SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

November 2023, Vol. 66 212201:1–212201:14
https://doi.org/10.1007/s11432-021-3775-4

Reinforcement learning-based cost-sensitive classifier for imbalanced fault classification

Xinmin ZHANG, Saite FAN* & Zhihuan SONG

State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou 310027*, China*

Received 11 July 2021/Revised 20 August 2022/Accepted 4 April 2023/Published online 17 October 2023

Abstract Fault classification plays a crucial role in the industrial process monitoring domain. In the datasets collected from real-life industrial processes, the data distribution is usually imbalanced. The datasets contain a large amount of normal data (majority) and only a small amount of faulty data (minority); this phenomenon is also known as the imbalanced fault classification problem. To solve the imbalanced fault classification problem, a novel reinforcement learning (RL)-based cost-sensitive classifier (RLCC) based on policy gradient is proposed in this paper. In RLCC, a novel cost-sensitive learning strategy based on policy gradient and the actor-critic of RL is developed. The novel cost-sensitive learning strategy can adaptively learn the cost matrix and dynamically yield the sample weights. In addition, RLCC uses a newly designed reward to train the sample weight learner and classifier using an alternating iterative approach. The alternating iterative approach makes RLCC highly flexible and effective in solving the imbalanced fault classification problem. The effectiveness and practicability of the proposed RLCC method are verified through its application in a real-world dataset and an industrial process benchmark.

Keywords imbalanced fault classification, fault diagnosis, industrial process monitoring, deep reinforcement learning, cost-sensitive learning, policy gradient, sample weights

Citation Zhang X M, Fan S T, Song Z H. Reinforcement learning-based cost-sensitive classifier for imbalanced fault classification. Sci China Inf Sci, 2023, 66(11): 212201, https://doi.org/10.1007/s11432-021-3775-4

1 Introduction

With the widespread use of the Internet of Things, cloud computing, and new measurement devices, large amounts of data have been recorded and collected in the modern manufacturing industry [1]. Data are key enablers for driving the development of smart manufacturing and are used for process monitoring [2], industrial safety analysis [3], quality prediction [4], and optimization [5]. Data-driven fault classification is a significant application of industrial process data analysis techniques and fault diagnosis domains [6]. The goal of fault classification is to distinguish faults from all data using easy-to-measure process variables. Compared with the traditional first-principle-based modeling methods, which use prior knowledge or experiences, data-driven methods are characterized by flexibility, ease of use, and low cost [7]. However, most real-life industrial process data usually exhibit a skewed distribution or an extremely imbalanced class distribution. Hence, if the classes are assumed to be of equal importance, the same error classification cost will be implicitly assigned to all classes of errors. The classifier tends to correctly classify the highly frequent (majority) classes rather than the infrequent (minority) classes. Therefore, there is an urgent need to improve the overall accuracy of the classifier without unduly sacrificing the accuracy of any majority or minority class. As illustrated in [8], imbalanced fault classification is a key issue in the fault diagnosis community and should be given more research attention. To solve the imbalanced classification problem, building a model that can correctly classify infrequent but important fault samples in the imbalanced dataset is the key [9]. Several methods have been proposed to solve this problem. We can briefly divide these methods into three categories: (i) data-level methods, (ii) algorithm-level methods, and (iii) hybrid methods [10]. Data-level methods tend to mitigate the degree of imbalance through

* Corresponding author (email: fansaite@zju.edu.cn)

various data sampling methods. Algorithm-level methods usually use weights or cost models, which involve modifying basic learners or their output to reduce bias toward the majority group. Hybrid methods strategically combine both data-level techniques and algorithm-level methods [10]. To solve imbalanced fault classification problems, Wang et al. [11] proposed an active learning framework based on alternative important sampling; the framework involved selecting important majority-class instances and generating informative minority-class instances. Jiang et al. [12] proposed a novel data augmentation classifier based on generative adversarial networks to solve the imbalanced classification problem. Fan et al. [13] proposed a novel general imbalanced sample selection strategy (DiagSelect) based on deep reinforcement learning (RL) to implement imbalanced sample selection. Yue et al. [14] proposed a new computer-aided diagnosis method for automated endoscopic image classification by introducing a novel class imbalance loss to the classical deep neural network. Santos et al. [15] argued that although some insightful information can be gained from related studies, the joint effect of class overlap and imbalance is still not fully understood and advocated for the need to move toward a unified view of the class overlap problem in imbalanced domains. In this paper, we focus on the algorithm-level method.

Cost-sensitive learning is a type of algorithm-level method that deals with imbalanced classification problems by incurring different costs for different classes. Cost-sensitive learning has a strong theoretical appeal, which tackles the problem of the traditional error-based learning algorithm from the perspective of an algorithm principle design. A common strategy of the cost-sensitive learning methods is to intentionally increase the weights of samples with high misclassification costs during a boosting process [16]. For example, AdaCost [17] uses the cost of misclassifications to update the training distribution on successive boosting rounds. Krempl et al. [18] proposed a cost-sensitive probabilistic active learning, which determines the optimal misclassification loss with the prior domain knowledge. Castro et al. [19] proposed a cost-sensitive multilayer perceptron (CSMLP), which uses a single cost parameter to distinguish the importance of class errors. Zheng [20] proposed a cost-sensitive boosting neural network (CSBNN), which incorporates the weight-updating rule of a boosting procedure to associate the samples with misclassification costs. Zhang et al. [21] proposed a cost-sensitive deep belief network with differential evolution, which directly adds the cost-sensitive functions to its classification paradigm and employs differential evolution to optimize the cost matrix. Although these methods have some effectiveness in solving the imbalanced fault classification problems, several problems that still plague imbalanced cost-sensitive learning are listed as follows: (1) Misclassification costs are usually unknown. (2) Setting a proper cost matrix is usually difficult [22] and requires high expertise and experience. (3) The assignment of costs to different classes, even samples, is usually conducted before the training process and is independent of the training process. (4) Even if the cost matrix can be determined via boosting, reselecting the best division point of the current base classifier each time is difficult. These issues tend to hinder cost-sensitive learning methods. To overcome these problems, we propose an unbiased classification model that can improve the recognition ability of under-represented class samples. Furthermore, developing a cost matrix that can accurately describe the costs of each sample by alternately iterating the sample weight learner and the classifier in the training process is desirable.

Therefore, we propose a novel RL-based cost-sensitive classifier (RLCC) based on policy gradient and an actor-critic to improve the performance of imbalanced fault classification methods. In RLCC, a novel cost-sensitive learning strategy that can adaptively learn the cost matrix and dynamically yield the sample weights is developed. First, the REINFORCE loss [23] of deep RL is utilized as the loss of the actor network for learning sample weights according to the designed reward. Second, the reward is calculated using the designed reward function. Third, multilayer perceptron (MLP) is used as the classifier to build a critic network. The loss function of the critic network is the novel weighted cross-entropy function, which combines the output of the MLP classifier with sample weights. By learning the sample weights alternately based on the actor-critic mechanism, the MLP classifier performance can be improved.

The main contributions of this paper are summarized as follows: (1) a novel cost-sensitive strategy based on policy gradient is designed to learn sample weights; (2) an alternating iterative approach of the actor-critic mechanism is used in the novel cost-sensitive strategy; the approach can adaptively learn the cost matrix and dynamically yield the sample weights; (3) a newly designed reward is used to guide the sample weight learning process in the actor-critic mechanism; (4) the proposed method is evaluated on a real-world dataset and an industrial process benchmark.

The rest of this paper is organized as follows. In Section 2, the cost-sensitive learning framework and the basic formulation of the cost matrix are briefly introduced. In Section 3, details of the proposed method for imbalanced fault classification are presented. In Section 4, the effectiveness of the proposed

method is verified using a real-world dataset and an industrial process benchmark. In Section 5, the conclusion is presented.

2 Preliminary

2.1 Cost-sensitive learning framework

Cost-sensitive learning considers the varying costs of different misclassifications. The principle of the cost matrix is that it encodes the penalty of misclassifying samples from one class as those from another class. Let $\boldsymbol{\xi}_{i,j}$ denote the cost of predicting a sample from classes i to j. For example, $\boldsymbol{\xi}_{+,-}$ is the cost of misclassifying a positive (minority class) sample as a negative (majority class) sample, and $\boldsymbol{\xi}_{-,+}$ is the cost of misclassifying a negative sample as a positive sample. In solving the issue of imbalanced classification, the discrimination of positive samples is more important than that of negative samples. Therefore, the cost of misclassifying a positive sample is greater than that of misclassifying a negative sample (i.e., $\boldsymbol{\xi}_{+,-} > \boldsymbol{\xi}_{-,+}$), that is, classifying a correct negative sample usually presents a low weight. Hence, cost-sensitive learning minimizes high classification errors or total misclassification cost under this assumption [24]. A cost-sensitive learning method considers the cost matrix during the training process and enables the classifier to have the lowest total error. The cost-sensitive learning methods can be divided into three groups:

(1) Making a specific classifier prune subtrees or choosing the best attribute [25];
(2) Using Bayes risk theory and assigning each sample to its lowest risk class [26];
(3) Weighting the data space [27].

The first and second groups of methods adapt to the existing learning methods at the algorithm level. These groups of methods assume that the cost matrix is known for different types of errors or samples. However, the cost matrix is often unavailable for a dataset. The third group of methods is also based on the cost-sensitive learning of sample weights [27], in which sample-dependent costs are converted into sample weights. The weighted training samples are used in the standard classifier. Our research focuses on the cost-sensitive learning of the third group.

The weight of the data space is described as sample weights. The bias of the training process is modified using the misclassification cost. Thus, the modified error is biased toward high weight, which can be explained based on the translation theorem [27]. To distinguish fault samples from the normal space, let us define a data space with domain $\boldsymbol{x} \times \boldsymbol{y} \times \boldsymbol{\xi}$ as the cost-space, where $\boldsymbol{x}$ is the input space, $\boldsymbol{y}$ is the output space, and $\boldsymbol{\xi}$ is the cost associated with the mislabeling of samples. If we draw samples from a distribution D in the cost space, we can have another distribution $\hat{D}$ in the normal space:

$$\hat{D}(\boldsymbol{x}, \boldsymbol{y}) \equiv \frac{\boldsymbol{\xi}}{E_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi}\sim D}[\boldsymbol{\xi}]} D(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}), \tag{1}$$

where $E_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{\xi}\sim D}[\boldsymbol{\xi}]$ is the expectation of cost values. According to the translation theorem, the optimal error rate of the classifier for $\hat{D}$ is an optimal cost minimizer for D. Therefore, when we update the weight of each sample, choosing the hypothesis to minimize the rate of errors under $\hat{D}$ is equivalent to choosing the assumption to minimize the expected cost under D. Using this framework, the cost-sensitive boosting methods, such as AdaCost [17], AdaC1, AdaC2, and AdaC3 [28], and CSB1 and CSB2 [29], have been proposed previously. AdaUBoost (AdaBoost with an unequal loss function) [30] optimizes an unequal loss on the imbalanced training dataset via preprocessing, and manipulates the training distribution within successive boosting rounds. AsymBoost (Asymmetric AdaBoost) [31] uses the asymmetric misclassification cost to update the training distribution on successive boosting rounds. In addition, Zhou et al. [32] used the manually designed cost matrix based on expert judgment, which is a tedious task for several classes.

2.2 Basic formulation of cost matrix

In cost-sensitive learning, given an input sample $\boldsymbol{x}$ and the cost matrix $\boldsymbol{\xi}$, the goal of the classifier is to minimize the expected risk $\mathcal{R}(\hat{\boldsymbol{y}}|\boldsymbol{x})$, where $\hat{\boldsymbol{y}}$ is the predicted labels made by the classifier. The expected risk can be expressed as

$$\mathcal{R}(\hat{\boldsymbol{y}}|\boldsymbol{x}) = \sum_{i}^{C} \boldsymbol{\xi}_{\boldsymbol{y},\hat{\boldsymbol{y}}_i} p(\hat{\boldsymbol{y}}_i|\boldsymbol{x}), \tag{2}$$

where $\boldsymbol{\xi}_{\boldsymbol{y},\hat{\boldsymbol{y}}_i}$ represents the cost associated with predicting $\boldsymbol{y}$ as $\hat{\boldsymbol{y}}_i$, and $p(\hat{\boldsymbol{y}}_i|\boldsymbol{x})$ is the posterior probability over predicted labels i given sample $\boldsymbol{x}$. According to the Bayes decision theory, an ideal classifier will give a decision in favor of the $\hat{\boldsymbol{y}}^*$ with the minimum expected risk,

$$\hat{\boldsymbol{y}}^* = \arg\min_{\hat{\boldsymbol{y}}} \mathcal{R}(\hat{\boldsymbol{y}}|\boldsymbol{x}) = \arg\min_{\hat{\boldsymbol{y}}} E_{\boldsymbol{x},D}[\boldsymbol{\xi}], \tag{3}$$

where D is the cost-space domain $\boldsymbol{x} \times \boldsymbol{y} \times \boldsymbol{\xi}$. Because $p(\hat{\boldsymbol{y}}|\boldsymbol{x})$ cannot be easily found, we use the empirical distribution derived from the training dataset. Given a training dataset $D_{\text{trn}} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$, N is the total number of samples, and the empirical risk is defined as follows:

$$\mathcal{R}(\hat{\boldsymbol{y}}|\boldsymbol{x}) = E_{\boldsymbol{x},D}[\boldsymbol{\omega}] = \frac{1}{N}\sum_{i=1}^{N} \ell(\boldsymbol{\xi}, \boldsymbol{y}_i, \boldsymbol{o}_i), \tag{4}$$

where $\boldsymbol{o}_i$ is the output of the neural network for sample i, and $\ell(\cdot)$ is the misclassification error (0-1 loss) or a surrogate loss function. The sample weights $\boldsymbol{\omega}$ are calculated using the cost matrix $\boldsymbol{\xi}$ according to specified rules. The cost matrix is converted into a cost parameter for each class as follows:

$$\boldsymbol{\xi}_i = \sum_{j}^{C} \boldsymbol{\xi}_{i,j}, \tag{5}$$

where C is the number of classes. The sample weights corresponding to the i-th class are shown as follows:

$$\boldsymbol{\omega}_i = \boldsymbol{\xi}_i \frac{N_i}{\sum_j^C \boldsymbol{\xi}_j N_j}, \tag{6}$$

where N_i is the sample size of class i.

3 RL-based cost-sensitive classifier

3.1 The definition of RLCC

We consider the RL framework [33] to learn the cost matrix. For RL, the framework contains the state space $\boldsymbol{s}_t \in \boldsymbol{S}$, action space $\boldsymbol{a}_t \in \boldsymbol{A}$, and immediate rewards $\boldsymbol{r}_t \in \boldsymbol{R}$. The detailed definitions of RLCC are described as follows:

• State $\boldsymbol{S}$. The state is determined based on the training sample. At the beginning of the training process, the agent receives the first batch samples $\boldsymbol{x}_1$ as its initial state $\boldsymbol{s}_1$. The state $\boldsymbol{s}_t$ corresponds to the batch samples $\boldsymbol{x}_t$. When a new episode begins, the environment shuffles the order of samples in the training dataset.

• Action $\boldsymbol{A}$. The action of the agent is associated with the cost matrix $\boldsymbol{\xi}$. $\boldsymbol{A}$ is the action space of the sample weights based on the cost matrix $\boldsymbol{\xi}$. The action $\boldsymbol{a}_t$ taken by the agent is the sample weight.

• Reward $\boldsymbol{R}$. The reward $\boldsymbol{r}_t$ is the feedback of actions. To guide the agent to learn better actions in the imbalanced dataset, the absolute reward value of the minority-class samples is higher than that of the majority-class samples. When the agent correctly or incorrectly recognizes minority-class samples, the environment feeds back a larger reward or punishment to the agent.

• Episode. An episode in RL is a transition trajectory from the initial tuple to the terminal tuple $\{(\boldsymbol{s}_1, \boldsymbol{a}_1, \boldsymbol{r}_1), (\boldsymbol{s}_2, \boldsymbol{a}_2, \boldsymbol{r}_2), \ldots, (\boldsymbol{s}_N, \boldsymbol{a}_N, \boldsymbol{r}_M)\}$. M is the length of the episode. An episode ends when all samples in the training dataset are classified.

• Policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$. The policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ is a probability distribution, where $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$ denotes the probability of choosing action $\boldsymbol{a}$ in state $\boldsymbol{s}$ by an agent. The policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ can be considered a learner with the parameter $\boldsymbol{\theta}$.

With the abovementioned definitions, the imbalanced fault classification problem is formally defined to find the policy $\boldsymbol{\pi}^*$, which maximizes the cumulative rewards in the actor critic mechanism [34].

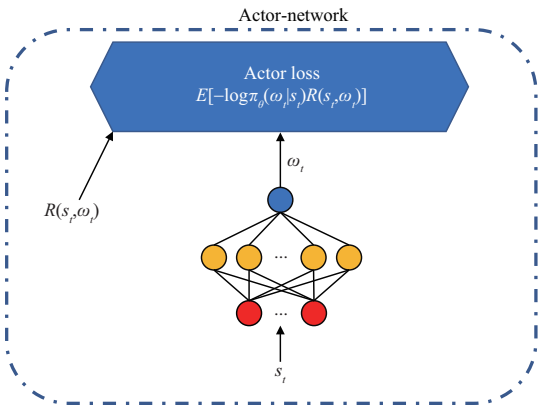

Figure 1 (Color online) Structure and loss function of the actor network.

3.2 Actor network of RLCC

In this paper, the action is the sample weight $\boldsymbol{\omega}$, which corresponds to the cost matrix $\boldsymbol{\xi}$. The elements of the sample weights are continuous values. The policy-based RL method [33] is selected to find the optimal action. The ultimate goal of RL is to maximize the rewards received. We assume that the objective function of RL is $J(\boldsymbol{\pi_\theta})$, and the ultimate goal is to maximize this objective function. $J(\boldsymbol{\pi_\theta})$ denotes the long-run average reward under policy $\boldsymbol{\pi_\theta}$. In the Markov decision process [35], the long-run average reward is defined as follows:

$$J(\boldsymbol{\pi_\theta}) = \sum_{\boldsymbol{s}\in\boldsymbol{S}} \boldsymbol{d}(\boldsymbol{s}) \sum_{\boldsymbol{\omega}\in\boldsymbol{D}} \boldsymbol{\pi_\theta}(\boldsymbol{\omega}|\boldsymbol{s})\boldsymbol{R}(\boldsymbol{s},\boldsymbol{\omega}). \tag{7}$$

To learn the optimal policy $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} J(\boldsymbol{\pi_\theta})$, the gradient of $J(\boldsymbol{\pi_\theta})$ is derived as follows:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\pi_\theta}) &= \nabla_{\boldsymbol{\theta}} \sum_{\boldsymbol{s}\in\boldsymbol{S}} \boldsymbol{d}(\boldsymbol{s}) \sum_{\omega\in D} \boldsymbol{\pi_\theta}(\boldsymbol{\omega}|\boldsymbol{s})\boldsymbol{R}(\boldsymbol{s},\boldsymbol{\omega}) \\
&= \sum_{\boldsymbol{s}\in\boldsymbol{S}} \boldsymbol{d}(\boldsymbol{s}) \sum_{\boldsymbol{\omega}\in\boldsymbol{D}} \nabla_{\boldsymbol{\theta}} \boldsymbol{\pi_\theta}(\boldsymbol{\omega}|\boldsymbol{s})\boldsymbol{R}(\boldsymbol{s},\boldsymbol{\omega}) \\
&= \sum_{\boldsymbol{s}\in\boldsymbol{S}} \boldsymbol{d}(\boldsymbol{s}) \sum_{\boldsymbol{\omega}\in\boldsymbol{D}} \boldsymbol{\pi_\theta}(\boldsymbol{\omega}|\boldsymbol{s}) \frac{\nabla_{\boldsymbol{\theta}} \boldsymbol{\pi_\theta}(\boldsymbol{\xi}|\boldsymbol{s})}{\boldsymbol{\pi_\theta}(\boldsymbol{\omega}|\boldsymbol{s})} \boldsymbol{R}(\boldsymbol{s},\boldsymbol{\omega}) \\
&= E\left[\nabla_{\boldsymbol{\theta}} \log \boldsymbol{\pi_\theta}(\boldsymbol{\omega}|\boldsymbol{s})\boldsymbol{R}(\boldsymbol{s},\boldsymbol{\omega})\right],
\end{aligned} \tag{8}$$

where $\boldsymbol{R}(\boldsymbol{s},\boldsymbol{\omega})$ denotes the single-stage expected rewards. $\boldsymbol{\pi_\theta}$ is a parameterized randomized stationary policy with parameter $\boldsymbol{\theta}$, which maps each state $\boldsymbol{s}$ to a probability distribution. $\boldsymbol{d} = \{\boldsymbol{d}(\boldsymbol{s}), \boldsymbol{s} \in \boldsymbol{S}\}$ denotes the unique stationary probability distribution of each time step. To obtain the optimal solution of (8) using gradient descent, we can learn the optimal policy $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}}[-J(\boldsymbol{\pi_\theta})]$ by the actor loss

$$\nabla_{\boldsymbol{\theta}}[-J(\boldsymbol{\pi_\theta})] = E\left[-\nabla_{\boldsymbol{\theta}} \log \boldsymbol{\pi_\theta}(\boldsymbol{\omega}|\boldsymbol{s})\boldsymbol{R}(\boldsymbol{s},\boldsymbol{\omega})\right]. \tag{9}$$

The structure of the actor network is shown in Figure 1.

3.3 Critic network of RLCC

The update of the critic network is discussed in this subsection. The weighted cross-entropy loss can be utilized for the critic network [36]. The output of the critic network $\boldsymbol{f_\phi}$ can be expressed as

$$\boldsymbol{o}_t = \boldsymbol{f_\phi}(\boldsymbol{s}_t). \tag{10}$$

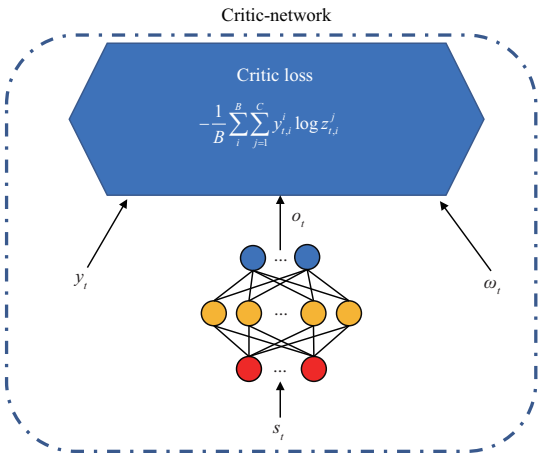

Figure 2 (Color online) Structure and loss function of the critic network.

Figure 3 (Color online) Structure of the reward.

The novel weighted cross-entropy loss is used to minimize the deviation of the predicted label from the true label, and it can be expressed as

$$\boldsymbol{z}_t(\boldsymbol{\omega}_t, \boldsymbol{o}_t) = \begin{bmatrix} \boldsymbol{z}_t{}^1 \\ \boldsymbol{z}_t{}^2 \\ \vdots \\ \boldsymbol{z}_t{}^C \end{bmatrix} = \frac{1}{\sum_j^C \boldsymbol{\omega}_t^j \exp(\boldsymbol{o}_t^j)} \begin{bmatrix} \boldsymbol{\omega}_t{}^1 \exp(\boldsymbol{o}_t{}^1) \\ \boldsymbol{\omega}_t{}^2 \exp(\boldsymbol{o}_t{}^2) \\ \vdots \\ \boldsymbol{\omega}_t{}^C \exp(\boldsymbol{o}_t{}^C) \end{bmatrix}, \tag{11}$$

$$\ell_t(\boldsymbol{y}_t, \boldsymbol{z}_t) = \frac{1}{B}\sum_{i}^{B} \ell_t^i = -\frac{1}{B}\sum_{i}^{B}\sum_{j=1}^{C} y_{t,i}^j \log z_{t,i}^j, \tag{12}$$

where C is the number of classes, B is the batch size, $\boldsymbol{z}_t$ contains the sample weights $\boldsymbol{\omega}_t$ and is related to the output $\boldsymbol{o}_t$, and $\boldsymbol{z}_t^1 = [z_{t,1}^1, \ldots, z_{t,B}^1]$. The structure of the critic network is shown in Figure 2.

3.4 Reward function for RLCC

The minority-class samples are difficult to be identified correctly in the imbalanced dataset. To better recognize the minority-class samples, RLCC should be highly sensitive to the minority-class. A large reward or punishment is returned to the agent when it meets a minority sample. The reward function is defined as follows:

$$\boldsymbol{R}(\boldsymbol{s}_t, \boldsymbol{\omega}_t) = \begin{cases} \boldsymbol{\rho}_t, & \text{Correct classify,} \\ \boldsymbol{\rho}_t \boldsymbol{y}_t \boldsymbol{z}_t, & \text{Misclassify,} \end{cases} \tag{13}$$

where $\boldsymbol{\rho}_t = [\rho_{t,1}, \ldots, \rho_{t,B}]$ is the vector, which is composed of the imbalance ratio for each sample. The imbalance ratio (IR) $\rho_i = N_{\max}/N_k$, which is the sample size of the majority class divided by the sample size of the class k, to which sample i belongs. $\boldsymbol{s}_t$ is the state t. The sample set in batch t is the state $\boldsymbol{s}_t$. The batch samples $\boldsymbol{x}_t$ form the training process batch, which is obtained by dividing the whole training samples. $\boldsymbol{\omega}_t$ is the sample weight of batch t. The value of the reward function is the reward of the agent for learning the sample weights. For an imbalanced dataset, the reward of the minority class is higher than that of the majority class. ρ is a trade-off parameter to adjust the importance of different classes. The structure of the reward is shown in Figure 3. The designed reward function is straightforward but has some limitations. (1) Owing to the sensitivity to the outlier, the performance may be unstable. (2) If there is a class imbalance, then the reward function designed is extremely suitable. Moreover, if there is an imbalance between the samples within the class, the effectiveness of the designed function is slightly reduced. (3) Under the condition of a small number of samples, the designed reward may be restricted by the number of samples. The pseudo-code of the proposed method is summarized as Algorithm 1. The structure diagram of RLCC is drawn under the above designed framework in Figure 4.

Algorithm 1 RLCC

Require: The critic network $\boldsymbol{f_\phi}$, actor network $\boldsymbol{\pi_\theta}$, training dataset D_{trn}, the length of the episode M;
Initialize the critic network $\boldsymbol{f_\phi}$ and actor network $\boldsymbol{\pi_\theta}$ with random parameters ϕ and θ;
Standardize training dataset D_{trn};
for (epi = 0; stopping condition; epi = epi + 1) **do**
  Shuffle training dataset D_{trn} to obtain $\boldsymbol{s}^{\text{epi}} = \{s_1, s_2, \ldots, s_M\}^{\text{epi}}$;
  if epi == 0 **then**
    Set the initial sample weights $\boldsymbol{\omega}^{\text{epi}} = \{\omega_1, \omega_2, \ldots, \omega_M\}^0$ to 1;
  else
    Obtain the $\omega^{\text{epi}} = \{\omega_1, \omega_2, \ldots, \omega_M\}^{\text{epi}}$ by actor network $\boldsymbol{\pi_\theta}(\boldsymbol{\omega}^{\text{epi}}|\boldsymbol{s}^{\text{epi}})$;
  end if
  Obtain the episode $\{(s_1, \omega_1), (s_2, \omega_2), \ldots, (s_M, \omega_M)\}^{\text{epi}}$
  for each $(s_t, \omega_t) \in$ episode **do**
    Update the parameters ϕ of critic network $\boldsymbol{f_\phi}$ following (12);
  end for
  Calculate the reward using (13);
  Obtain the episode $\{(s_1, \omega_1, r_1), (s_2, \omega_2, r_2), \ldots, (s_M, \omega_M, r_M)\}^{\text{epi}}$;
  for each $(s_t, \omega_t, r_t) \in$ episode **do**
    Update the parameters θ of actor network $\boldsymbol{\pi_\theta}$ following (9);
  end for
  Obtain the actor network $\boldsymbol{\pi_\theta}$;
end for
Ensure: $\boldsymbol{f_\phi}, \boldsymbol{\pi_\theta}$;

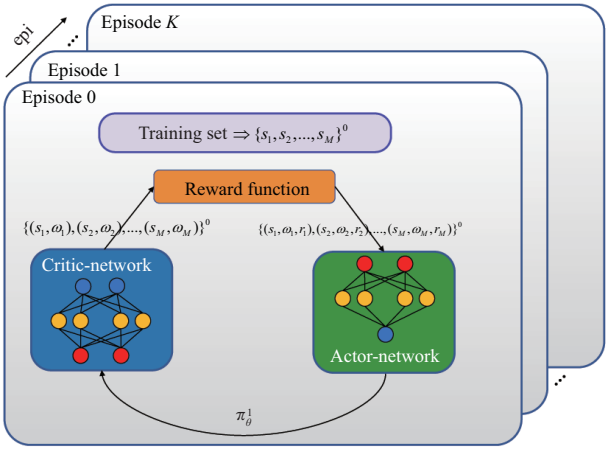

Figure 4 (Color online) Structure and learning process of RLCC.

Table 1 Multiclass confusion matrix

Real	Predicted				total(n_{i+})
	1	2	$\cdots$	k	
1	n_{11}	n_{12}	$\cdots$	n_{1k}	n_{1+}
2	n_{21}	n_{22}	$\cdots$	n_{2k}	n_{2+}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
k	n_{k1}	n_{k2}	$\cdots$	n_{kk}	n_{k+}
total(n_{+j})	n_{+1}	n_{+2}	$\cdots$	n_{+k}	N

4 Case studies

In this section, the effectiveness and practicability of the proposed method are verified through its application to two case studies: the wind turbine freezing failure (WTFF) forecast dataset and the Tennessee Eastman (TE) process dataset.

4.1 Evaluation metrics

In the case studies, evaluation metrics are introduced to evaluate the performance of the proposed method. The confusion matrix contains real classes displayed in columns and predicted classes in rows. In Table 1, the diagonal elements are the right ones, and the elements out of the diagonal are the wrong ones [37].

Table 2 Information of WTFF dataset

Original dataset (feature dimension: 26)			After preprocessing (feature dimension: 2650)		
State	Freeze	Normal	State	Freeze	Normal
#15	23888	344463	Train	1382	17711
#21	10692	168853	Test	143	2503
Label	0	1	Label	0	1

In Table 1, N is the total number of samples. $P_j = n_{jj}/n_{+j}$ and $R_i = n_{ii}/n_{i+}$ are the precision and recall by each class. For multiclass imbalanced classification, only precision and recall are no longer proper measures because the minority classes have very little impact on the accuracy compared with the majority classes [38]. In this manner, balanced accuracy (bAcc) [39], Macro-F_1 [40], and G-mean [41] are used as evaluation metrics.

4.2 Comparisons method

For reasonable comparison, the classification performance of the proposed RLCC is compared with those of different types of cost-sensitive methods: MLP, CoSen-MLP, AdaCost-MLP, CSMLP, and CSBNN. These methods are described as follows.

- MLP: a deep learning classifier that trains the fully connected (FC) neural network using the normal cross-entropy loss function without any optimization technique for multiclass imbalanced fault classification.
- CoSen-MLP: an algorithm-level method that assigns greater misclassification cost to minority classes and smaller cost to majority classes in the loss function of the MLP classifier [32].
- AdaCost-MLP: a misclassification cost-sensitive boosting method that uses the misclassification costs to update the training distribution in successive boosting rounds [17].
- CSMLP: a cost-sensitive deep learning method that uses a single cost parameter to differentiate misclassification errors in the MLP classifier [19].
- CSBNN: a cost-sensitive boosting neural network that incorporates the weight-updating rule of the boosting procedure to associate the samples with misclassification costs [20].

4.3 Case 1: WTFF forecast dataset

The wind turbine manufacturer provided the WTFF dataset[1)] for the prognostic and health management (PHM) competition held by the Chinese government (Ministry of Industry and Information Technology, MIIT). The dataset is generated by the SCADA of the wind power system. The dataset contains 28-dimensional continuous time series, including working conditions, environmental parameters, and state parameters. The two wind turbines (#15 and #21) are marked with the normal and frozen durations (start and end times). The unlabeled data in the training dataset are invalid and are not used for training in the case study. In this case study, the 9000th–149999th timestamp samples of #21 wind turbine are selected as a test set, while the rest are combined with the data of #15 wind turbine as the training dataset. The timestamp interval in the original data is 5–10 s. For wind turbine freezing, the observed value does not change much in a few seconds, and the difference between the samples is small. The time series are segmented into time windows by the sliding window mechanism [42]. The length of the time window is determined as 106, and the sliding step size is 20. The specific information of the WTFF dataset is shown in Table 2. The experimental results were obtained through 10-times 5-fold stratified cross-validation [43]. Table 3 shows all parameters of the RLCC method in the WTFF dataset. Table 4 shows the classification results of different methods in the WTFF dataset in terms of the bAcc, Macro-F_1, and G-mean. The mean of the evaluation metrics represents the effectiveness of the methods. The standard variance of the evaluation metrics shows the stability of the methods. The bold numbers in each row represent the best results.

As shown in Table 4, MLP achieves poor classification performance with low evaluation metric values. Compared with MLP, different cost-sensitive methods for the imbalanced classification problem alleviate the impact of imbalance problems to a certain extent and exhibit higher classification performance than MLP. In this case study, the proposed RLCC exhibits better classification performance than the other methods. RLCC, as a method of updating sample weights in cost-sensitive learning, effectively learns

1) 2017 Industrial Big Data Innovation Competition. http://www. industrial-bigdata.com/. Accessed on June 18 2021.

Table 3 Architecture of the RLCC in WTFF dataset

RLCC architecture	
Actor-network	Critic-network
FC(2650, 56) ReLU	FC(2650, 56) ReLU
FC(56, 2) ReLU	FC(56, 56) ReLU
FC(2, 1) Sigmoid	FC(56, 2) Softmax
Hyperparameter of RLCC	
Learning rate: 0.001, batch size: 128	
iteration: 200, epochs: 100, optimizer: Adam	

Table 4 Classification results of different methods in the WTFF dataset (mean±standard variance)

Metric	MLP	CoSen-MLP	AdaCost-MLP	CSMLP	CSBNN	RLCC
bAcc	0.712±0.056	0.735±0.194	0.845±0.189	0.859±0.113	0.855±0.198	**0.872±0.091**
Macro-F_1	0.6186±0.056	0.624±0.0967	0.740±0.099	0.704±0.173	0.738±0.109	**0.816±0.092**
G-mean	0.668±0.076	0.724±0.201	0.741±0.188	0.740±0.105	0.745±0.191	**0.763±0.031**

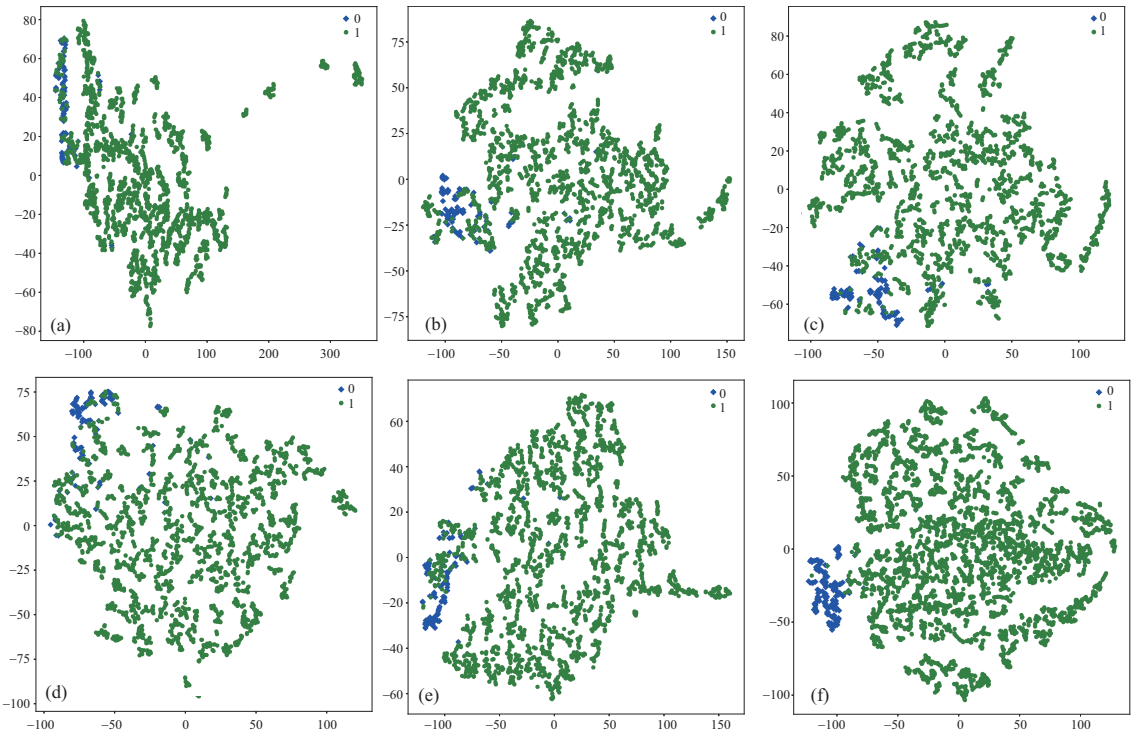

Figure 5 (Color online) 2-D projection scatters of the last layer by t-SNE in the WTFF dataset. 0: Freeze samples; 1: Normal samples. (a) MLP; (b) CoSen-MLP; (c) AdaCost-MLP; (d) CSMLP; (e) CSBNN; (f) RLCC.

the weights of samples in an imbalanced distribution. The sample weights learned by the actor network enable the critic network to classify the minority samples with high sample weights.

The 2-D projection scatters of the last layer output of all methods in the test dataset (Normal and Freeze) are depicted in Figure 5 according to t-SNE [44], where the blue color corresponds to Freeze samples and the green color corresponds to Normal samples. As shown in Figure 5(a), MLP exhibits a poor ability to classify samples and has the overlapping problem. The Freeze samples are basically covered by the Normal samples. Moreover, the distribution of Normal samples is relatively scattered, which is typically the dispersion problem. Thus, almost all of the fault samples are difficult to identify. As shown in Figure 5(b), compared with naive MLP, the overlapping problem of CoSen-MLP is solved to a certain extent, and the number of overlapping samples is reduced. The dispersion problem of CoSen-MLP is also resolved because, different from MLP, CoSen-MLP simply assigns the inverse frequency of the sample size of classes as the weight of different samples. As shown in Figures 5(c)–(e), compared with CoSen-MLP, AdaCost-MLP, CSMLP, and CSBNN are effective for the imbalanced classification. The

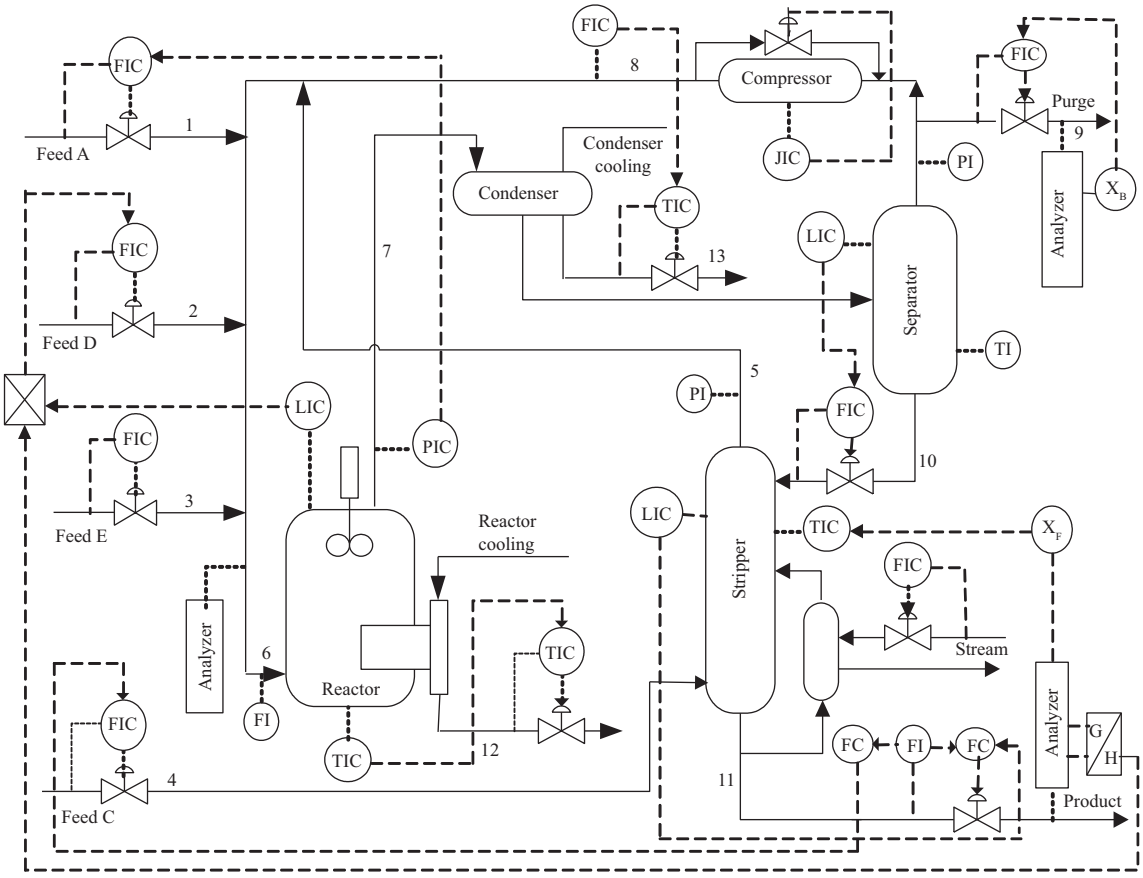

Figure 6 Flowchart of the TE process.

Table 5 Architecture of the RLCC in TE process dataset

RLCC architecture	
Actor-network	Critic-network
FC(32, 64) ReLU	FC(32, 62) ReLU
FC(64, 2) ReLU	FC(62, 16) ReLU
FC(2, 1) Sigmoid	FC(16, number of classes) Softmax
Hyperparameter of RLCC	
Learning rate: 0.0001, batch size: 16	
iteration: 100, epochs: 50, optimizer: Simple Adam	

reason is that those methods use an iterative weight update mechanism to learn more accurate sample weights and obtain great classification performance. However, the figures show that the overlapping problem is still pronounced, and it is difficult to classify some samples. As for RLCC, Figure 5(f) shows that the decision boundaries between classes are more pronounced and that the overlapping samples are significantly reduced. Moreover, the Normal samples are also more evenly distributed. The alternating iterative actor-critic for learning sample weights is an effective cost-sensitive strategy. The proposed RLCC can solve the overlapping problem and the dispersion problem in the WTFF dataset.

4.4 Case 2: TE process dataset

An industrial benchmark of the TE process is used to evaluate the proposed RLCC. This industrial process was first introduced by Downs and Vogel [45], and has been widely used for testing and evaluating various process monitoring algorithms and control strategies. As shown in Figure 6, five operation units exist in the TE process benchmark. Twenty-eight faults are available for simulation in this process [46]. In this case study, binary-class and multiclass experiments are conducted to verify the feasibility and effectiveness of the proposed method. For the binary-class experiment, the number of Normal samples for training is

Table 6 Binary-class results in the TE process dataset in terms of G-mean

Dataset	IR	MLP	CoSen-MLP	AdaCost-MLP	CSMLP	CSBNN	RLCC
Fault5-Normal	1	0.7581	**0.7600**	0.7591	0.7571	0.7532	0.7588
	2	0.5731	0.5823	0.5971	0.6291	0.6847	**0.7551**
	10	0.3575	0.3365	0.3495	0.4105	0.6438	**0.7042**
	20	0.3550	0.3513	0.3548	0.3304	0.3803	**0.7057**
	100	0.2227	0.2586	0.2430	0.2643	0.3995	**0.6995**
Fault12-Normal	1	**0.6549**	0.6499	0.6420	0.6410	0.6421	0.6407
	2	0.4976	0.5217	**0.5437**	0.5300	0.5099	0.5407
	10	0.3739	0.4646	0.4612	0.4300	0.4146	**0.5106**
	20	0.3564	0.4305	0.4413	0.4090	0.3881	**0.4933**
	100	0.0844	0.0719	0.0671	0.0924	0.0316	**0.1632**
Fault15-Normal	1	0.4611	**0.4635**	0.4609	0.4608	0.4623	0.4607
	2	0.3755	0.3675	0.3904	0.3563	0.3856	**0.4201**
	10	0.2375	0.2305	0.2303	0.2356	0.2562	**0.2990**
	20	0.2020	0.2054	0.1988	0.1994	0.2001	**0.2447**
	100	0.0941	0.1023	0.0960	0.1078	0.0850	**0.1784**

Table 7 Multiclass results of different methods in the TE process dataset (mean±standard variance)

Metric	MLP	CoSen-MLP	AdaCost-MLP	CSMLP	CSBNN	RLCC
bAcc	0.7312±0.123	0.7625±0.102	0.775±0.085	0.7938±0.043	0.8063±0.098	**0.8188±0.013**
Macro-F_1	0.7071±0.154	0.7380±0.104	0.7610±0.073	0.7752±0.053	0.8036±0.112	**0.8184±0.021**
G-mean	0.6348±0.201	0.6708±0.124	0.7154±0.142	0.7306±0.103	0.7772±0.103	**0.7978±0.018**

400. The number of samples used for training in Fault 5, Fault 12, and Fault 15 varies with the imbalance ratio. For the multiclass experiment, Normal, Fault 2, Fault 6, and Fault 12 are selected to construct an imbalanced dataset. The number of samples used for training in Normal, Fault 2, Fault 6, and Fault 12 are 720, 356, 222, and 173, respectively. The number of samples used for testing in Normal, Fault 2, Fault 6, and Fault 12 is 80. Table 5 shows all parameters of the RLCC method in the TE process dataset. Tables 6 and 7 show the results of different methods for the binary-class and multiclass experiments, respectively. The bold numbers in each row represent the best results.

For the binary-class experiment, five imbalance ratios of three fault datasets are investigated to evaluate the effectiveness of each method. As shown in Table 6, when the imbalance ratio is 1, that is, when the data are balanced, the performance of each method is not much different. With a gradual decrease in the imbalance rate, the G-mean of all methods sharply decreases, owing to the impact of the imbalance problem. Figure 7 shows the detailed experimental results of each method through the line chart. In the line chart, which is plotted by the imbalance ratio on the horizontal axis and the G-mean on the vertical, each point represents the G-mean value of one method in one dataset. From the table and the line chart, we can see that the effect decline is generally serious on MLP. Different cost-sensitive methods alleviate the impact of the imbalance problems to a certain extent. As shown in Figure 7, the RLCC line is higher than other lines in most datasets, which reflects that for most datasets, the proposed RLCC is better than other methods under different imbalance ratios. Furthermore, the decline effect of RLCC is not as significant as those of other methods when the imbalance ratio increases. RLCC basically outperforms the other five methods under extreme imbalance situations. Therefore, the higher the imbalance ratio is, the more significant the advantages of RLCC are.

For the multiclass experiment, Figure 8 represents the reward over training episodes. RLCC shows faster optimization in the training episodes. RLCC reaches the saturation region at about 200 episodes. This shows that RLCC can automatically optimize the performance of the imbalanced classifier. As shown in Table 7, the proposed RLCC outperforms the other methods. MLP achieves poor classification performance with low evaluation metric values. Different cost-sensitive methods are applied to solve the imbalanced classification problems, and they yield higher classification performance. The updating sample weight mechanism in cost-sensitive learning is effective for the imbalanced classification problem. The actor network can more accurately learn sample weights according to the specific sample classification results. The 2-D projection scatters of the last layer output of all methods for the test dataset (Normal, Fault 2, Fault 6, and Fault 12) are depicted in Figure 9 according to t-SNE [44]. As shown in Figure 9(a),

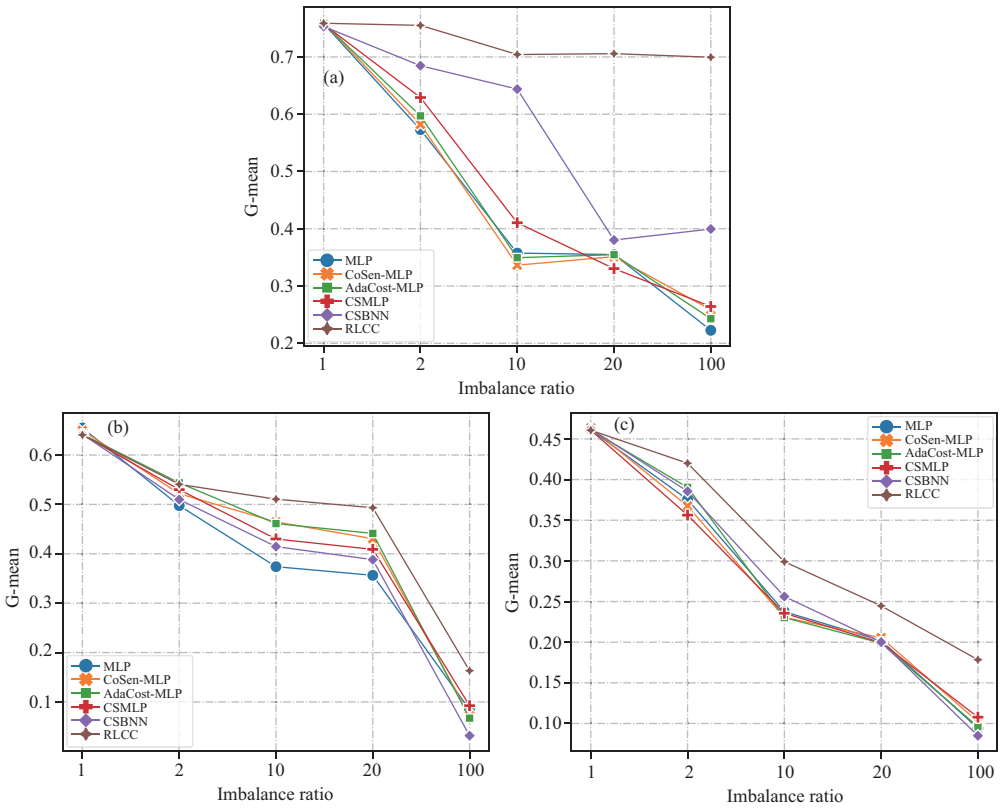

Figure 7 (Color online) Results of comparison methods with respect to G-mean on different designed TE datasets. (a) Fault5-Normal; (b) Fault12-Normal; (c) Fault15-Normal.

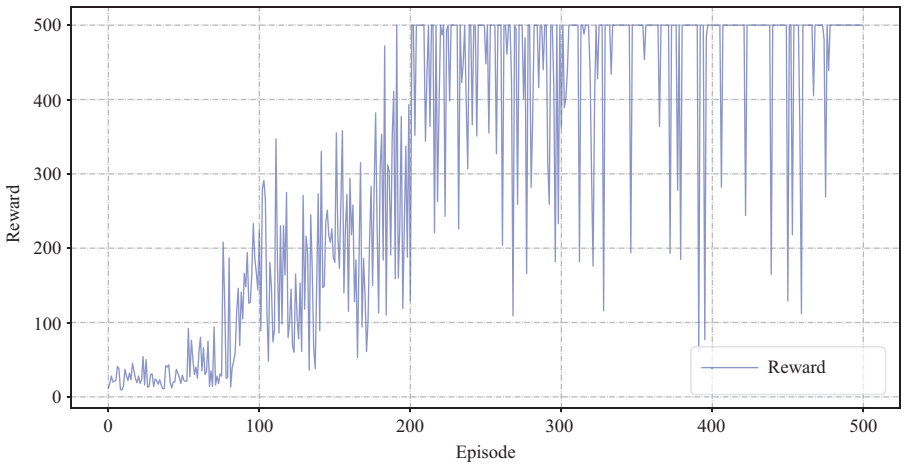

Figure 8 (Color online) Reward of RLCC for the multiclass experiment in the TE process dataset.

MLP has a poor ability to classify samples. The overlapping problem still exists in Figure 9(a), particularly among Normal, Fault 2, and Fault 12. Thus, Normal, Fault 2, and Fault 12 are difficult to identify. As shown in Figures 9(b)–(e), the comparison methods outperform naive MLP to a certain extent. The iterative weight update mechanism endows the classifier with great classification performance. As shown in Figure 9(f), the decision boundaries among Normal, Fault 2, and Fault 12 are more significant, and the overlapping samples are significantly reduced. Thus, the proposed RLCC can solve the overlapping problem and the dispersion problem in the TE process dataset.

In summary, RLCC is a good technique for dealing with imbalanced classification problems without changing the data distribution. As expected, RLCC can alleviate the impact of imbalance distribution for fault classification. RLCC gives a higher reward for minority samples, which increases the probability

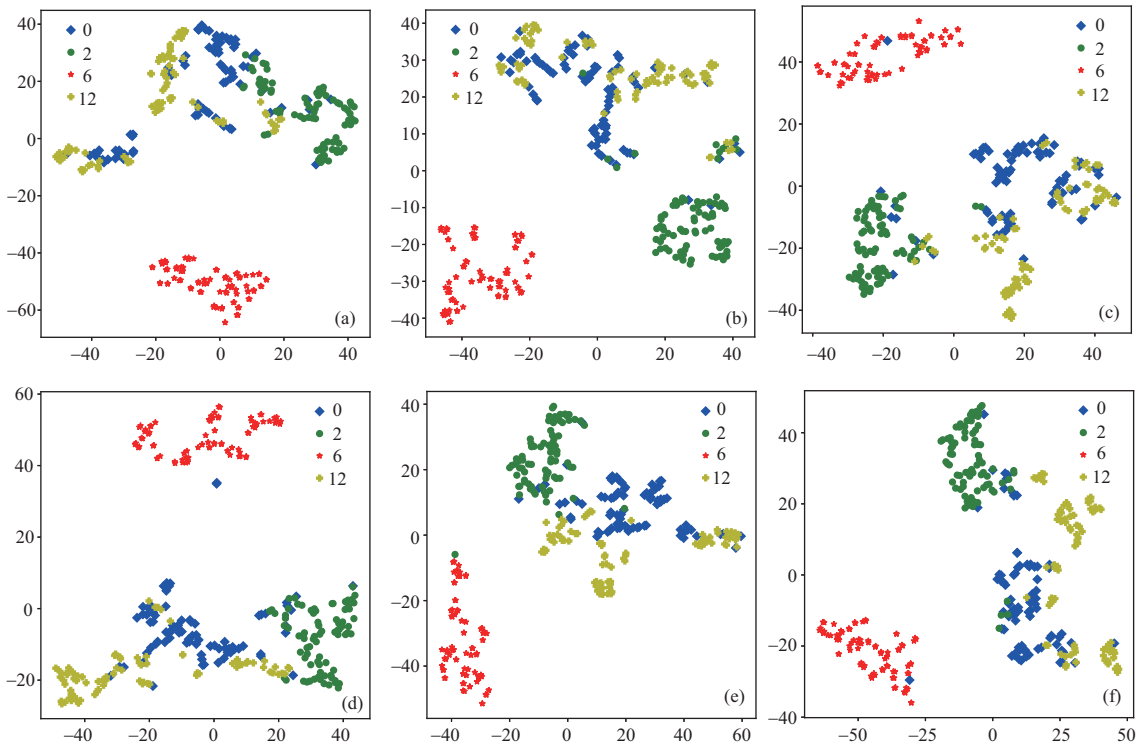

Figure 9 (Color online) 2-D projection scatters of the last layer by t-SNE in the TE process dataset. 0: Normal; 2: Fault 2; 6: Fault 6; 12: Fault 12. (a) MLP; (b) CoSen-MLP; (c) AdaCost-MLP; (d) CSMLP; (e) CSBNN; (f) RLCC.

that minority samples are correctly classified. In addition, RLCC can dynamically yield the cost matrix to assign a higher weight for important samples. The result reflects that the proposed RLCC can more effectively handle the imbalanced fault classification problem.

5 Conclusion

In this paper, a novel RLCC is proposed to handle the imbalanced fault classification. Compared with the basic actor-critic mechanism of RL, we modified the loss of the actor network via policy gradient and introduced a novel cost matrix into the critic network. Moreover, we designed appropriate rewards to guide the novel actor-critic mechanism to learn the optimal cost matrix for the imbalanced fault classification problem. The usefulness and advantages of the proposed RLCC were validated through two case studies. The experimental results showed that RLCC has the following remarkable advantages: (1) A novel weighted cross-entropy function that combines the output of the MLP classifier with the cost matrix is effective for solving the imbalanced fault classification problem. (2) The REINFORCE loss of RL can be used to learn the sample weights in combination with the classification results. (3) The alternating iterative approach of the actor-critic mechanism can be used to learn sample weights and optimize the classification performance. Finally, the application results demonstrate the superior abilities of the proposed method compared with the other cost-sensitive methods.

Acknowledgements This work was supported in part by National Natural Science Foundation of China (Grant Nos. 62003301, 61833014) and Natural Science Foundation of Zhejiang Province (Grant No. LQ21F030018).

References

1 Kano M, Nakagawa Y. Data-based process monitoring, process control, and quality improvement: recent developments and applications in steel industry. Comput Chem Eng, 2008, 32: 12–24
2 Ge Z, Song Z, Ding S X, et al. Data mining and analytics in the process industry: the role of machine learning. IEEE Access, 2017, 5: 20590–20616
3 Zhu Z R, Chai Y, Yang Z M. A novel kind of sufficient conditions for safety judgement based on control barrier function. Sci China Inf Sci, 2021, 64: 199205
4 Zhang X, Wei C, Song Z. Fast locally weighted PLS modeling for large-scale industrial processes. Ind Eng Chem Res, 2020, 59: 20779–20786
5 Khatibisepehr S, Huang B, Khare S. Design of inferential sensors in the process industry: a review of Bayesian methods. J Process Control, 2013, 23: 1575–1596

6 Zhou D H, Qin L G, He X, et al. Distributed sensor fault diagnosis for a formation system with unknown constant time delays. Sci China Inf Sci, 2018, 61: 112205
7 Huang D Q, Fu Y Z, Qin N, et al. Fault diagnosis of high-speed train bogie based on LSTM neural network. Sci China Inf Sci, 2021, 64: 119203
8 Chen G, Liu Y, Ge Z. K-means Bayes algorithm for imbalanced fault classification and big data application. J Process Control, 2019, 81: 54–64
9 Kubat M, Holte R C, Matwin S. Machine learning for the detection of oil spills in satellite radar images. Machine Learn, 1998, 30: 195–215
10 Krawczyk B. Learning from imbalanced data: open challenges and future directions. Prog Artif Intell, 2016, 5: 221–232
11 Wang X Y, Liu B, Cao S Y, et al. Important sampling based active learning for imbalance classification. Sci China Inf Sci, 2020, 63: 182104
12 Jiang X, Ge Z. Data augmentation classifier for imbalanced fault classification. IEEE Trans Automat Sci Eng, 2020, 18: 1206–1217
13 Fan S, Zhang X, Song Z. Imbalanced sample selection with deep reinforcement learning for fault diagnosis. IEEE Trans Ind Inf, 2021, 18: 2518–2527
14 Yue G, Wei P, Liu Y, et al. Automated endoscopic image classification via deep neural network with class imbalance loss. IEEE Trans Instrum Meas, 2023, 72: 1–11
15 Santos M S, Abreu P H, Japkowicz N, et al. On the joint-effect of class imbalance and overlap: a critical review. Artif Intell Rev, 2022, 55: 6207–6275
16 Hoskins J C, Himmelblau D M. Artificial neural network models of knowledge representation in chemical engineering. Comput Chem Eng, 1988, 12: 881–890
17 Hastie T, Rosset S, Zhu J, et al. Multiclass AdaBoost. Stat Its Interface, 2009, 2: 349–360
18 Krempl G, Kottke D, Lemaire V. Optimised probabilistic active learning (OPAL). Mach Learn, 2015, 100: 449–476
19 Castro C L, Braga A P. Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data. IEEE Trans Neural Netw Learn Syst, 2013, 24: 888–899
20 Zheng J. Cost-sensitive boosting neural networks for software defect prediction. Expert Syst Appl, 2010, 37: 4537–4543
21 Zhang C, Tan K C, Ren R. Training cost-sensitive deep belief networks on imbalance data problems. In: Proceedings of International Joint Conference on Neural Networks (IJCNN), 2016. 4362–4367
22 Sun Y, Kamel M S, Wang Y. Boosting for learning multiple classes with imbalanced class distribution. In: Proceedings of the 6th International Conference on Data Mining (ICDM'06), 2006. 592–602
23 Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn, 1992, 8: 229–256
24 Chawla N V, Bowyer K W, Hall L O, et al. SMOTE: synthetic minority over-sampling technique. J Artif Intell Res, 2002, 16: 321–357
25 Ling C X, Yang Q, Wang J, et al. Decision trees with minimal costs. In: Proceedings of the 21st International Conference on Machine Learning, 2004. 69
26 Zadrozny B, Elkan C. Learning and making decisions when costs and probabilities are both unknown. In: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001. 204–213
27 Zadrozny B, Langford J, Abe N. Cost-sensitive learning by cost-proportionate example weighting. In: Proceedings of the 3rd IEEE International Conference on Data Mining, 2003. 435–442
28 Sun Y, Kamel M S, Wong A K C, et al. Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition, 2007, 40: 3358–3378
29 Ting K M. A comparative study of cost-sensitive boosting algorithms. In: Proceedings of the 17th International Conference on Machine Learning, 2000. 983–990
30 Shawe-Taylor G K J, Karakoulas G. Optimizing classiffiers for imbalanced training sets. In: Proceedings of Advances in Neural Information Processing Systems, 1999. 11: 253
31 Viola P, Jones M. Fast and robust classiffication using asymmetric adaboost and a detector cascade. In: Proceedings of Advances in Neural Information Processing System, 2001. 14
32 Zhou Z-H, Liu X-Y. Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Trans Knowl Data Eng, 2005, 18: 63–77
33 Sutton R S, McAllester D A, Singh S P, et al. Policy gradient methods for reinforcement learning with function approximation. In: Proceedings of Advances in Neural Information Processing Systems, 2000. 1057–1063
34 Konda V R, Tsitsiklis J N. Actor-critic algorithms. In: Proceedings of Advances in Neural Information Processing Systems, 2000. 1008–1014
35 Bhatnagar S, Sutton R S, Ghavamzadeh M, et al. Natural actor-critic algorithms. Automatica, 2009, 45: 2471–2482
36 Chen J, Tsai C-A, Chen J J, et al. Decision threshold adjustment in class prediction. SAR QSAR Environ Res, 2006, 17: 337–352
37 Alejo R, Sotoca J M, Casañ G A. An empirical study for the multiclass imbalance problem with neural networks. In: Proceedings of Iberoamerican Congress on Pattern Recognition, 2008. 479–486
38 Joshi M V, Kumar V, Agarwal R C. Evaluating boosting algorithms to classify rare classes: comparison and improvements. In: Proceedings of IEEE International Conference on Data Mining, 2001. 257–264
39 Brodersen K H, Ong C S, Stephan K E, et al. The balanced accuracy and its posterior distribution. In: Proceedings of the 20th International Conference on Pattern Recognition, 2010. 3121–3124
40 Opitz J, Burst S. Macro F1 and Macro F1. 2019. ArXiv:1911.03347
41 Alejo R, García V, Sotoca J M, et al. Improving the performance of the RBF neural networks trained with imbalanced samples. In: Proceedings of International Work Conference on Artificial Neural Networks, 2007. 162–169
42 Wu Z, Lin W, Ji Y. An integrated ensemble learning model for imbalanced fault diagnostics and prognostics. IEEE Access, 2018, 6: 8394–8402
43 Moreno-Torres J G, Sáez J A, Herrera F. Study on the impact of partition-induced dataset shift on k-fold cross-validation. IEEE Tr ans Neural Netw Learn Syst, 2012, 23: 1304–1312
44 van Maaten L D. Accelerating t-SNE using tree-based algorithms. J Machine Learn Res, 2014, 15: 3221–3245
45 Lyman P R, Georgakis C. Plant-wide control of the Tennessee Eastman problem. Comput Chem Eng, 1995, 19: 321–331
46 Yin S, Ding S X, Haghani A, et al. A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process. J Process Control, 2012, 22: 1567–1581