Proceedings of the 39th Chinese Control Conference
July 27-29, 2020, Shenyang, China

A Composite Cost-Sensitive Neural Network for Imbalanced Classification

Lei Chen[1,2], Yuan Zhu[1,2]*
1. School of Automation, China University of Geosciences, Wuhan, 430074, P.R. China
2. Hubei Key Laboratory of Advanced Control and Intelligent Automation for Complex Systems, Wuhan, 430074, China
E-mail: zhuyuan@cug.edu.cn

Abstract: Imbalanced data with skewed class distributions and different misclassification costs is common in many real-world applications. Traditional classification approach does not work well for imbalanced data, because they assume equal costs for each class. To deal with this problem, cost-sensitive approaches assign different misclassification costs for different classes without disrupting the true original distributions of samples. However, due to lack of prior knowledge, the misclassification costs are usually unknown and hard to choose in practice. Whats more, even instances in the same class may have different misclassification costs. As an extension of class-dependent costs, this paper presents a composite cost-sensitive deep neural network (CCS-DNN) for imbalanced classification. A specifically-designed cost-sensitive matrix, which is composed of example-dependent costs and class-dependent costs, is embedded into the loss function to improve the classification performance. And the parameters of both the cost-sensitive matrix and the network are jointly optimized during training. The results of comparative experiments on some benchmark datasets indicate that the CCS-DNN performs better than other baseline methods.

Key Words: Cost-sensitive Learning, Composite Costs, Imbalanced Classification, Deep Learning

1 Introduction

Classification is a common problem in real life, there are many algorithms such as XGBoost, SVM, ELM and deep learning have been widely used in classification, and most of them were proposed under the assumption that the data is balanced. But the phenomenon of class imbalance can be easily seen in classification, such as clinical diagnosis, credit card fraud identification, telecommunications fraud and CTR prediction. In binary classification with imbalanced data, traditional algorithms tend to classify many instances belonging to minority class as majority class. In other words, few instances are classified as minority class, which illustrates that the accuracy of minority class is ignored extremely. However, minority instances are much more important than majority instances in sometimes. Therefore, we expect to obtain higher accuracy of minority class in the results of binary classification.

Recently, many approaches have been proposed to explore this problem. And these approaches can be roughly divided into two categories, sampling methods and cost-sensitive algorithms. The sampling methods obtain a balanced distribution of data by reducing or producing instances of different classes. In order to balance the imbalanced samples, Hui *et al*. [1] employed random oversampling to increase the volume of minority samples, it may bring overfitting problem. As a synthetic minority oversampling technique, SMOTE is a popular method of oversampling. It creates synthetic examples by interpolation rather than by oversampling, but it will definitely lead to the increment of training time and a lot of work should be done on data preprocessing. Undersampling [2] is also commonly used to get balanced distributions of classes, but it may lead to the loss of original information. In addition to sampling methods, many cost-sensitive algorithms have been proposed for imbalance learning. For instance, weighted ELM was proposed by assigning different weights for each instance; Yang *et al*. [3] employed lopsided margins to derive a more general model for SVM cost-sensitive class-imbalanced learning and cost-sensitive decision tree ensembles for effective imbalanced classification was proposed by Krawczyk [4].

Compared with other classification algorithms, deep learning can not only automatically extract the effective features of data, but also has super nonlinear fitting ability. However, it is still a challenge to address the problem of imbalance learning in deep neural networks (DNN) [5]. Many approaches based on oversampling and undersampling have been introduced for imbalance learning in DNN. As is discussed above, these methods based on sampling have some drawbacks. Several cost-sensitive algorithms based on DNN have been proposed to improve the performance of classifier. For example, two novel loss functions, mean false error and mean squared false error, were proposed by Wang [6]. Castro [7] presented a new cost-sensitive algorithm based on a joint objective function that uses a cost parameter to distinguish the importance of class errors. Ghazikhani [8] gave two separate cost-sensitive strategies, a fixed and an adaptive misclassification cost matrix, for imbalance learning.

However, the cost-sensitive deep learning mentioned above only focuses on the cost of misclassification between class, while they ignore the difference of the misclassification cost among samples within same class. In this paper, a novel cost-sensitive deep learning method based on stack denoising autoencoders (SDAE) [9] is proposed for classification on imbalanced data. A composite cost matrix is embedded into the loss function to adjust the outputs of SDAE (the outputs of output layer), by balancing the error of predicted probability distribution and the real probability distribution. Cross entropy is selected as the loss function due to its excellent property for classification.

The paper is organized as follows. In Section 2, we first formalize the cost-sensitive classification problem and review related works. Section 3 introduce the details of the proposed method CCS-DNN. The benchmark datasets and the experiment results are illustrated in section 4. Finally, section 5 concludes the paper and gives the future works.

2 Related Works

We will illustrate the cost-sensitive classification problem and cost-sensitive neural network before introducing our new proposed works.

2.1 Cost-sensitive Classification

We first introduce the classification problem and then extend it to the cost-sensitive setting. The N-class classification problem comes with a size-M training set $\{(x_m, y_m)\}_{m=1}^{M}$, where each input vector x_m is within an input space X, and each label y_m is within a label space $Y = \{1, 2, \cdots, N\}$. The goal of classification is to train a classifier $g : X \rightarrow Y$ such that the expected error $[[y \neq g(x)]]$ on test examples (x, y) is small.

Cost-sensitive classification extends classification by penalizing each type of misclassification error differently based on some given costs. Specifically, consider a N by N cost matrix C, where each entry $C_{(p,n)} \in [0, \infty)$ denotes the cost for predicting a class p example as class n. The goal of cost-sensitive classification is to train a classifier g such that the expected cost $C(y, g(x))$ on test examples is small.

The cost-matrix setting is also called cost-sensitive classification with class-dependent costs. Another popular setting is to consider example-dependent costs, which means coupling an additional cost vector $c \in [0, \infty)^N$ with each example (x, y), where the n-th component $c[n]$ denotes the cost for classifying x as class n. During training, each c_m that accompanies (x_m, y_m) is also fed to the learning algorithm to train a classifier g such that the expected cost $c[g(x)]$ is small with respect to the distribution that generates (x, y, c) tuples. The cost-matrix setting can be cast as a special case of the cost vector setting by defining the cost vector in (x, y, c) as row y of the cost matrix C.

In this work, we will eventually propose a composite cost-sensitive deep learning algorithm that works under the more appropriate cost-vector setting.

2.2 Cost-sensitive Neural Network

Few existing works have studied cost-sensitive classification using neural networks. Zhou and Liu [10] focused on studying the effect of sampling and threshold-moving to tackle the class imbalance problem using neural network as a core classifier rather than proposing general cost-sensitive neural network algorithms. Kukar and Kononenko [11] proposed four approaches of modifying neural networks for cost-sensitivity. The first two approaches train a usual multiclass classification neural network, and then make the prediction stage of the trained network cost-sensitive by including the costs in the prediction formula; the third approach modifies the learning rate of the training algorithm base on the costs; the fourth approach, called MIN (minimization of the misclassification costs), modifies the loss function of neural network training directly. Among the four proposed algorithms, MIN consistently achieves the lowest test cost. Chung and Lin [12] proposed a model that combines cost sensitive and deep neural network DNN, and is also the first cost sensitive deep learning algorithm that is truly improved from the algorithm itself, pre-training is used to obtain the data with cost sensitivity, so that the whole model can maintain cost sensitivity in the training process. Zhang [13] applied differential evolution algorithm to the optimizing of cost matrix to solve the problem of unbalanced data. Khan *et al*. [5] proposed a cost sensitive deep neural network that can automatically learn data feature representation, which can not only solve dichotomy and multi-classification problems but also has strong robustness. The model is trained through the joint optimization of the cost parameters and the parameters of the network itself, and the co-optimization of the two parameters is accomplished by keeping one parameter unchanged and training the other. In addition, this training method can be applied to different loss functions.

Different from the methods mentioned above, this paper will embed both example-dependent costs and class-dependent costs into neural network to design an cost-sensitive algorithm. Meanwhile, the cost matrix can be adjusted adaptively.

3 CCS-DNN

This section presents a novel composite cost-sensitive deep neural network (CCS-DNN). The architecture of CCS-DNN appears in Figure 1.

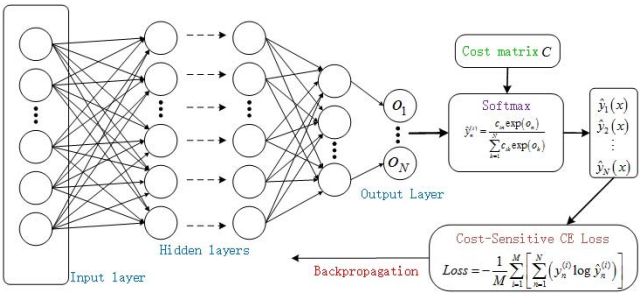

Fig. 1: The architecture of CCS-DNN

The network architecture consists of a stacked denoising autoencoders (SDAE) that extract robust features, and a cost-sensitive loss function that improve the sensitivity to imbalanced data. Each of the different components is described in the following subsections.

3.1 Deep neural network

We stack several denoising autoencoders (DAEs) to form a deep network, where each layer receives its input from the latent representation of the previous layer, except the weight layers, the network has an extra cost-sensitive loss layer which will be introduced in subsection 3.3. First, we use greedy layer-wise pre-training to initialize the network parameters of each weight layer. Then, according to the cost-sensitive loss function, BP algorithm is used to fine-tune the network parameters. Since there are a huge number of parameters in the network, it will take a lot of time to optimize the parameters. Therefore, we freeze part of the network after pre-training, and only fine-tune the parameters of the last two weight layers. The DNN trained without the cost-sensitive loss layer will be used as the baseline DNN in our experiments. Note that the baseline DNN architecture is exactly the same as the CCS-DNN, except that the final layer is a sample CE loss layer.

3.2 Cost Matrix

In this subsection, we propose a new cost matrix C for binary classification. For simplicity, we consider binary classi-

fication problems with classes 1 (minority class, which was treated as the positive class) and 2 (majority class, which was treated as the negative class). Then the cost matrix can be defined as formula (1).

$$C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ \vdots & \vdots \\ c_{M1} & c_{M2} \end{bmatrix} \tag{1}$$

In order to distinguish among different misclassified samples, a predefined cost value can be assigned to each sample in the dataset. The cost for a sample according to its actual class *vs*. its predicted class is shown in table 1.

Table 1: Cost matrix for imbalanced classification

	True Positive ($y_i = 1$)	True Negative ($y_i = 2$)
Predicted Positive ($\hat{y}_i = 1$)	$c_{i1} = 1$	$c_{i2} = c_i + c^{NP}$
Predicted Negative ($\hat{y}_i = 2$)	$c_{i1} = c_i + c^{PN}$	$c_{i2} = 1$

Where $c_{in}(n \in [1,2], i \in [1, M]$, M being the number of samples) denotes the cost of dividing the $i-th$ sample into the class n, c^{PN} and c^{NP} denote the class-dependent cost which can be optimized adaptively, c_i denotes the example-dependent cost which is defined by the distribution of samples in the feature space.

However, it will consume a lot of computing resources and time to calculate the example-dependent cost of each sample, especially when it comes to big datasets. In consideration of imbalanced classification, the samples in minority class tend to attract more attention. So we set the example-dependent cost of the samples in majority class to zero, and only compute the example-dependent cost of the samples in minority class. First, taking Euclidean distance as the metric, we use KNN algorithm to find the nearest neighbors of each minority sample, then count the number $n_p^{(i)}$, $n_n^{(i)}$ of samples belonging to minority class and majority class in the nearest k neighbors separately. Next, we calculate the average distance $d_n^{(i)}$ from each sample, which belongs to negative class, of k neighbors to that sample, and the average distance $d_p^{(i)}$ from each sample, which belongs to positive class, of k neighbors to that sample. Note that these distances are calculated in the feature space where each sample is a 2-D feature vector obtained from the pretrained SDAE. In order to unify the scale, we map the non-zero example-dependent cost to the interval $[C_{\min}, C_{\max}]$ which denotes the search domain of the class-dependent cost. Formally, the example-dependent costs c_i of each sample can be defined as formula (2).

$$c_i = \begin{cases} 0 & y_i = 2 \\ \frac{C_{\max} - C_{\min}}{\xi_{\max} - \xi_{\min}} (\xi_i - \xi_{\min}) + \xi_{\min} & y_i = 1; n_n^{(i)} n_p^{(i)} \neq 0 \\ 0 & y_i = 1; n_n^{(i)} n_p^{(i)} = 0 \end{cases} \tag{2}$$

Where ξ_i denotes ratio between $d_n^{(i)}$ and $d_p^{(i)}$, $\xi_{\max}$ and $\xi_{\min}$ denote the maximum and the minimum value in ξ_i.

3.3 Cost Sensitive Loss Function

Cost sensitive learning is addressed during training. S-DAE network obtains robust features o by encoding the input data layer by layer, then a modified softmax function is used to calculate the posterior probability.

$$\hat{y}_n^{(i)} = \frac{c_{in} \exp(o_n)}{\sum_{k=1}^{N} c_{ik} \exp(o_k)} \tag{3}$$

Where $\hat{y}$ is the posterior probability over all possible classes given a sample x. The cost-sensitive error function is expressed as the loss function over the training set:

$$E(\theta, C) = \frac{1}{M} \sum_{i=1}^{M} \ell(y^{(i)}, \hat{y}_{(\theta,C)}^{(i)}) \tag{4}$$

where $\theta = \{W, b\}$ are the weights and biases in the network, and C is the cost matrix. $y \in \{0,1\}^{1 \times N}$ is the desired output $(s.t. \sum_{i=1}^{N} \mathrm{y}_n = 1)$, and N denotes the total number of neurons in the output layer, which equals the number of classes. Therefore, the cost-sensitive classification optimization objective is:

$$(\theta^*, C^*) = \arg\min_{\theta, C} E(\theta, C) \tag{5}$$

where the optimal parameters (θ^*, C^*) are the objectives of the learning algorithm, giving the minimum possible cost E in formula (5). The loss function $\ell(\cdot)$ in formula (4) could be any suitable loss function, such as the cross-entropy loss function used here.

Cost-sensitive Cross Entropy Loss: The cross-entropy loss function maximizes the predictions for the desired output and is given by formula (6).

$$\ell(y, \hat{y}) = -\sum_{n=1}^{N} (y_n \log(\hat{y}_n(\theta, C))) \tag{6}$$

The output relates to the previous layer output via the softmax function to calculate the probability distribution of different possible outcomes.

3.4 Optimal Parameters Learning

The goal in optimizing the learning parameters is to jointly learn the two types of parameters using the functions in CCS-DNN: the stacked denoising autoencoders parameters $\theta = \{W, b\}$, and the cost matrix parameters $C_{class} = \left[c^{PN}, c^{NP}\right]$. The two types of parameters are solved alternately by keeping one fixed and minimizing the loss with respect to the other.

Since the costs affect the softmax output $\hat{y}$ but the gradient formulas remain unchanged [5]. Therefore stochastic gradient descent with a backpropagation error is used to optimize θ.

As for the optimization of cost matrix , we choose the particle swarm optimization (PSO) as our optimization method, because it is very mature and easy to implement. In addition, many experiments claim that PSO has equal effectiveness but superior efficiency over GA. The position of the particle represents the cost parameters $C_{class} = \left[c^{PN}, c^{NP}\right]$,

and the range of c^{NP} and c^{PN} are empirically chosen to $[1, 10]$. Because the accuracy is not a suitable measure of imbalanced classification, we inject the appropriate measures, G-mean into the fitness function of the classifier in the training with PSO to adjust the position of a particle. The G-mean is the geometric mean of accuracies measured separately on each class, which is commonly utilized when performance of each class is concerned and expected to be high simultaneously [14].

Algorithm 1 Iterative Optimization for Parameters (θ, C_{class})

Require:
Training set (X_T, Y_T), Validation set (X_V, Y_V), Max epochs m, Learning rate γ_θ for θ

Ensure:
Learned parameters $(\theta^*, {C_{class}}^*)$

$Net \leftarrow$ construct DNN
$\theta \leftarrow$ Pre-train Net
Freeze the first $n-2$ weight layers
Initialize PSO population and particle swarm velocity
Looping in number of epochs
for $e \in [1, m]$ **do**
  for each particle(C_{class}) **do**
    $C \leftarrow C\,(C_{example}, C_{class})$
    for $b \in [1, batchsize]$ **do**
      $out_b \leftarrow$ forward-pass(X_T, Y_T, Net, θ)
      $grad_b \leftarrow$ backward-pass$(out_b, X_T, Y_T, Net, \theta, C)$
      $\theta^* \leftarrow$ update-NetParams$(Net, \theta, \gamma_\theta, grad_b)$
      $\theta \leftarrow \theta^*$
    end for
    $G - mean \leftarrow$ forward-pass (X_V, Y_V, Net, θ)
    Update the optimal fitness of the individual and the optimal position of the individual
  end for
  Update global optimal fitness and global optimal position
  Update the position of particles and particle swarm velocity
end for
return (θ^*, C^*_{class})

4 Experiments

In this section, we will validate the performance of CCS-DNN on classification with imbalanced data. All experiments are carried out in TensorFlow environment on a PC with 3.2GHz Intel Core i7 CPU, 16GB RAM, and windows 10 64-bit ultimately.

4.1 Datasets Description

To evaluate the classification performance of our proposed methods in different binary class classification tasks, and to compare with other methods specifically devised for imbalanced data. We have experimented on 5 imbalanced datasets from the UCI repository [15] and a real credit card fraud detection dataset from kaggle [16]. As for the datasets which have multi-class labels, we convert them into the datasets suitable for binary classification by setting one of these classes as a minority and all the others as a majority. The detailed information of experimental datasets are given in Table 2, where the dataset name is appended with the label of the minority class.

The imbalance ratio (IR) denotes to the size ratio between the majority class and minority class.

Table 2: The detailed information of datasets

Datasets	Features	Instances	Imbalance Ratio
Segment-1	19	2310	6
Sick	29	3772	15
Hypothyroid	25	3163	20
Letter-26	16	20000	26
Abalone-19	8	4177	130
Credit card	30	284807	578

4.2 Evaluation Metrics

Considering an imbalance classification problem, experimental results are evaluated by three appropriate metrics: the area under curve (AUC), F-measure and G-mean. The calculation of those metrics mentioned above is based on confusion matrix. And for simplicity, true positive, false negative, false positive and true negative are denoted as TP, FN, FP and TN respectively. Thus, the metrics mentioned above can be formulated in formula (7) to formula (8) respectively.

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{7}$$

$$G - mean = \sqrt{\frac{TP}{TP + FN} + \frac{TN}{TN + FP}} \tag{8}$$

4.3 Experimental Results and Analysis

In this subsection, we investigate the performance of DNN in different settings that include DNN, Smote-DNN, CS-DNN (which is exactly the same as the CCS-DNN, except that the cost matrix consists only of class-dependent cost) and CCS-DNN. We report the results over a total of runs on 6 imbalanced benchmark datasets in terms of AUC, F1-score and G-mean, respectively. A detailed summary can be found at table 3, with the best results being highlighted in boldface.

With each dataset, we observed that CCS-DNN showed obvious improvements in terms of AUC, F-measure and G-mean in the datasets with an imbalance distribution. Additionally, the more imbalanced the dataset is, the better the CCS-DNN performance than the other methods, this is a promising result for effectively classifying imbalanced data sets.

In order to further evaluate the performance of CCS-DNN, we applied it to the same dataset with different degrees of imbalance. To represent different levels of imbalance in the data distribution, we adjust the value of IR by reducing random samples of the minority class in datasets. The experimental results are shown in figure 2 and figure 3.

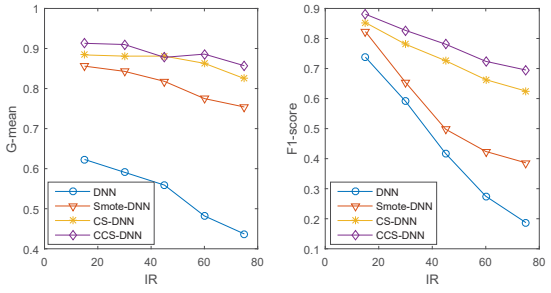

Fig. 2: Evaluation on sick dataset

Table 3: experimental results

Datasets	DNN			Smote-DNN			CS-DNN			CCS-DNN		
	AUC	F1	G-mean	AUC	F1	G-mean	AUC	F1	G-mean	AUC	F1	G-mean
Segment-1	0.989	0.906	0.993	0.992	0.921	0.995	**0.996**	0.929	0.997	0.995	**0.932**	**0.998**
Sick	0.724	0.587	0.623	0.816	0.658	0.812	0.888	0.818	0.911	**0.945**	**0.839**	**0.913**
Hypothyroid	0.873	0.717	0.849	0.925	0.767	0.932	0.954	0.785	**0.960**	**0.973**	**0.792**	0.959
Letter-26	0.985	0.738	0.802	0.988	0.823	0.874	**0.991**	0.853	0.943	0.990	**0.881**	**0.945**
Abalone-19	0.759	0.694	0.763	0.793	0.774	0.798	0.833	0.807	**0.892**	**0.875**	**0.850**	0.891
Credit Card	0.867	0.233	0.416	0.981	0.188	0.938	0.974	0.472	0.985	**0.984**	**0.568**	**0.988**

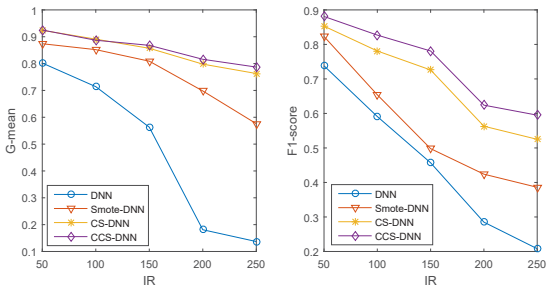

Fig. 3: Evaluation on letter-26 dataset

From figure 2 and figure 3, we can observe that as the imbalance ratio of the dataset increases, the performance of each classification method becomes worse and worse in terms of F1-score and G-mean, but our method is better than other methods in most cases.

Although the CCS-DNN performs well in imbalanced classification, it still has some disadvantages, especially in time complexity. Since the PSO algorithm is introduced to optimize the cost matrix, the time complexity of the CCS-DNN is $O\left(m * n * b\right)$ (where m denotes the the number of iterations in PSO algorithm, n is the number of particles and b is the number of iterations in stochastic gradient descent). Compared with the time complexity of baseline DNN, the time complexity of CCS-DNN is obviously much higher.

5 Conclusion

This paper presents a novel composite cost-sensitive deep neural network (CCS-DNN) that deals with imbalanced data problems commonly found in real-world business. A specifically-designed cost-sensitive matrix, which is composed of example-dependent costs and class-dependent costs, is embedded into the cross entropy loss function, then CCS-DNN feeds the cost-sensitive loss function into a network architecture to improve the classification. Alternating optimization algorithms then efficiently learn a resulting imbalanced cost-sensitive matrix along with network parameters at epoch level. The results of experiments with imbalanced datasets show that our approach demonstrates superior performance over the other baseline methods. Additionally, the more imbalanced the dataset is, the better the CCS-DNN performance than the other methods. In the future, we will focus on how to get more accurate cost matrix and how to accelerate the optimization process of cost matrix. What's more, we will try to extend the CCS-DNN to the multiclass classification problem.

References

[1] H. Li, J. Li, P.C. Chang, and J. Sun, Parametric prediction on default risk of Chinese listed tourism companies by using random oversampling, isomap, and locally linear embeddings on imbalanced samples, *International Journal of Hospitality Management*, 2013, 35(16): 141-151.

[2] P. Kaur, A. Gosain, Comparing the Behavior of Oversampling and Undersampling Approach of Class Imbalance Learning by Combining Class Imbalance Problem with Noise, *ICT Based Innovations*, 2018, 47(8): 34-46.

[3] C. Y. Yang, J. S. Yang, J.J. Wang, Margin calibration in SVM class-imbalanced learning, *Neurocomputing*, 2009, 73(1): 397-411.

[4] B. Krawczyk, M. Wozniak, and G. Schaefer, Cost-sensitive decision tree ensembles for effective imbalanced classifica-tion, *Applied Soft Computing*, 2014, 14(1): 554-562.

[5] S.H. Khan, M. Hayat, M. Bennamoun, et al, Cost-Sensitive Learning of Deep Feature Representations From Imbalanced Data, *IEEE Transactions on Neural Networks and Learning Systems*, 2018, 29(8): 3573-3587.

[6] S. Wang, W. Liu, J. Wu, et al, Training deep neural networks on imbalanced data sets, *IEEE International Joint Conference on Neural Networks*, 2016, 85(19): 4368-4374.

[7] C. L. Castro, and A. P. Braga, Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data, *IEEE Transactions on Neural Networks and Learning Systems*, 2013, 24(6): 888-899.

[8] A. Ghazikhani, R. Monsefi, and H. S. Yazdi, Online cost-sensitive neural network classifiers for non-stationary and imbalanced data streams, *Neural Computing and Applications*, 2013, 23(5):1283-1295.

[9] P. Vincent, H. Larochelle, et al, Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion, *Journal of Machine Learning Research*, 2010, 11(12): 3371-3408.

[10] Z. H. Zhou, X. Y. Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, *IEEE Transactions on Knowledge and Data Engineering*, 2006, 18(1): 63-77.

[11] M. Kukar, I. Kononeko. Cost-sensitive learning with neural networks, *ECAI*, 1998, 15(27): 88-94.

[12] Y. A. Chung, H. T. Lin, Cost-aware Pre-training for Multiclass Cost-sensitive Deep Learning, *arXiv*, 2015, 9(337): 1511.

[13] C. Zhang, K. C. Tan, et al, A Cost-Sensitive Deep Belief Network for Imbalanced Classification, *IEEE Transactions on Neural Networks and Learning Systems*, 2018, 31(18): 218-232.

[14] B. Yuan, W. Liu, A Measure Oriented Training Scheme for Imbalanced Classification Problems, *Pacific-Asia Conference on Knowledge Discovery and Data Mining Workshop*, 2011: 293C303.

[15] M. Lichman, UCI Machine Learning Repository, *https://archive.ics.uci.edu/ml/*.

[16] *https://www.kaggle.com/mlg-ulb/creditcardfraud*.