Markov Decision Process for imbalanced classification

1st Chunyu Xuan
School of Automation Science and Engineering
Xi'an Jiaotong University
Xi'an, China
diary180729@stu.xjtu.edu.cn

2nd Jing Yang
School of Automation Science and Engineering
Xi'an Jiaotong University
Xi'an, China
jasmine1976@xjtu.edu.cn

3rd Zhou Jiang
School of Software Engineering
Xi'an Jiaotong University
Xi'an, China
2560401450@qq.com

4th Dong Zhang
School of Automation Science and Engineering
Xi'an Jiaotong University
Xi'an, China
dongzhangcv@qq.com

***Abstract*—In imbalanced classification problems, the samples of different classes are so imbalanced that the model cannot effectively identify the minority class samples. To solve this problem, this article proposes a new algorithm which is named TargetValue algorithm. It constructs a Markov Decision Process according to the imbalanced data set. And the reward function is carefully designed. Since the constructed Markov Decision Process has simple dynamics, the action value function can be directly calculated by derivation and handed over to the neural network for fitting. The neural network classifies unknown samples by comparing the values of different action. This article analyzes the reasons for the effectiveness of the algorithm from two perspectives: the reward function influence both the target value and the gradient of the long-term expected return. And binary classification and multi-classification experiments on multiple imbalanced data sets are conducted to verify the effectiveness of the algorithm.**

***Index Terms*—Markov Decision Process, action value function, imbalanced data classification, reward function**

I. Introduction

In machine learning, especially in supervised learning, the problem of sample imbalance is widespread and common [1]–[3]. Sample imbalance means there is a serious imbalance in the number of different kinds of samples. When training a neural network with an imbalanced data set, the trained model tends to judge unknown samples as majority class samples. In this case, even if the overall classification accuracy is high, the model does not have much practical value. Because the model does not have enough ability to identify minority class samples. However, this ability is very important in many cases. For example, in medicine, the samples of diseases are relatively rare, but the identification of these samples is of great significance to the diagnosis of the disease; in security, the abnormal data received by attacks and deceptions are relatively rare, but the identification of these small amounts of data is of great significance, for it is very important for safety protection.

For the imbalanced data classification, the traditional solutions [4] still have some shortcomings. Re-sampling methods change the distribution of the data, which is easy to cause the loss of data features [5], or the mechanical repetition of some features [6], so that the model cannot objectively learn the original characteristics of the data. Algorithm level methods also have their drawbacks. For example, cost-sensitive [7] learning have high computational and training costs. And in the face of different data, these methods are not universal.

This article proposes the TargetValue(TV) algorithm, which is a method to solve the imbalanced data classification problem from the algorithm level. It can be implemented simply, and is universal on different data sets. The TV algorithm transforms the imbalanced data classification problem into a Markov Decision Process(MDP) [8]. In this MDP, the environment is an imbalanced data set, the state faced by the agent is a single sample, and the action made by the agent is to determine which class the sample belongs to. Reward is an artificially set parameter. When the agent judges correctly, it returns a positive reward to the agent, and when the agent judges incorrectly, it returns a negative reward to punish it. Different reward and punishment values are set for different classes to guide the agent to pay attention to the minority samples. Due to the simple dynamics of the constructed MDP, complex algorithms are not needed to solve the optimal policy. The action value function is directly calculated, and a neural network is used to fit it. Using the inherent generalization performance of neural networks, when an unknown sample is input, the classification can be completed by comparing the magnitude of the output action value. In this article, experiments on imbalanced binary classification and imbalanced multi-classification are carried out on multiple data sets. The TV algorithm has a significant improvement over the baseline, and is on par with DQNimb [9], which has advantages over several traditional algorithms.

To summarize, the main contributions of this paper are as follows:

1) The TV algorithm is proposed. Unlike using complex methods such as reinforcement learning algorithms to solve the MDP, it directly calculates the action value function through theoretical derivation, eliminating the iterative process.
2) The MDP is constructed through the data set, and a new

This work was supported by the National Natural Science Foundation of China under Grant No. 62073257, 62141223, and the Key Research and Development Program of Shaanxi Province of China under Grant No. 2022GY-076.

reward function design method is proposed.
3) The theoretical reasons for the effectiveness of the TV algorithm are analyzed from two perspectives.
4) Two-class and multi-class experiments are carried out on one text data set(IMDB) [10] and three image data sets(Cifar10 [11], MNIST [12], Fashion-MNIST [13]). The data sets were resampled to form different imbalance ratios. The DQNimb algorithm is used for comparison. The effectiveness of the TV algorithm to solve the imbalanced classification problem is verified.

The rest of this article is organized as follows: In section II, the related work is introduced. Section III introduces the overall flow and the theoretical analysis of the algorithm. The experimental results for binary and multi-class classification are introduced in section IV. Section V is the conclusion and outlook.

II. Related Work

In some previous works, the method of constructing an MDP and then solving the MDP with reinforcement learning was used to solve the problem of data imbalance [14]–[18]. DiagSelect [19] solves the problem from the data level. It builds an MDP similar to a multi-armed bandits(MAB) problem. The agent selects some data from the data set each time and sends it to the model for training. After the training is completed, the model is tested on the validation set, and the results on the validation set are used as feedback. The agent finally finds the best way to filter the data. As a data-level algorithm, this algorithm can be used in conjunction with different models. DQNimb constructs an MDP based on the degree of imbalance of the data set, and transforms the classification problem into a game. The agent can get rewards for correct classification, and get punishment for wrong classification. The goal of the agent is to get the highest score. And in order to emphasize the importance of the minority class samples, a higher score is set for them. Converting different data sets to MDP is a general procedure, and both algorithms are universal on different data sets. The difference is that the DiagSelect algorithm solves an MDP with an unknown solution. Though it is a simple MAB problem, humans do not know how to select data to achieve better classification results. The optimal policy can only be left to the agent to explore and learn. However, DQNimb is dealing with a completely known model. Once the labels of the samples are known, the optimal policy is to judge each sample correctly. Although humans know the optimal policy, the agent needs to learn this policy through the goal setting and training process of deep reinforcement learning. The TV algorithm adopted in this article is closer to DQNimb. The game rules in DQNimb are adopted, but a new reward function is proposed. The reward function in DQNimb is based on empirical settings and has no theoretical basis. The reward function in this article is also constructed based on experience. In the case of binary classification, the reward function of DQNimb is an approximation of the reward function designed in this article. In the case of multi-classification, the reward function designed in this article can obtain better results. The game environment constructed in this article is simple and known at the same time, so different from DiagSelect and DQNimb, the MDP is not solved by reinforcement learning. On the contrast, the action value function under the optimal policy is directly calculated and given to the neural network for fitting .

III. TargetValue

This section introduces the process of the TV algorithm and the details of each link. In addition, the reason why the algorithm is effective is also analyzed. The first part introduces the overall flowchart of the TV algorithm, which mainly includes two parts: training and testing. The second part presents the details of the MDP construction. The third and the fourth part describes how to calculate the value function and train the neural network. The last part analyzes the algorithm.

A. Overall Flowchart

The overall flowchart of using TV to achieve imbalanced data classification is shown in Figure. 1, which includes two parts: training and testing. These two parts contain a total of five steps.

- Training

Step 1: Construct an MDP based on the train set. The states are different samples in the set, the action is to determine which class a sample belongs to, the reward function is set based on the degree of imbalance, and the dynamics of the MDP is set manually.

Step 2: Calculate the action value function under the optimal policy by derivation. The derivation process is based on the dynamics of the MDP.

Step 3: The mean squared loss function is used to train the neural network model to fit the calculated action value function. The input of the network is a single sample and the output is a one-dimensional vector, each component representing the value of an action.

- Testing

Step 4: For an unknown sample on the test set, feed it into the trained model, and the model will output a vector with each component representing the value of an action.

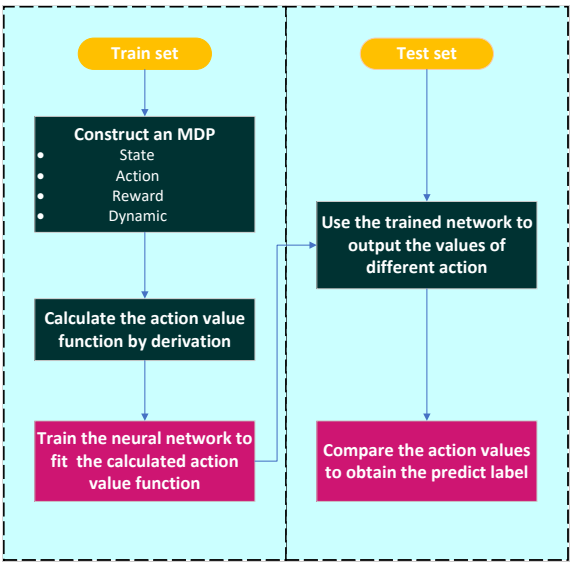

Fig. 1. Flowchart of TargetValue applied to imbalanced data classification

Step 5: Compare the values of these actions and take the action with the greatest value, i.e. judge the sample into which class.

B. Details for the MDP

In this MDP, a sample from the train set is randomly taken out each time, and the agent is asked to determine which class it belongs to. The correct judgment will increase the score, and the wrong judgment will reduce the score. Keep repeating this process. However, in order to emphasize the importance of minority class samples, different scores are set for different classes of samples. And if the agent makes a misjudgment when faced with a minority class sample, the game is over.

The three elements of state, action and reward in the MDP are specifically set as follows.

- State: Given the imbalanced train set $D_t = \{x_k, y_k\}_{k=1}^{N}$, where $N = \sum_{i=0}^{C-1} n_i$ is the number of samples, n_i is the number of samples from class i, and C is the number of classes. The entire train set constitutes the state space of the MDP. During the game, the state s of the agent at each moment corresponds to a specific sample x_k. When the agent completes the judgment, it randomly obtains an other sample as the state s' of the next moment. If the current state is a minority class sample and the agent makes a wrong judgment, the game will be over.
- Action: Faced with the state of each moment, that is, a sample in a training set, the action a that the agent can take is to choose the label for the sample. We directly use the label selected to represent action a. For example, if the agent judges that the sample belongs to class 0, then $a = 0$.
- Reward: function φ maps a sample to its label. It is defined in (1):

$$\varphi(x_k) = y_k \tag{1}$$

  As discussed above, s is a sample in the train set D_t, a is the label chosen by the agent, the reward function is defined as follows:

$$R(s,a) \doteq (-1)^{bool(\varphi(s)=a)+1}(1 - n_{\varphi(s)}/N) \tag{2}$$

  This definition makes the minority class samples more rewarding and punishing, meaning that they are more important for achieving high scores.

C. Calculation of Action Value Function

Based on the particularity of the constructed MDP, this part directly calculates the action value function under the optimal policy.

The action value function can be obtained directly for two reasons. On the one hand, the model of the game is known, that is, the dynamics and the reward function are known. On the other hand, the constructed MDP is somewhere between the contextual bandits problem and the full reinforcement learning problem.

In a contextual bandits problem, the state at the next moment is independent of the action taken at the current moment. While in a full reinforcement learning problem, the action taken affects the state of the agent at the next moment. This makes full reinforcement learning more complicated. Even if the model is completely known, it is difficult to directly solve the action value, and it needs to be approximated by methods such as dynamic programming or sampling.

In the MDP constructed in this section, the action taken also affects the state at the next moment, because misjudging the minority class will lead to the end of the game. But most of the time, the state of the environment jumps randomly, which makes our MDP far simpler than a full reinforcement learning problem.

$q_*(s,a)$ is the value of taking action a in state s under the optimal policy, $q_*(s',a')$ is the value of taking action a' in the state of the next moment s' under the optimal policy, γ is a parameter with a value between 0 and 1. Write the Bellman Optimal Equation in the form of expectation:

$$q_*(s,a) = R(s,a) + \gamma E[\max_{a'} q_*(s',a')|s,a] \tag{3}$$

$E[\max_{a'} q_*(s',a')|s,a]$ can be seen as a function $E(s,a)$. When s belongs to minority class and a is an incorrect judgement, $E(s,a)$ is equal to 0, otherwise $E(s,a)$(abbreviated as E) satisfies (4).

$$E = \sum_{i=1}^{C-1} \frac{n_i}{N}(R(x_{k_i}, y_{k_i}) + \gamma E) \tag{4}$$

where x_{k_i} is a sample selected from class i, and y_{k_i} is the label of x_{k_i}, R is the reward function defined in (2).

simplify (4) and there is:

$$E = \frac{\sum_{i=1}^{C-1} \frac{n_i}{N} R(x_i, y_i)}{(1-\gamma)} \tag{5}$$

Therefore, the action value function under the optimal policy is obtained.

$$q_*(s,a) = R(s,a) + \gamma E(s,a) \tag{6}$$

D. Training of the model

After the action value function is obtained, a neural network is used to fit the action value function. A mean squared loss function is used to train the network. The input of the neural network is the sample in the training set, and the output is a vector of length C with each component represents the value of an action. For the sample x_k, let its output be o_k, and its target value obtained from the action value function is $t_k = (q_*(x_k, a_0), q_*(x_k, a_1), ..., q_*(x_k, a_{C-1}))$. Then the loss function of parameter θ is shown in (7).

$$L(\theta) = \sum (t_k - o_k)^2 \tag{7}$$

The pseudocode of the TV algorithm is shown in Algorithm 1.

E. Analysis of the Method

This part will analyze why TV can performance well in imbalanced classification. Before analyzing the effectiveness, it is necessary to analyze why the algorithm is feasible. Solving the MDP optimizes the ability of an agent to achieve a specific goal in a closed environment, while showing weak generalization performance outside the environment. However, for classification problems, generalization performance is exactly what one is looking for. The reason why a model with generalization performance is obtained is because the neural network itself has generalization performance, and the

test set and the training set have a high similarity as a game environment. For a sample on the test set, the neural network can also receive its input and output the corresponding result.

Models trained on imbalanced data tend to perform poorly on minority classes. Due to the small proportion of the minority class samples in the gradient update, the loss of the minority class samples is difficult to reduce. Therefore, the output of the minority class sample does not adequately fit the target value. In the TV algorithm, this problem still exists. Fig. 2 shows the output of a binary classification model trained on the Cifar10 data set. 200 samples were randomly selected from the test set, of which 100 samples were of the majority class and the other 100 were of the minority class. It can be seen that the output of the majority class sample is relatively closer to the target value, while the output of the minority class sample oscillates within a larger range. However, due to the setting of the reward function, the target value of the minority class is further away from the dividing line. In this way, even if the output oscillates in a large range, as long as the relative magnitude of the two action values does not change, that is, the output points do not cross the dividing line, the classification can still be performed correctly.

The setting of the reward function influence not only the target value, but also the gradient of the long-term expected return. Take the case of binary classification as an example. Let the long-term expected return be U, the proportion of minority class samples to all samples be p, the recall of minority class samples be m, the recall of majority class samples be n, the reward for minority class samples be r_1, and the reward for majority class samples be r_2. According to the definition of expectation:

$$\begin{aligned} U =&(1-p)(n(r_2+\gamma U)+(1-n)(-r_2+\gamma U)) \\ +&\quad p(m(r_1+\gamma U)+(1-m)(-r_1)) \end{aligned} \tag{8}$$

Algorithm 1: TargetValue

Input: The imbalanced data set D_{imb}, the neural network π_θ, the reduction factor γ, the hyperparameter epi

Output: π_θ

split D_{imb} into validation set D_v and training set D_t.
calculate the long-term expected return $E(s,a)$ following(5).
calculate the action value function $q_*(s,a)$ following (6), where γ is used.
use $q(s,a)$ to obtain target value t_k for every x_k in D_t.
for $i = 1, ..., epi$ **do**
    shuffle the data set randomly.
    for *batches in* D_t **do**
        update the parameter θ of the network π_θ according to the loss function (7).
    end
    test on the validation set.
end

Move items to get (9):

$$U = \frac{r_2(1-p)(2n-1)+r_1p(2m-1)}{1-\gamma(1+pm-p)} \tag{9}$$

U, m, n are all functions of the neural network parameter θ. The gradient of U with respect to θ is shown in (10):

$$\frac{\nabla U}{\nabla \theta} = \frac{\partial U}{\partial m}\frac{\nabla m}{\nabla \theta} + \frac{\partial U}{\partial n}\frac{\nabla n}{\nabla \theta} \tag{10}$$

The functions $m(\theta)$ and $n(\theta)$ are only related to the data set and the neural network itself, and have nothing to do with the setting of the reward function. So the setting of r_1 and r_2 will not affect $\frac{\nabla m}{\nabla \theta}$, $\frac{\nabla n}{\nabla \theta}$, only $\frac{\partial U}{\partial m}$, $\frac{\partial U}{\partial n}$, which are shown in (11) and (12).

$$\frac{\partial U}{\partial m} = \frac{r_1[2p(1-\gamma)+p^2\gamma]+r_2(1-p)p(2n-1)\gamma}{[1-\gamma(1+pm-p)]^2} \tag{11}$$

$$\frac{\partial U}{\partial n} = \frac{2r_2(1-p)[1-\gamma(1+pm-p)]}{[1-\gamma(1+pm-p)]^2} \tag{12}$$

It can be seen from (11) and (12) that increasing r_1 can increase the relative size of $\frac{\partial U}{\partial m}$, and increasing r_2 can increase the relative size of $\frac{\partial U}{\partial n}$. When n is relatively small($2n-1<0$), increasing r_2 will also reduce $\frac{\partial U}{\partial m}$. By adjusting $\frac{\partial U}{\partial m}$ and $\frac{\partial U}{\partial n}$, the direction of the gradient can be adjusted. The process of training the network to fit to the action value function is also a process of improving the expected return U. When updating the network, θ is roughly changing in the direction of increasing U. So the recall of the minority class m can be improved when the direction of $\frac{\nabla m}{\nabla \theta}$ is more consistent with the direction of $\frac{\nabla U}{\nabla \theta}$. In addition, if the reward and punishment value of the minority class is too large or that of the majority class is too small, the recall of the majority class will be poor, since the weight of $\frac{\nabla n}{\nabla \theta}$ in $\frac{\nabla U}{\nabla \theta}$ is too small.

The process of setting the reward function and the rules of the game is an indirect process of designing the loss function. The change of the loss function affects the training of the network. Therefore, setting an appropriate reward function can improve the classification performance of the model. A good reward function must first lead to the right policy. When the MDP construction is completed, the optimal policy already exists objectively. This objectively existing

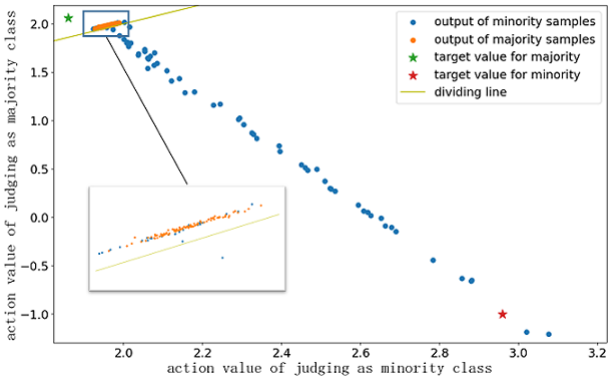

Fig. 2. Output of a trained model

optimal policy needs to be consistent with the behavior that humans want the agent to do. However, when different reward functions lead to the same policy, the trained network will still be different. Because the neural network can only obtain an approximation of the optimal policy. Different reward functions lead to different approximate results, for they will affect the the gradient of the expected return, and the action value function, which will change the target value of the output. In TV algorithm, the rewards are appropriately adjusted. Therefore, the classification performance on the minority class is improved.

IV. Experiment

In this section, experiments are used to verify the performance of the algorithm in solving the imbalanced classification problem, which mainly includes two parts: binary classification and multi-classification.

A. Binary Classification

a) Setup Details: This part conducts experiments on the text data set IMDB and three image data sets Cifar10, MNIST, Fashion-MNIST. The image data sets are all ten-class data sets, which need to be artificially integrated into two classes. The data sets are sampled into different imbalanced proportions, which is shown in TABLE I.

The comparison algorithm was DQNimb. Baseline directly trains the neural network with the cross-entropy loss function. All methods use the same network structure. A single-layer LSTM network with an embedding layer is employed on the text data set, and a convolution neural network with two convolution layers is employed on the image data sets. The output of the neural network is a vector of length 2 containing the action values of the two actions. The Adam optimizer is used in the experiment, and the learning rate is set to 0.00025. G-mean is used as the evaluation criterion. It is the geometry mean of the recall of different classes of samples.

TABLE I
Sampling for different data sets

Data set	Classes	Ratio	Minority	Majority
IMDB	0|1	10%	1250	12500
		5%	625	
		2%	250	
Cifar10	1|3,4,5,6	4%	800	20000
		2%	400	
		1%	200	
		0.5%	100	
	7|8,9	4%	400	10000
		2%	200	
		1%	100	
		0.5%	50	
Fashion-MNIST	0,2|1,3	4%	480	12000
		2%	240	
		1%	120	
		0.5%	60	
	4,5,6|7,8,9	4%	720	18000
		2%	360	
		1%	180	
		0.5%	90	
MNIST	2|0,1,3-9	1%	540	54000
		0.2%	108	
		0.1%	54	
		0.05%	27	

TABLE II
Experimental results on different data sets

Data set	Ratio	TV	DQNimb	Baseline
IMDB	10%	**0.806**	0.749	0.587
	5%	**0.688**	0.652	0.419
	2%	0.580	**0.593**	0.134
Cifar10(1)	4%	**0.937**	0.916	0.869
	2%	**0.904**	0.887	0.824
	1%	0.850	**0.870**	0.730
	0.5%	**0.794**	0.753	0.579
Cifar10(2)	4%	**0.913**	0.907	0.815
	2%	**0.875**	0.874	0.758
	1%	0.807	**0.819**	0.677
	0.5%	0.693	**0.706**	0.513
Fashion-MNIST(1)	4%	**0.969**	0.953	0.921
	2%	0.959	**0.961**	0.885
	1%	**0.959**	0.952	0.853
	0.5%	**0.939**	0.930	0.757
Fashion-MNIST(2)	4%	0.963	**0.981**	0.951
	2%	0.975	**0.979**	0.926
	1%	0.970	**0.982**	0.872
	0.5%	**0.953**	0.945	0.821
MNIST	1%	**0.996**	0.974	0.967
	0.2%	**0.985**	0.978	0.923
	0.1%	0.954	**0.975**	0.856
	0.05%	**0.967**	0.876	0.694

b) Result Analysis: The experimental results are shown in TABLE II. It can be found that the performance of the algorithm on different data sets is different. The more complex the data set, the worse the performance of the algorithm. Among the image data sets, Cifar10 is the most complex data set and MNIST is the simplest data set, achieving the lowest and highest classification metrics, respectively. IMDB is more complex as a text data set, and is correspondingly more difficult to classify. In addition, the more imbalanced the positive and negative sample data, the more difficult the classification. Taking the IMDB data set as an example, when the imbalance ratio is 2%, the baseline classifier has basically lost the ability to classify samples of minority classes, and G-mean of the TV algorithm is lower than 0.6.

The TV algorithm is a variant of the DQNimb algorithm. Both construct MDPs based on imbalanced data sets and solve optimal policies. However, the DQNimb algorithm uses the classical DQN algorithm to solve the MDP, while the TV uses the derivation to directly obtain the value function. Both have achieved good results on imbalanced classification tasks, and have significantly improved the classification ability compared to the baseline.

B. multi-classification

a) Setup Details: This experiment is carried out on three ten-class data sets, MNIST, Fashion-MNIST, and Cifar10. In the experiment, two sampling methods were adopted, 25% of the samples of some classes were sampled as the minority class, and the number of samples of the other classes remained unchanged, as the majority class.

The algorithm and neural network structure adopted in the experiment are the same as the binary classification experiment. And G-mean$_{total}$ [20] is used as the evaluation criterion instead of G-mean. Accuracy is also used as the evaluation criterion.

TABLE III
ACCURACY OF THE EXPERIMENT RESULTS

Data set	Sampling classes	TV	DQNimb	Baseline
MNIST		**99.3%**	99.2%	98.4%
Fashion-MNIST	0,1	91.1%	91.5%	87.7%
Cifar10		**63.2%**	62.4%	52.8%
MNIST		**98.9%**	98.8%	98.7%
Fashion-MNIST	0-7	**89.9%**	88.8%	87.4%
Cifar10		**58.8%**	57.8%	51.5%

TABLE IV
G-MEAN$_{total}$ OF THE EXPERIMENT RESULTS

Data set	Sampling classes	TV	DQNimb	Baseline
MNIST		0.991	**0.992**	0.984
Fashion-MNIST	0,1	0.909	**0.918**	0.882
Cifar10		0.626	**0.633**	0.535
MNIST		0.988	**0.989**	0.987
Fashion-MNIST	0-7	**0.896**	0.891	0.875
Cifar10		0.579	**0.589**	0.524

b) Result Analysis: The experimental results under the two evaluation criterion are shown in TABLE III and TABLE IV.Compared with the binary classification problem, the multi-classification problem is more difficult, and the results obtained tend to be relatively poor. So this part is just a preliminary experiment. The sampling method adopted is relatively simple, and the imbalance ratio is relatively moderate. Cifar10, as the most complex data set among the three data sets, achieved the worst classification performance. And MNIST can achieve good classification results even under the baseline.

Both the TV and DQNimb algorithms significantly improve the performance of imbalanced classification relative to the baseline. The results of the two algorithms are similar. However, the accuracy of TV is generally better than that of DQNimb, while the G-mean$_t otal$ of DQNimb is better. It indicates that the model obtained by DQNimb has a more balanced classification effect on samples of different classes. DQNimb solves the MDP through the DQN algorithm, and the value function estimated by DQN deviates from the real value function, which may lead to the difference in the results of the two algorithms.

V. CONCLUSION

This paper proposes an algorithm based on constructing and solving MDP to solve the problem of imbalanced classification, the TV algorithm. The method first transforms the imbalanced data set into an MDP. Based on the simple dynamics of this MDP, the action value function under the optimal policy can be directly calculated, and a neural network is used to fit this value function. For unknown samples on the test set, the value of different actions can be compared to make judgment actions. This method is relatively simple to implement and achieves good classification results on multiple imbalanced data sets. In addition, this article also analyzes the reason why the TV algorithm is effective from two perspectives. The setting of the reward function can influence both the target value and the gradient of the long-term expected return. The shortcomings of this article are that the setting of the reward function is more dependent on the adjustment of experience, and there is no good theoretical basis. In the future, with the inverse reinforcement learning method, it may be possible to set a more appropriate reward function.

REFERENCES

[1] Japkowicz, N., Stephen, S. (2002). The class imbalance problem: A systematic study. Intelligent data analysis, 6(5), 429-449.

[2] Weiss, G. M. (2004). Mining with rarity: a unifying framework. ACM Sigkdd Explorations Newsletter, 6(1), 7-19.

[3] He, H., Garcia, E. A. (2008). Learning from imbalanced data, IEEE T. Knowl. Data En., 21, 1263–1284.

[4] Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. Expert systems with applications, 73, 220-239.

[5] Mani, I., Zhang, I. (2003, August). kNN approach to unbalanced data distributions: a case study involving information extraction. In Proceedings of workshop on learning from imbalanced datasets (Vol. 126, pp. 1-7). ICML.

[6] Chawla, N. V., Bowyer, K. W., Hall, L. O., Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.

[7] Peng, P., Zhang, W., Zhang, Y., Xu, Y., Wang, H., Zhang, H. (2020). Cost sensitive active learning using bidirectional gated recurrent neural networks for imbalanced fault diagnosis. Neurocomputing, 407, 232-245.

[8] Puterman, M. L. (1990). Markov decision processes. Handbooks in operations research and management science, 2, 331-434.

[9] Lin, E., Chen, Q., Qi, X. (2020). Deep reinforcement learning for imbalanced classification. Applied Intelligence, 50(8), 2488-2502.

[10] Tripathi, S., Mehrotra, R., Bansal, V., Upadhyay, S. (2020, September). Analyzing sentiment using IMDb dataset. In 2020 12th International Conference on Computational Intelligence and Communication Networks (CICN) (pp. 30-33). IEEE.

[11] Doon, R., Rawat, T. K., Gautam, S. (2018, November). Cifar-10 classification using deep convolutional neural network. In 2018 IEEE Punecon (pp. 1-5). IEEE.

[12] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE signal processing magazine, 29(6), 141-142.

[13] Xiao, H., Rasul, K., Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747.

[14] Nishino, T., Ozaki, R., Momoki, Y., Taniguchi, T., Kano, R., Nakano, N., ... Nakamura, K. (2020, November). Reinforcement learning with imbalanced dataset for data-to-text medical report generation. In Findings of the Association for Computational Linguistics: EMNLP 2020 (pp. 2223-2236).

[15] Wu, Y., Huang, X. (2022). Unsupervised Reinforcement Adaptation for Class-Imbalanced Text Classification. arXiv preprint arXiv:2205.13139.

[16] Yang, J., El-Bouri, R., O'Donoghue, O., Lachapelle, A. S., Soltan, A. A., Clifton, D. A. (2022). Deep Reinforcement Learning for Multi-class Imbalanced Training. arXiv preprint arXiv:2205.12070.

[17] Zhang, H., Liu, W., Liu, Q. (2020). Reinforcement online active learning ensemble for drifting imbalanced data streams. IEEE Transactions on Knowledge and Data Engineering.

[18] Lee, J., Sun, Y. G., Sim, I., Kim, S. H., Kim, D. I., Kim, J. Y. (2022). Non-Technical Loss Detection Using Deep Reinforcement Learning for Feature Cost Efficiency and Imbalanced Dataset. IEEE Access, 10, 27084-27095.

[19] Fan, S., Zhang, X., Song, Z. (2021). Imbalanced sample selection with deep reinforcement learning for fault diagnosis. IEEE Transactions on Industrial Informatics, 18(4), 2518-2527.

[20] Espíndola, R. P., Ebecken, N. F. (2005). On extending f-measure and g-mean metrics to multi-class problems. WIT Transactions on Information and Communication Technologies, 35.