\partial \mathbf{a}_i} \frac{\partial \mathbf{a}_i}{\partial \mathbf{W}_1}$$

$$\frac{\partial J_{sparse}}{\partial \mathbf{b}_1} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial J_{sparse}}{\partial \mathbf{a}_i} \frac{\partial \mathbf{a}_i}{\partial \mathbf{b}_1}$$

For the label error term J_{label}, the partial derivative with respect to the hidden layer representation $\mathbf{Y}$ can be expressed as:

$$\frac{\partial J_{label}}{\partial \mathbf{Y}} = \frac{\partial J_{label}}{\partial \hat{\mathbf{T}}} \frac{\partial \hat{\mathbf{T}}}{\partial \mathbf{Y}} = \frac{1}{n}(\hat{\mathbf{T}} - \mathbf{T})\beta^T$$

Similarly, the label error term only acts between the input and hidden layers, generating error only between them. Therefore, the partial derivatives of J_{label} with respect to $\mathbf{W}_1$ and $\mathbf{b}_1$ are:

$$\frac{\partial J_{label}}{\partial \mathbf{W}_1} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial J_{label}}{\partial \mathbf{y}_i} \frac{\partial \mathbf{y}_i}{\partial \mathbf{W}_1}$$

$$\frac{\partial J_{label}}{\partial \mathbf{b}_1} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial J_{label}}{\partial \mathbf{y}_i} \frac{\partial \mathbf{y}_i}{\partial \mathbf{b}_1}$$

Combining the partial derivatives of the reconstruction error, weight decay term, sparsity constraint term, and label error term, the partial derivatives of the SLRAE objective function can be obtained as:

$$\frac{\partial J_{SLRAE}}{\partial \mathbf{W}_1} = \frac{\partial J_{AE}}{\partial \mathbf{W}_1} + \lambda \frac{\partial J_{wd}}{\partial \mathbf{W}_1} + \beta \frac{\partial J_{sparse}}{\partial \mathbf{W}_1} + \gamma \frac{\partial J_{label}}{\partial \mathbf{W}_1}$$

$$\frac{\partial J_{SLRAE}}{\partial \mathbf{b}_1} = \frac{\partial J_{AE}}{\partial \mathbf{b}_1} + \lambda \frac{\partial J_{wd}}{\partial \mathbf{b}_1} + \beta \frac{\partial J_{sparse}}{\partial \mathbf{b}_1} + \gamma \frac{\partial J_{label}}{\partial \mathbf{b}_1}$$

3 Experimental Results and Analysis

To test the algorithm' s performance, we compared the proposed SLRAE with AE, SAE, and ELM on several datasets, then applied them to a softmax classifier for classification. The datasets mainly include six UCI datasets: Pen Digits, Iris, Glass, Seeds, ISOLET, Page Blocks, and one USPS dataset. Detailed dataset information is shown in . The algorithms were implemented in MATLAB R2014a on a Windows 7 operating system with an Intel(R) Core(TM) i3-3.40 GHz CPU and 4 GB of memory.

The SLRAE algorithm has parameters λ, β, and γ that control the relative importance of each term in the objective function. We discuss these parameters by varying them according to the rule $\{10^e|e = -5, -4, -3, -2, -1, 0, 1, 2, 3, 4\}$. [Figure 2: see original paper] discusses the performance of SLRAE with different parameters λ, β, and γ using the Pen Digits database. Each figure shows a different λ value, the x and y axes represent the variation range of parameters β and γ, and the color bar represents accuracy. From [Figure 2: see original paper], it can be seen that when $\lambda = 0.0001$, $\beta = 1000$, and $\gamma = 0.1$, SLRAE achieves optimal performance with classification accuracy reaching 99.4%.

To ensure fair comparison, the compared networks have the same architecture (i.e., the same depth and number of hidden layer nodes). Using the parameters that yielded the highest classification results in the above experiments, we then investigated the impact of the number of neurons on performance. We first set optimal model parameters for each dataset, then varied the number of hidden layer nodes while fixing other parameters. [Figure 3: see original paper] shows the classification results on Pen Digits, USPS, and ISOLET datasets. The number of hidden layer neurons was set according to the original feature dimensions of each dataset. By changing the number of hidden layer neurons, SLRAE consistently outperforms other algorithms in classification accuracy. As shown in Figure 3: see original paper, when the number of hidden layer nodes reaches 100, the classification accuracy of SLRAE reaches a stable state, with accuracy approaching 100%. From Figure 3: see original paper, it can be seen that the classification accuracy of SLRAE remains around 93%, with consistently stable results. The classification accuracy of SLRAE is significantly higher than other algorithms. Figure 3: see original paper shows that SLRAE's classification accuracy is clearly superior to other algorithms, remaining around 95.5% regardless of the number of hidden layer nodes.

In subsequent experiments, we investigated the impact of different iteration numbers on the classification accuracy of SLRAE, SAE, AE, ELM, LRAE, DBN, and Adaptive DBN. We obtained the optimal number of hidden layer nodes for each method on each dataset in the previous experiment. [Figure 4: see original paper] shows the impact of iteration numbers on each method on Pen Digits, USPS, and ISOLET datasets.

As shown in [Figure 4: see original paper], SLRAE' s classification accuracy is significantly higher than SAE, AE, ELM, LRAE, DBN, and Adaptive DBN

across all iteration numbers. From Figure 4: see original paper, the impact of iteration numbers on SLRAE' s classification accuracy is minimal, with accuracy as high as 99.41%. From Figure 4: see original paper, when the iteration number reaches 400, SLRAE' s classification accuracy reaches a stable state at 94.4%. From Figure 4: see original paper, SLRAE' s classification accuracy is smooth and does not change dramatically with increasing iteration numbers, reaching a maximum of 95.7% when the iteration number is 800.

shows the accuracy ranges over 100 runs on each database. The upper row of each dataset represents the average accuracy and fluctuation to maximum accuracy, while the lower row represents the fluctuation to minimum accuracy. It can be seen that SLRAE exhibits small fluctuations in classification accuracy across datasets. Although sometimes the fluctuation is larger than DBN (which shows zero fluctuation on USPS, Iris, Glass, Seeds, and Page Blocks datasets), SLRAE' s average classification accuracy is higher than DBN and other algorithms, demonstrating SLRAE' s clear advantage for classification.

Using only the sparsity constraint term without target information is not conducive to classification. Using only the label term leads to learned features that cannot well represent the original data, resulting in weak generalization ability and easy overfitting. To address this issue and leverage the advantages of both unsupervised and supervised learning, we propose a semi-supervised auto-encoder that adds both sparsity and label constraints. By simultaneously optimizing auto-encoder parameters and classification parameters, the method can preserve original data features, ensure model generalization capability, and achieve good classification performance.

In this paper, we incorporate sparse and label regularization into the auto-encoder to propose a semi-supervised auto-encoder (SLRAE). The added sparsity constraint term can effectively extract hidden layer structures, learn more complex nonlinear functions, and more accurately extract essential features of samples. The added label constraint term improves classification accuracy above supervised and unsupervised models by reducing the error between actual and desired labels.

References

[1] Hinton G E, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets [J]. Neural Computation, 2014, 18(7): 1527-1554.

[2] Bengio Y. Learning deep architectures for AI [M]. [S.l.]: Now Publishers, 2009.

[3] Li Deng, Dong Yu. Deep learning: methods and applications [J]. Foundations & Trends in Signal Processing, 2014, 7(3): 197-387.

[4] Bengio Y, Lamblin P, Popovici D, et al. Greedy layer-wise training of deep

networks [C]// Advances in Neural Information Processing Systems. 2007: 153-160.

[5] Sun Wenjun, Shao Siyu, Zhao Rui, et al. A sparse auto-encoder-based deep neural network approach for induction motor faults classification [J]. Measurement, 2016, 89: 171-178.

[6] Vincent P, Larochelle H, Bengio Y, et al. Extracting and composing robust features with denoising autoencoders [C]// Proc of International Conference on Machine Learning. New York: ACM Press, 2008: 1096-1103.

[7] Vincent P, Larochelle H, Lajoie I, et al. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion [J]. Journal of Machine Learning Research, 2010, 11(12): 3371-3408.

[8] Rifai S, Mesnil G, Vincent P, et al. Higher order contractive auto-encoder [M]// Machine Learning and Knowledge Discovery in Databases. Berlin: Springer, 2011: 645-660.

[9] Liu Yanan, Feng Xiaoqing, Zhou Zhiguang. Multimodal video classification with stacked contractive autoencoders [J]. Signal Processing, 2016, 120: 721-733.

[10] Du Bo, Wei Xiong, Jia Wu, et al. Stacked Convolutional Denoising Auto-Encoders for Feature Representation [J]. IEEE Trans on Cybernetics, 2016, 99: 1-11.

[11] Liu Weifeng, Ma Tengzhou, Xie Qiangsheng, et al. LMAE: a large margin auto-encoders for classification [J]. Signal Processing, 2017, 141: 137-143.

[12] http://www.datatang.com/data/11927 [EB/OL].

[13] Blake C, Merz C. Uci repository of machine learning databases [DB/OL].

[14] Liu Ping, Han Shizhong, Meng Zibo, et al. Facial expression recognition via a boosted deep belief network [C]// Proc of IEEE Conference on Computer Vision and Pattern Recognition. Washington DC: IEEE Computer Society, 2014: 1805-1812.

[15] Kamada S, Ichimura T. An adaptive learning method of deep belief network by layer generation algorithm [C]// Proc of Region 10 Conference. 2017: 1201-1206.

[16] Zou Qiang, Fang Hui, Liu Fei, et al. Comparative study of distance discriminant analysis and BP neural network for identification of rapeseed cultivars using visible/near infrared spectra [C]// IFIP Advances in Information & Communication Technology, 2017, 347: 124-133.

[17] Huang GuangBin, Zhu QinYu, Siew C K. Extreme learning machine: Theory and applications [J]. Neurocomputing, 2006, 70(1–3): 489-501.

[18] Huang Guangbin, Wang Dianhui, Lan Yuan. Extreme learning machines: a survey [J]. International Journal of Machine Learning & Cybernetics, 2011, 2(2): 107-122.

[19] Lin Yu. Research on fusion algorithm of extreme learning machine and auto-encoder [D]. Changchun: Jilin University, 2016.

Note: Figure translations are in progress. See original paper for figures.