Electronic Research Archive

ERA, 31(5): 2501–2518.
DOI: 10.3934/era.2023127
Received: 31 December 2022
Revised: 19 February 2023
Accepted: 23 February 2023
Published: 06 March 2023

http://www.aimspress.com/journal/ERA

Research article

Improving performance of decision threshold moving-based strategies by integrating density-based clustering technique

Mengke Lu, Shang Gao, Xibei Yang and Hualong Yu*

School of Computer, Jiangsu University of Science and Technology, Zhenjiang, Jiangsu, China

* **Correspondence:** Email: yuhualong@just.edu.cn; Tel: +8615952894360.

Abstract: Class imbalance learning (CIL), which aims to addressing the performance degradation problem of traditional supervised learning algorithms in the scenarios of skewed data distribution, has become one of research hotspots in fields of machine learning, data mining, and artificial intelligence. As a postprocessing CIL technique, the decision threshold moving (DTM) has been verified to be an effective strategy to address class imbalance problem. However, no matter adopting random or optimal threshold designation ways, the classification hyperplane could be only moved parallelly, but fails to vary its orientation, thus its performance is restricted, especially on some complex and density variable data. To further improve the performance of the existing DTM strategies, we propose an improved algorithm called CDTM by dividing majority training instances into multiple different density regions, and further conducting DTM procedure on each region independently. Specifically, we adopt the well-known DBSCAN clustering algorithm to split training set as it could adapt density variation well. In context of support vector machine (SVM) and extreme learning machine (ELM), we respectively verified the effectiveness and superiority of the proposed CDTM algorithm. The experimental results on 40 benchmark class imbalance datasets indicate that the proposed CDTM algorithm is superior to several other state-of-the-art DTM algorithms in term of G-mean performance metric.

Keywords: class imbalance learning; decision threshold moving; clustering; DBSCAN

1. Introduction

In recent years, learning from imbalanced data distributions has gradually developed to be a

hotspot issue in machine learning field due to this issue is emerging in more and more practical applications, including medical diagnosis [1], industrial fault diagnosis [2,3], network intrusion detection [4], financial fraud detection [5,6], text classification [7,8], bioinformatics [9], soil classification [10], air performance prediction [11], and criminal linkage detection [12].

In nature, class imbalance problem happens as the number of instances belonging to a specific class is overwhelming to that belonging to the other classes, further causing some traditional supervised learning algorithms exceedingly focus on the majority class, but depress the performance of minority classes. In recent two decades, a number of learning algorithms for addressing imbalanced classification problem have been proposed, and they could be roughly divided into four following categories: sampling [13–15], cost-sensitive learning [16–18], decision threshold moving (DTM) [19–22], and ensemble learning [23–26]. Sampling can be regarded as a pre-processing technique for dealing with class imbalance learning problem as it balances the data distribution of different classes by either adding the instances belonging to the minority class or decreasing the instances belonging to the majority class. Cost-sensitive learning inserts the concept of cost or weight into traditional supervised learning algorithms, and it generally provides higher cost or weight for the instances belonging to the minority class to make the learning model focus more on such class. As the cost-sensitive learning technique is directly merged into traditional supervised learning algorithms, thus it can be seen as a in the processing technique for addressing imbalanced classification problem. As for DTM strategy, it first trains a biased model by a traditional classification algorithm, and then moves the trained classification hyperplane towards the majority class empirically or optimally. Therefore, the DTM strategy can be seen as a *postprocessing* class imbalance learning technique. Ensemble learning combines Bagging, Boosting or random space paradigm with one of single class imbalance learning strategies which are mentioned above to further improve the robustness of learning model. Specifically, among the existing class imbalance learning techniques, we can't say which one is always better than others as each one of them has its own pros and cons.

In this study, we focus on DTM class imbalance learning strategy because that its postprocessing mechanism makes it not only avoid to destroying the potential data distribution, but also be independent of classification algorithm. Its disadvantage lies in that it is not easy to find the appropriate threshold to maximize the classification performance. Then we have to face a problem, that is, how to determine the decision threshold? The existing DTM algorithms generally solve this problem based on one of two following strategies: empirical [19,20] and optimal [21,22]. The empirical strategies calculate the decision threshold according to class imbalance ratio, which tend to ignore the real data distribution. While the optimal strategies could adaptively provide an appropriate decision threshold based on the performance feedback from imbalanced training data, which do not need to directly explore the data distribution. Although the optimal DTM strategies have been verified to be effective, their performances are still limited by some potential factors about data distribution, for example, when there exists density variation in data distribution, the performance of the adjusted classifier cannot be guaranteed as the classification boundary can be only parallel translated, but cannot be changed direction. To address the problem referred above, we propose a modified DTM algorithm called clustering-based decision threshold moving (CDTM) in this paper.

Specifically, the proposed CDTM algorithm combines a popular density-based clustering algorithm named Density-Based Spatial Clustering of Applications with Node (DBSCAN) [27–29] and one of the traditional optimal DTM algorithms [21,22]. DBSCAN takes charge of dividing all

majority instances into multiple different groups according to their density distribution information. Then, for each majority instance group, it is integrated with all minority instances to constitute a training subset. Next, on each training subset, an optimal DTM algorithm is conducted to search the best decision threshold. When a unseen instance is received, it is first decided which group it falls in by the Gaussian naïve bayes (GNB) rule [30,31], and then the corresponding DTM algorithm is called to classify for it. The proposed CTDM algorithm uses the idea of divide-and-conquer method, thus it can adapt the complex density variation existing in feature space well. In this paper, we adopt the optimal decision output compensation (ODOC) [22] as the basic strategy, and in context of support vector machine (SVM) [32,33] and extreme learning machine (ELM) [34,35] respectively, to verify the effectiveness and feasibility of CTDM. The experimental results on 40 benchmark class imbalance datasets indicate that the proposed CDTM algorithm is superior to several other state-of-the-art DTM algorithms in term of G-mean performance metric. The novelty of this work reflects at two following aspects: 1) to our best knowledge, it is the first DTM technique which can change the direction of original classification boundary, and 2) it can better adapt complex data distribution than the emerging DTM techniques as the idea of divide-and-conquer has been embedded inside it.

The rest of this paper is organized as follows: Section 2 reviews the previous work related to DTM, as well indicates why all existing DTM algorithms could not adapt data distribution well. In Section 3, the proposed CTDM algorithm is described in detail. Section 4 provides some compared experimental results and gives the corresponding discussions. Finally, Section 5 concludes the findings of this paper, and indicates future work.

2. Related work

As mentioned in Section 1, there are two different kinds of strategies to decide the threshold in DTM methods. One is empirical, and the other is optimal. In fact, the primitive DTM approaches tend to designate an empirical value for the threshold, e.g., SVM-THR algorithm which is proposed by Lin and Chen [20]. In SVM-THR, the threshold θ is calculated by the following equation:

$$\theta = \frac{N^- - N^+}{N^- + N^+ + 2} \tag{1}$$

where N^- and N^+ denote the number of majority and minority instances in the training set, respectively. It is clear that the class imbalance ratio, i.e., N^-/N^+, is higher, then the threshold θ is larger. It seems to conform to the actual situation as in general, on a highly skewed data distribution, the minority class is inclined to be severer destroyed. However, the rule is not absolute as the impact of class imbalance distribution is not only related with the class imbalance ratio, but also associates several other factors [36]. Therefore, adopting the empirical threshold in DTM may under-compensate or over-compensate the classification boundary, further causing poor classification performance.

In comparison with the empirical DTM strategies, the optimal strategies can effectively adapt the data distribution. In [21], an optimal DTM strategy called OTHR is presented. The OTHR firstly trains a biased classification model which is required to provide soft output for each training instance, and then it collects all minority instances which have been misclassified by the trained model. Next, we determine all candidate thresholds according to the following equation,

$$\theta_i = \frac{-h(x_i^+) - h(n_i^+)}{2} \tag{2}$$

where θ_i denotes the ith candidate threshold, while $h(x_i^+)$ and $h(n_i^+)$ represent the decision outputs of the ith misclassified minority instance and its nearest majority neighbor emerging in the direction of majority class. Finally, on the original training set, we compare the harmonic performance of F-measure and G-mean of adopting each candidate threshold, further determine the optimal threshold. The other optimal DTM strategy is ODOC [22] which regards the threshold designation as an optimization problem in continuous space. Therefore, the ODOC adopts mature optimization algorithms to search the best threshold for DTM. Specifically, for binary-class imbalance problem, the ODOC uses the golden section algorithm [37], while for multi-class imbalance problem, it uses the particle swarm optimization (PSO) algorithm. G-mean performance metric is used as the fitness function to evaluate the quality of threshold. We note that both OTHR and ODOC strategies do not explore data distribution directly, by adapt it in an indirect fashion.

However, we note that no matter empirical or optimal DTM strategies cannot adapt data distribution well. Figure 1 explains the corresponding reason. In Figure 1, it is not difficult to observe that the traditional DTM strategies can only parallelly move the initially trained classification boundary, but cannot change its direction, further providing a sub-optimal solution. Therefore, it is necessary to amend this bias by providing individual threshold for the instances lying in different regions in feature space.

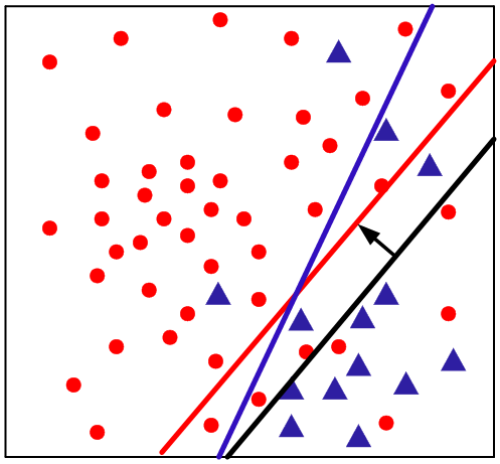

Figure 1. Why traditional DTM strategies cannot adapt data distribution well, where the black line denotes the classification boundary trained directly on original data, the red line denotes the amended boundary by traditional DTM strategies, and the blue line denotes the ideal amended classification boundary.

3. Methods

3.1. Clustering majority instances by DBSCAN technique

To realize the individual DTM, we firstly adopt a well-known density-based clustering algorithm called DBSCAN [27–29] to divide majority instances into multiple disconnected groups. We select to use DBSCAN because density variation existing in data distribution often asks for more

discrepant thresholds.

Specifically, the DBSCAN is a traditional density clustering method that portrays the closeness of data distribution based on a pair of parameters (E, M), where E denotes the local neighborhood radius and M describes the threshold of the number of instances in a local neighborhood. DBSCAN searches for clusters by examining the E local neighborhood of each instance in the data set, and if the E neighborhood of an instance x contains more than M instances, a cluster with x as the core object would be created, and then the set of instances which are density reachable from these core objects is iteratively aggregated. This process is roughly described in Figure 2(a), where the red points represent the core instances and the circles denote their E local neighborhoods. The core objects connected by arrows mean that they are density reachable, and all instances in the E neighborhoods of these core objects are clustered into a single group. Then, we can divide instances into multiple disconnected regions, and each of them can be seen as an independent cluster (see Figure 2(b)). Specifically, the DBSCAN algorithm has been shown to discover the clusters with arbitrary shapes in data with noise.

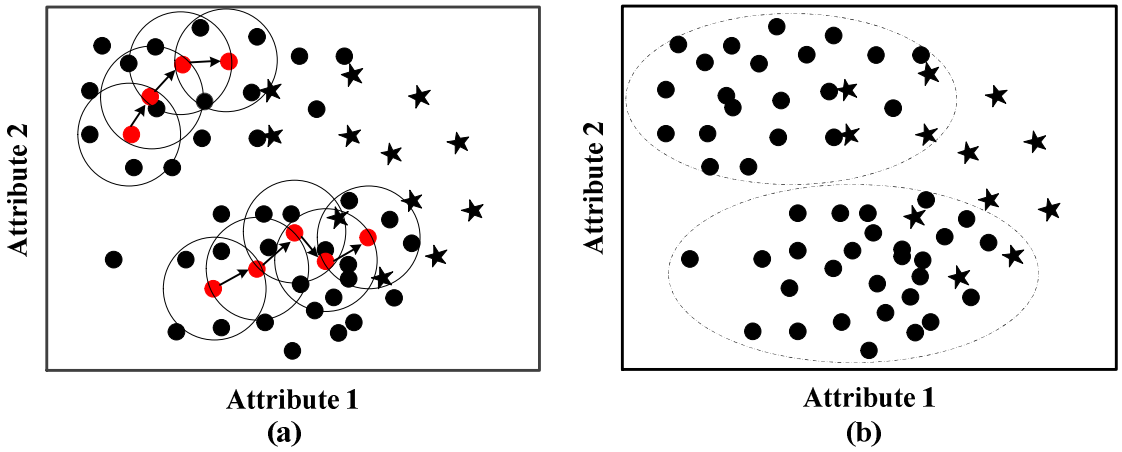

Figure 2. Graphical representation of the DBSCAN operating mechanism.

In DBSCAN, there are two important issues requiring to be focused on: one is how to designate the pair of parameters (E, M), and the other is how to determine which cluster a new unseen instance belongs to. For the first issue, we know both parameters are very important for the final clustering results. In this paper, we empirically designated the pair of parameters as (0.15, 5) according to the feedback from lots of experiments. As for the second issue, we suppose that all instances in each cluster satisfy a multivariate Gaussian distribution, and hence adopt GNB rule [30,31] to judge which cluster an instance belongs to. Suppose $P(c_i)$ and $P(x|c_i)$ denote the prior probability of the cluster c_i, and the conditional probability of an instance x relative to the cluster c_i, respectively. Then, the probability of the instance x belonging to the cluster c_i could be calculated as follows,

$$P(c_i|x) = \frac{P(c_i)P(x|c_i)}{P(x)} \tag{3}$$

Obviously, it is more reasonable for adopting GNB to determine the cluster the instance falls in than comparing the distance between each cluster centroid and the instance as the GNB rule can adapt the potential density variation existing among different clusters well.

3.2. Description about CDTM algorithm

In our proposed CDTM algorithm, the DTM strategy is conducted on each training subset that contains all majority instances from a specific cluster divided by DBSCAN and all minority training instances. Specifically, considering that the ODOC strategy has presented more robust performance than OTHR [22], we adopt the ODOC as the basic DTM strategy in our CDTM algorithm. That is to say, on each sub-optimization problem, the golden section optimization algorithm is used to search the best threshold. The procedure of the proposed CDTM algorithm is simply described as follows.

Algorithm: CDTM

Input: An imbalanced training set $\Phi = \{(x_1, y_1), (x_2, y_2), \ldots\ldots, (x_N, y_N)$, where N denotes the number of training instances, and $y_i \in \{-1,1\}$; a test set $\Psi = \{x'_1, x'_2, \ldots, x'_K\}$; the pair of parameters (E, M) in DBSCAN.

Output: An optimal threshold set $\theta = \{\theta_1, \theta_2, \ldots, \theta_M\}$, where M denotes the number of clusters generated by DBSCAN; the predict results for test set $Y = \{y'_1, y'_2, \ldots, y'_K\}$

Training Procedure:

1. The training set is first divided into a majority set Φ^- and a minority set Φ^+;
2. The DBSCAN algorithm is used to divide the majority set Φ^- into S disconnected clusters;
3. for i=1: S
4. The i-th majority cluster c_i is combined with the minority set Φ^+ to constitute Φ_i, and we then use it to train a soft output classifier $Classifier_i$;
5. Implementing the ODOC DTM strategy on Φ_i to amend the $Classifier_i$ and obtaining the corresponding best threshold θ_i;
6. end for
7. Output the optimal threshold set θ.

Testing Procedure:

8. for i = 1: K
9. Using the reserved distribution information to determine the cluster c_j the instance x'_i belongs to;
10. Calling $Classifier_j$ to classify the instance x'_i and acquiring its soft output $h(x'_i)$;
11. Calculating the amended output $h'(x'_i)$ for x'_i by calling θ_j;
12. Calculating the predict label y'_i for x'_i;
13. end for
14. Output the predict results Y for test set.

In addition, we also provide a flowchart to describe how to train a CDTM model (see Figure 3). Specifically, in CDTM training procedure, the classifier is required to provide soft outputs. Specifically, in this paper, we used two different soft-output classifiers, namely SVM [32,33] and ELM [34,35], respectively. While in the testing procedure, each test instance x'_i first requires determining which cluster it belongs to, and then its amended output could be calculated by,

$$h'(x'_i) = h(x'_i) + \theta_j \tag{4}$$

where $h(x'_i)$ and $h'(x'_i)$ respectively denote the original and amended decision output for the testing instance x'_i, and θ_j denotes the threshold of the cluster the instance belongs to.

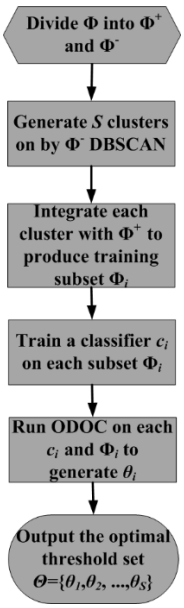

Figure 3. The flowchart of the CDTM training procedure.

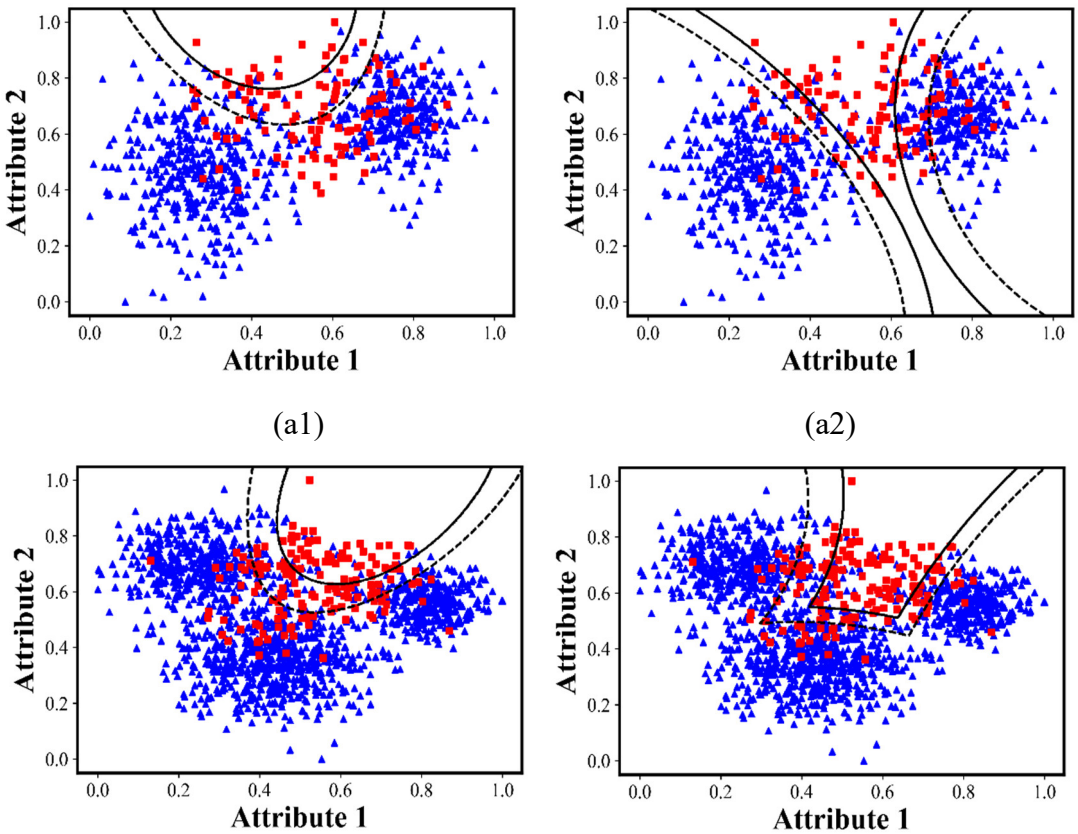

Figure 4. A comparison between traditional DTM strategy (see (a1) and (b1)) and the proposed CDTM strategy (see (a2) and (b2)), where the solid and dashed lines in each sub-figure denote the initially trained and amended classification boundaries, respectively.

In Figure 4, we use two artificial imbalanced data sets to show the difference between the traditional DTM strategies and the proposed CDTM strategy, further illustrate the necessarily of developing CDTM algorithm. It is clear that in contrast to DTM, the proposed CDTM tend to capture detailed information referring to data distribution, thus it can better adapt the data distribution and provide more appropriate amended boundary.

3.3. Time complex analysis

Except classification performance, the time complex is also an important metric to evaluate the quality of a learning algorithm.

Suppose there are N training instances, then for CDTM algorithm, the DBSCAN clustering procedure requires $O(N^2)$ training time, and the SVM training procedure also needs to consume $O(N^2)$ time, but in worst case, it can cost $O(N^3)$ time. In contrast to SVM, training ELM is more time-saving as it only consumes $O(L^2N)$ time, where L denotes the number of layer-nodes. Finally, the golden section search procedure requires costing only $O(log_2N)$ time. Therefore, we can say that although the CDTM algorithm is always more time-consuming than several traditional DTM algorithms, its time-complex is still acceptable.

4. Experiments

4.1. Datasets

In our experiments, we used 40 binary-class imbalanced datasets acquired from Keel data repository [38] to verify the effectiveness and superiority of the proposed CDTM algorithm. Specifically, these datasets have a wide variation at aspect of number of instances, features and class imbalance ratios. The details about these used datasets could be found in Table 1.

Table 1. Details about the used datasets.

Dataset	Number of instances	Number of features	Class imbalance ratio
pima	768	8	1.87
saheart	462	8	1.89
tae-3_vs_1-2	151	5	1.9
led7digit-0-1-2_vs_3-4-5-6-7	398	7	1.99
glass0	214	9	2.06
tae-1_vs_2-3	151	5	2.08
phoneme	5404	5	2.41
haberman	306	3	2.78
winequality-red-7_vs_6	837	11	3.21
ecoli1	336	7	3.36
winequality-white-4_7_vs_5-6	4698	11	3.5
led7digit-0-1_vs_2-3-4-5-6-7	398	7	3.85
appendicitis	106	7	4.05
newthyroid2	215	5	5.14

Continued on next page

winequality-white-4_vs_7	1043	11	5.4
glass-3-5_vs_1-2-6-7	214	9	6.13
glass-7_vs_1-2-3-5-6	214	9	6.38
winequality-red-7_vs_5-6	1518	11	6.63
pageblocks2345	5473	10	8.77
page-blocks0	5472	10	8.79
winequality-white-4_vs_5	1620	11	8.94
yeast-0-3-5-9_vs_7-8	506	8	9.12
yeast-0-2-5-6_vs_3-7-8-9	1004	8	9.14
yeast-0-2-5-7-9_vs_3-6-8	1004	8	9.14
ecoli-0-4-6_vs_5	203	6	9.15
ecoli-0-1_vs_2-3-5	244	7	9.17
ecoli-0-2-6-7_vs_3-5	224	7	9.18
ecoli-0-6-7_vs_5	220	6	10
ecoli-0-1-4-7_vs_2-3-5-6	336	7	10.59
winequality-white-4_vs_6	2361	11	13.48
ecoli4	336	7	15.8
abalone9-18	731	7	16.4
page-blocks-1-3_vs_4	472	10	15.86
zoo-3_vs_1-2-4-5-6-7	101	16	19.2
yeast-1-4-5-8_vs_7	693	8	22.1
winequality-white-4_vs_5-6	3818	11	22.42
yeast-2_vs_8	482	8	23.1
yeast4	1484	8	28.1
yeast-1-2-8-9_vs_7	947	8	30.57
yeast6	1484	8	41.4

4.2. Experimental settings

All experiments were run on a 2.40 GHz Intel(R) Core (TM) i5 10200H 8-core CPU with 16 GB RAM, and the Python 3.8 environment.

In context of SVM [32,33] and ELM [34,35], we respectively compared the proposed CDTM algorithm with four other algorithms. One is baseline without any process for class imbalance, and three others are THR [20], OTHR [21] and ODOC [22] strategies, respectively.

To guarantee the impartiality of compared experiments, for each classifier, we used the same parameter settings. Specifically, the SVM uses RBF kernel with default parameters, that is, the kernel width $\sigma = 0.01$ and the penalty factor $C = 1$, while the ELM uses sigmoid activation function with 20 hidden nodes. As for our proposed CDTM algorithm, it used the empirical DBSCAN parameter pair (0.15, 5) which has been indicated in Section 3.

Specifically, all experiments were conducted on standardized datasets. That means for each dataset, its each feature should be first proportionally scaled into [0, 1] by,

$$x'_{ij} = \frac{x_{ij} - x_{minj}}{x_{maxj} - x_{minj}} \tag{5}$$

where x_{ij} and x'_{ij} respectively denote the original and standardized values for the *i*th instance on the *j*th feature, while x_{maxj} and x_{minj} represent the maximal and minimal values in the original *j*th feature space throughout all training instances.

In addition, we adopted a most popular performance metric, i.e., G-mean [39], to evaluate and compare the quality of various compared algorithms. G-mean metric can be calculated as follows:

$$\mathrm{G-mean} = \sqrt{Acc_+ \times Acc_-} \tag{6}$$

where Acc_+ and Acc_- represent the predicted accuracy of positive (minority) class and negative (majority) class, respectively. Therefore, we can say that the G-mean metric reflects the tradeoff between the classification performance of the minority class and that of the majority class.

Finally, to exclude some random factors for experimental results, we also conducted 10 times' external five-folds cross validation, and further provided experimental results in the form of mean ± standard deviation.

4.3. Comparison between the proposed CDTM and several other DTM algorithms

Tables 2 and 3 present the G-mean results of various DTM strategies in context of SVM and ELM, respectively.

Table 2. G-mean results of five compared algorithms in context of SVM, where the best result on each dataset have been highlighted in bold.

Dataset	SVM	SVM-THR	SVM-OTHR	ODOC-SVM	SVM-CDTM
pima	0.6574 ± 0.0403	0.6733 ± 0.0401	0.6773 ± 0.0369	**0.6820 ± 0.0406**	0.6667 ± 0.0350
saheart	0.5512 ± 0.0459	0.5704 ± 0.0499	0.5732 ± 0.0488	0.5680 ± 0.0426	**0.5745 ± 0.0464**
tae-3_vs_1-2	0.7036± 0.0910	0.6891 ± 0.0857	0.6911 ± 0.0855	**0.7313 ± 0.0827**	0.7187 ± 0.1150
led7digit-0-1-2_vs_3-4-5-6-7	0.8563 ± 0.0329	0.8610 ± 0.0343	0.8668 ± 0.0329	**0.8732 ± 0.0335**	0.8668 ± 0.0349
glass0	0.7939 ± 0.0607	0.7129 ± 0.0687	**0.7956 ± 0.0638**	0.7945 ± 0.0645	0.7545 ± 0.0754
tae-1_vs_2-3	0.6435 ± 0.1007	0.6636 ± 0.0950	0.6941 ± 0.0885	**0.6942 ± 0.0797**	0.6705 ± 0.0865
phoneme	0.8450 ± 0.0132	0.8645 ± 0.0087	**0.8651 ± 0.0087**	0.8645 ± 0.0084	0.8641 ± 0.0097
haberman	0.4159 ± 0.0980	0.4393 ± 0.1334	0.4479 ± 0.0826	0.4430 ± 0.1835	**0.4974 ± 0.1622**
winequality-red-7_vs_6	0.7235 ± 0.0376	0.7081 ± 0.0367	0.7380 ± 0.0450	**0.7506 ± 0.0324**	0.7255 ± 0.0390
ecoli1	0.8436 ± 0.0637	0.8461 ± 0.0428	0.8480 ± 0.0545	**0.8569 ± 0.0458**	0.8448 ± 0.0506
winequality-white-4_7_vs_5-6	0.6325 ± 0.0253	0.7115 ± 0.0193	0.7057 ± 0.0247	0.7191 ± 0.0149	**0.7201 ± 0.0155**
led7digit-0-1_vs_2-3-4-5-6-7	0.8581 ± 0.0594	0.8583 ± 0.0534	0.8371 ± 0.0535	**0.8673 ± 0.0433**	0.8600 ± 0.0487
appendicitis	0.6282 ± 0.2206	0.6299 ± 0.1263	0.6066 ± 0.2130	**0.6374 ± 0.1733**	0.6318 ± 0.1772
newthyroid2	0.9425 ± 0.0539	0.9775 ± 0.0236	0.9312 ± 0.0620	**0.9811 ± 0.0228**	**0.9811 ± 0.0228**
winequality-white-4_vs_7	0.8432 ± 0.0429	0.8523 ± 0.0228	0.7878 ± 0.0467	**0.8809 ± 0.0251**	0.8681 ± 0.0257
glass-3-5_vs_1-2-6-7	0.5692 ± 0.2067	**0.6980 ± 0.1294**	0.4889 ± 0.2721	0.6220 ± 0.2125	0.6225 ± 0.1919
glass-7_vs_1-2-3-5-6	0.8992 ± 0.0792	**0.9146 ± 0.0613**	0.9027 ± 0.0812	0.9122 ± 0.0549	0.9022 ± 0.0543
winequality-red-7_vs_5-6	0.7524 ± 0.0482	0.7544 ± 0.0306	0.7793 ± 0.0455	**0.7862 ± 0.0444**	0.7795 ± 0.0403
pageblocks2345	0.8744 ± 0.0215	0.9391 ± 0.0178	0.9402 ± 0.0172	**0.9407 ± 0.0158**	0.9405 ± 0.0143
page-blocks0	0.8707 ± 0.0245	0.9340 ± 0.0192	**0.9369 ± 0.0176**	0.9352 ± 0.0185	0.9355 ± 0.0168
winequality-white-4_vs_5	0.5561 ± 0.0734	0.6258 ± 0.0423	0.5358 ± 0.2030	0.6319 ± 0.0685	**0.6544 ± 0.0694**
yeast-0-3-5-9_vs_7-8	0.4679 ± 0.1714	0.5722 ± 0.0585	0.5991 ± 0.1187	0.5988 ± 0.1220	**0.6691 ± 0.0877**
yeast-0-2-5-6_vs_3-7-8-9	0.6869 ± 0.0725	0.7284 ± 0.0557	0.7385 ± 0.0653	0.7379 ± 0.0646	**0.7418 ± 0.0601**
yeast-0-2-5-7-9_vs_3-6-8	0.8648 ± 0.0535	0.8773 ± 0.0481	**0.8779 ± 0.0483**	0.8656 ± 0.0485	0.8706 ± 0.0498
ecoli-0-4-6_vs_5	0.8441 ± 0.2046	0.8858 ± 0.1102	0.8384 ± 0.2073	**0.8795 ± 0.1099**	0.8649 ± 0.1106

Continued on next page

ecoli-0-1_vs_2-3-5	0.7792 ± 0.1926	0.8243 ± 0.0892	0.7787 ± 0.2005	0.8584 ± 0.0997	**0.8742 ± 0.0840**
ecoli-0-2-6-7_vs_3-5	0.7894 ± 0.1147	0.8318 ± 0.0930	0.7013 ± 0.1119	0.8323 ± 0.1174	**0.8358 ± 0.1202**
ecoli-0-6-7_vs_5	0.8051 ± 0.1973	0.8415 ± 0.0885	0.7391 ± 0.1947	0.8579 ± 0.0957	**0.8618 ± 0.0943**
ecoli-0-1-4-7_vs_2-3-5-6	0.8076 ± 0.1057	0.7781 ± 0.0875	0.8066 ± 0.1226	0.8101 ± 0.1026	**0.8238 ± 0.0842**
winequality-white-4_vs_6	0.6315 ± 0.0664	0.7072 ± 0.0386	0.7204 ± 0.0570	0.7235 ± 0.0584	**0.7238 ± 0.0525**
ecoli4	0.8705 ± 0.1242	0.8390 ± 0.0920	0.8688 ± 0.1220	0.8590 ± 0.1204	**0.8773 ± 0.0898**
abalone9-18	0.6693 ± 0.1104	**0.7891 ± 0.0714**	0.7550 ± 0.1210	0.7420 ± 0.1270	0.7464 ± 0.1274
page-blocks-1-3_vs_4	0.9592 ± 0.0499	0.9661 ± 0.0311	0.9110 ± 0.0967	0.9658 ± 0.0361	**0.9665 ± 0.0306**
zoo-3_vs_1-2-4-5-6-7	0.5607 ± 0.3872	0.6324 ± 0.3615	0.5129 ± 0.3872	0.6867 ± 0.3165	**0.7253 ± 0.2484**
yeast-1-4-5-8_vs_7	0.1236 ± 0.1713	0.5127 ± 0.0643	0.5283 ± 0.1590	0.5399 ± 0.1703	**0.5543 ± 0.1648**
winequality-white-4_vs_5-6	0.4763 ± 0.0603	**0.6422 ± 0.0264**	0.6039 ± 0.0541	0.6131 ± 0.0505	0.6188 ± 0.0535
yeast-2_vs_8	0.5441 ± 0.2573	0.5827 ± 0.1528	0.5822 ± 0.2710	0.6105 ± 0.2492	**0.6107 ± 0.2490**
yeast4	0.5343 ± 0.1451	0.7171 ± 0.0524	**0.7332 ± 0.0885**	0.7151 ± 0.0858	0.7215 ± 0.0829
yeast-1-2-8-9_vs_7	0.5189 ± 0.1871	0.6249 ± 0.0551	0.6612 ± 0.1213	0.6594 ± 0.1346	**0.6784 ± 0.1148**
yeast6	0.7307 ± 0.1307	0.7530 ± 0.0558	**0.8351 ± 0.0641**	0.8269 ± 0.0714	0.8319 ± 0.0771

Table 3. G-mean results of five compared algorithms in context of ELM, where the best result on each dataset have been highlighted in bold.

Dataset	ELM	ELM-THR	ELM-OTHR	ODOC-ELM	ELM-CDTM
pima	0.6809 ± 0.0404	0.7155 ± 0.0393	0.7166 ± 0.0323	**0.7201 ± 0.0344**	0.6993 ± 0.0341
saheart	0.5787 ± 0.0564	0.6019 ± 0.0556	0.6027 ± 0.0563	0.6016 ± 0.0556	**0.6096 ± 0.0482**
tae-3_vs_1-2	0.6577 ± 0.0699	0.6628 ± 0.0678	0.6566 ± 0.0727	0.6671 ± 0.0669	**0.6936 ± 0.0939**
led7digit-0-1-2_vs_3-4-5-6-7	0.8457 ± 0.0348	0.8509 ± 0.0270	0.8437 ± 0.0336	**0.8548 ± 0.0282**	0.8219 ± 0.0526
glass0	0.6956 ± 0.0607	0.6962 ± 0.0555	**0.6995 ± 0.0557**	0.6994 ± 0.0609	0.6806 ± 0.0734
tae-1_vs_2-3	0.6630 ± 0.0870	0.6773 ± 0.0848	0.6620 ± 0.0917	**0.6774 ± 0.0835**	0.6763 ± 0.0992
phoneme	0.7983 ± 0.0136	0.8264 ± 0.0086	0.8321 ± 0.0092	0.8331 ± 0.0091	**0.8341 ± 0.0091**
haberman	0.4952 ± 0.0853	0.5794 ± 0.0768	0.5741 ± 0.0753	0.5743 ± 0.0682	**0.5879 ± 0.0781**
winequality-red-7_vs_6	0.6027 ± 0.0826	0.6916 ± 0.0270	**0.6972 ± 0.0484**	0.6935 ± 0.0470	0.6930 ± 0.0438
ecoli1	0.8000 ± 0.0656	0.8209 ± 0.0376	0.8235 ± 0.0552s	**0.8280 ± 0.0506**	0.7456 ± 0.0851
winequality-white-4_7_vs_5-6	0.4694 ± 0.0291	0.6998 ± 0.0178	0.7000 ± 0.0173	0.7009 ± 0.0165	**0.7014 ± 0.0140**
led7digit-0-1_vs_2-3-4-5-6-7	0.8552 ± 0.0508	**0.8788 ± 0.0366**	0.8826 ± 0.0362	0.8782 ± 0.0349	0.8664 ± 0.0550
appendicitis	0.4332 ± 0.2162	0.4264 ± 0.2152	0.4332 ± 0.2162	0.4552 ± 0.1741	**0.4777 ± 0.1317**
newthyroid2	0.7731 ± 0.1528	0.7535 ± 0.1514	0.7731 ± 0.1528	0.7569 ± 0.1507	**0.7799 ± 0.1339**
winequality-white-4_vs_7	0.7964 ± 0.0546	0.8428 ± 0.0265	0.8671 ± 0.0426	**0.8676 ± 0.0426**	0.8368 ± 0.0420
glass-3-5_vs_1-2-6-7	0.5295 ± 0.1836	**0.6622 ± 0.1077**	0.5855 ± 0.1544	0.5996 ± 0.1623	0.6383 ± 0.1321
glass-7_vs_1-2-3-5-6	**0.8558 ± 0.1122**	0.8496 ± 0.0803	**0.8558 ± 0.1122**	0.8394 ± 0.0850	0.8307 ± 0.0889
winequality-red-7_vs_5-6	0.5402 ± 0.0551	0.7657 ± 0.0258	0.7690 ± 0.0288	**0.7696 ± 0.0241**	0.7669 ± 0.0386
pageblocks2345	0.8305 ± 0.0293	0.9236 ± 0.0126	0.9307 ± 0.0152	0.9306 ± 0.0160	**0.9308 ± 0.0133**
page-blocks0	0.8287 ± 0.0294	0.9236 ± 0.0112	0.9318 ± 0.0157	0.9316 ± 0.0157	**0.9319 ± 0.0128**
winequality-white-4_vs_5	0.3368 ± 0.0752	0.6895 ± 0.0461	**0.6905 ± 0.0486**	0.6885 ± 0.0469	0.6538 ± 0.0639
yeast-0-3-5-9_vs_7-8	0.3810 ± 0.1429	0.6129 ± 0.0607	0.5923 ± 0.0978	0.5966 ± 0.0785	**0.6234 ± 0.0967**
yeast-0-2-5-6_vs_3-7-8-9	0.5991 ± 0.0740	0.7336 ± 0.0637	0.7358 ± 0.0561	0.7338 ± 0.0607	**0.7424 ± 0.0618**
yeast-0-2-5-7-9_vs_3-6-8	0.8278 ± 0.0489	0.8362 ± 0.0392	0.8666 ± 0.0456	0.8631 ± 0.0464	**0.8789 ± 0.0484**
ecoli-0-4-6_vs_5	0.6908 ± 0.2786	0.7365 ± 0.1782	0.6908 ± 0.2786	**0.7432 ± 0.1900**	0.7005 ± 0.1728
ecoli-0-1_vs_2-3-5	0.6580 ± 0.2459	0.7003 ± 0.2355	0.6921 ± 0.2490	0.7102 ± 0.2494	**0.7359 ± 0.1337**
ecoli-0-2-6-7_vs_3-5	0.6841 ± 0.1976	0.7273 ± 0.0949	0.6887 ± 0.1968	0.7498 ± 0.1131	**0.7726 ± 0.1147**
ecoli-0-6-7_vs_5	0.6693 ± 0.2696	0.7126 ± 0.1803	0.6678 ± 0.2695	0.6977 ± 0.2427	**0.7562 ± 0.1319**
ecoli-0-1-4-7_vs_2-3-5-6	0.7413 ± 0.1139	0.7707 ± 0.1005	0.7884 ± 0.1098	**0.7890 ± 0.1076**	0.7028 ± 0.1017
winequality-white-4_vs_6	0.4762 ± 0.0818	0.7603 ± 0.0275	0.7815 ± 0.0420	0.7773 ± 0.0447	**0.7963 ± 0.0450**
ecoli4	0.8379 ± 0.1076	0.8777 ± 0.0447	**0.8817 ± 0.0938**	0.8811 ± 0.0804	0.7501 ± 0.0916
abalone9-18	0.5512 ± 0.1183	0.7811 ± 0.0646	0.7698 ± 0.0852	0.7692 ± 0.0772	**0.7841 ± 0.1116**
page-blocks-1-3_vs_4	0.7506 ± 0.2621	0.7390 ± 0.2419	0.7506 ± 0.2621	0.7646 ± 0.2505	**0.8075 ± 0.1168**
zoo-3_vs_1-2-4-5-6-7	0.7259 ± 0.3311	0.7218 ± 0.2847	0.7259 ± 0.3311	0.7146 ± 0.2323	**0.7508 ± 0.1933**
yeast-1-4-5-8_vs_7	0.0957 ± 0.1603	**0.6089 ± 0.0842**	0.5596 ± 0.2037	0.5518 ± 0.1970	0.5819 ± 0.1604

Continued on next page

winequality-white-4_vs_5-6	0.2301 ± 0.1015	0.7246 ± 0.0244	0.7330 ± 0.0387	0.7348 ± 0.0428	**0.7352 ± 0.0457**
yeast-2_vs_8	0.4807 ± 0.2869	0.5632 ± 0.1940	0.5644 ± 0.2363	0.5743 ± 0.2381	**0.6024 ± 0.2179**
yeast4	0.3795 ± 0.1829	0.7645 ± 0.0724	0.7897 ± 0.0867	0.7882 ± 0.0840	**0.7947 ± 0.0796**
yeast-1-2-8-9_vs_7	0.4389 ± 0.2275	0.6792 ± 0.0911	**0.6979 ± 0.0829**	0.6266 ± 0.1789	0.5749 ± 0.1740
yeast6	0.5464 ± 0.1641	0.8103 ± 0.0580	**0.8662 ± 0.0644**	0.8583 ± 0.0910	0.8650 ± 0.0693

The results in Tables 2 and 3 not only show some known conclusions, but also reveal some new interesting phenomenon. First, we all know that the imbalanced data distribution is generally harmful for traditional classification models, especially when class imbalance ratio is high, this phenomenon tends to be severer. This conclusion is verified by the results in these two tables, again. It is clear that any one DTM strategy performs better and the baseline algorithm, namely SVM or ELM. In addition, we know that in general, the optimal DTM strategies outperform to the empirical ones. This conclusion has also been verified by the results in Tables 2 and 3. Obviously, both OTHR and ODOC strategies have produced significantly better results than the THR strategy as they could adapt data distribution well. Next, we know that the ODOC is more robust than OTHR [22], which has also been confirmed in our experimental results. Specifically, in context of SVM, the ODOC yields 13 best results while the OTHR only performs best on 6 datasets, and in context of ELM, the gap between their performances seems to be less, although the ODOC still performs a little better than the OTHR.

In these two tables, we also observe some new and interesting phenomenon. First of all, the proposed CDTM algorithm improves classification performance in comparison to several other DTM algorithms. This conclusion conforms to our anticipation that compared to several other DTM algorithms, the CDTM explores more details about data distribution, and hence it could better adapt the data distribution. Specifically, in contexts of SVM and ELM, the proposed CDTM algorithm performs best on 18 and 22 datasets, respectively. Furthermore, we note that on datasets with low class imbalance ratio, the CDTM has not shown the significant superiority in comparison with the ODOC, but on those highly imbalanced datasets, it performs significantly better. We believe that it attributes to the complexity of data distribution. In general, the datasets with low class imbalance ratios mean simple data distributions, while on this kind of dataset, the learning model generated by the complex CDTM algorithm tends to be overfitting. However, the datasets with high class imbalance ratios often have complex data distributions, which are appropriate for calling CDTM algorithm. According to the experimental results observed in Tables 2 and 3, we can safely draw a conclusion, that is, the proposed CDTM algorithm is robust, as well it is more appropriate used in the scenario with highly skewed data distributions.

4.4. Significance analysis in statistics

To present a thorough comparison of the various algorithms, we also provide their statistical results. We employed the Friedman test and Holm post-hoc test [40] to differentiate the performance of the comparative algorithms throughout 40 datasets. The Friedman test, or the Friedman bidirectional Rank variance analysis, is a statistical test of homogeneity of multiple (related) samples. The Friedman test is defined as：

$$\chi^2{}_F = \frac{12P}{J(J+1)}\left(\sum_{i=1}^{J} r_i^2 - \frac{J(J+1)^2}{4}\right) \tag{7}$$

where P is the number of datasets, J is the number of algorithms, and r_i is the average ranking of the ith algorithm. In such context, the statistical analysis results obtained at the confidence level of $\alpha = 0.05$ are presented in Tables 4 and 5.

Table 4. Statistical analysis results of comparative algorithms at the confidence level of a = 0.05 in context of SVM.

Algorithms	Average Ranking	*p*	*Holm*	Hypothesis
SVM-CDTM	1.96	-	-	-
ODOC-SVM	2.21	0.4795	0.05	Not Rejected
SVM-OTHR	3.15	0.000783	0.025	Rejected
SVM-THR	3.23	0.000356	0.016667	Rejected
SVM	4.45	0	0.0125	Rejected

Table 5. Statistical analysis results of comparative algorithms at the confidence level of a = 0.05 in context of ELM.

Algorithms	Average Ranking	*p*	*Holm*	Hypothesis
ELM-CDTM	2.25	-	-	-
ODOC-ELM	2.48	0.524518	0.05	Not Rejected
ELM-OTHR	2.65	0.257899	0.025	Not Rejected
ELM-THR	3.15	0.010909	0.016667	Rejected
ELM	4.48	0	0.0125	Rejected

The statistical results in Tables 4 and 5 show that in both contexts of SVM and ELM, the proposed CDTM algorithm has acquired the lowest Friedman's average rankings. Specifically, in context of SVM, the CDTM significantly outperforms to the baseline SVM, THR and OTHR, but we cannot say that it is significantly better than ODOC though it has a lower average ranking than ODOC. As for ELM, the CDTM performs significantly better than the baseline ELM and THR, and meanwhile, it isn't significantly better than two other DTM strategies in statistics.

4.5. Comparison about running time

Finally, we compared the running time of various algorithms in contexts of SVM and ELM, respectively. The results are presented in Tables 6 and 7.

Table 6. Running time (seconds) comparison of various compared algorithms in context of SVM.

Dataset	SVM	SVM-THR	SVM-OTHR	ODOC-SVM	SVM-CDTM
pima	0.4344	0.4347	0.4526	0.4824	0.8390
saheart	0.1766	0.1772	0.1795	0.2047	0.4086
tae-3_vs_1-2	0.0164	0.0165	0.0178	0.0256	0.0592
led7digit-0-1-2_vs_3-4-5-6-7	0.0096	0.0109	0.0121	0.0339	0.1492
glass0	0.0157	0.0165	0.0170	0.0290	0.0689
tae-1_vs_2-3	0.0191	0.0195	0.0207	0.0291	0.0682
phoneme	8.5301	8.5322	9.6428	8.8926	17.7087
haberman	0.1514	0.1515	0.1618	0.1720	0.3512
winequality-red-7_vs_6	0.2951	0.2955	0.2989	0.3521	0.7689

Continued on next page

ecoli1	0.0218	0.0222	0.0238	0.0441	0.0884
winequality-white-4_7_vs_5-6	17.7151	17.7174	18.8811	18.0274	36.5093
led7digit-0-1_vs_2-3-4-5-6-7	0.0096	0.0101	0.0139	0.0353	0.1434
appendicitis	0.0059	0.0061	0.0061	0.0125	0.0277
newthyroid2	0.0034	0.0039	0.0036	0.0176	0.0287
winequality-white-4_vs_7	0.0672	0.0674	0.0676	0.1336	0.6626
glass-3-5_vs_1-2-6-7	0.0091	0.0095	0.0094	0.0233	0.0533
glass-7_vs_1-2-3-5-6	0.0039	0.0044	0.0042	0.0180	0.0352
winequality-red-7_vs_5-6	0.4551	0.4558	0.4661	0.5613	1.4914
pageblocks2345	0.9641	0.9659	1.4178	1.3315	2.7590
page-blocks0	1.0244	1.0268	1.5007	1.4053	2.9358
winequality-white-4_vs_5	0.4025	0.4031	0.4068	0.5141	1.4740
yeast-0-3-5-9_vs_7-8	0.0774	0.0778	0.0851	0.1119	0.2139
yeast-0-2-5-6_vs_3-7-8-9	0.2691	0.2697	0.2944	0.3374	0.6518
yeast-0-2-5-7-9_vs_3-6-8	0.0919	0.0923	0.1027	0.1611	0.2961
ecoli-0-4-6_vs_5	0.0039	0.0039	0.0038	0.0178	0.0434
ecoli-0-1_vs_2-3-5	0.0053	0.0056	0.0057	0.0215	0.0536
ecoli-0-2-6-7_vs_3-5	0.0042	0.0045	0.0045	0.0193	0.0458
ecoli-0-6-7_vs_5	0.0088	0.0091	0.0095	0.0317	0.0903
ecoli-0-1-4-7_vs_2-3-5-6	0.6113	0.6124	0.6318	0.7805	2.3962
winequality-white-4_vs_6	0.0052	0.0056	0.0056	0.0278	0.0585
ecoli4	0.0429	0.0435	0.0482	0.0938	0.1604
abalone9-18	0.0078	0.0083	0.0081	0.0396	0.0807
page-blocks-1-3_vs_4	0.0028	0.0032	0.0031	0.0096	0.0274
zoo-3_vs_1-2-4-5-6-7	0.0874	0.0878	0.0996	0.1369	0.2665
yeast-1-4-5-8_vs_7	1.3300	1.3315	1.3977	1.6052	4.2289
winequality-white-4_vs_5-6	0.0188	0.0191	0.0212	0.0531	0.0966
yeast-2_vs_8	0.1317	0.1325	0.1619	0.2350	0.4480
yeast4	0.1070	0.1074	0.1204	0.1760	0.3351
yeast-1-2-8-9_vs_7	0.0837	0.0846	0.0990	0.1874	0.3522
yeast6	0.4344	0.4347	0.4526	0.4824	0.8390

Table 7. Running time (seconds) comparison of various compared algorithms in context of ELM.

Dataset	ELM	ELM-THR	ELM-OTHR	ODOC-ELM	ELM-CDTM
pima	0.0296	0.0311	0.1305	0.3595	0.7476
saheart	0.0151	0.0153	0.1398	0.2111	0.3221
tae-3_vs_1-2	0.0062	0.0068	0.0068	0.0798	0.1641
led7digit-0-1-2_vs_3-4-5-6-7	0.0131	0.0136	0.0408	0.1968	0.8808
glass0	0.0075	0.0079	0.0100	0.1166	0.2208
tae-1_vs_2-3	3.2730	3.2793	14.0879	5.5173	11.0635
phoneme	0.0119	0.0115	0.0634	0.1460	0.1937
haberman	0.0361	0.0386	0.1607	0.4074	1.1732
winequality-red-7_vs_6	0.0104	0.0112	0.0154	0.1715	0.2862
ecoli1	2.7256	2.7371	8.3337	4.8558	18.0189
winequality-white-4_7_vs_5-6	0.0103	0.0112	0.0157	0.1729	0.6201
led7digit-0-1_vs_2-3-4-5-6-7	0.0046	0.0043	0.0047	0.0571	0.0754
appendicitis	0.0086	0.0097	0.0093	0.1203	0.1532
newthyroid2	0.0389	0.0412	0.1076	0.4769	2.3145
winequality-white-4_vs_7	0.0076	0.0080	0.0094	0.1120	0.1757
glass-3-5_vs_1-2-6-7	0.0080	0.0079	0.0094	0.1149	0.1508
glass-7_vs_1-2-3-5-6	0.0795	0.0827	0.3604	0.7545	2.6373
winequality-red-7_vs_5-6	3.6900	3.6981	7.6258	6.1049	10.6912
pageblocks2345	3.7078	3.7164	7.6976	6.1335	10.7285
page-blocks0	0.0929	0.0965	0.4214	0.8233	3.0324

Continued on next page

winequality-white-4_vs_5	0.0171	0.0175	0.1150	0.2442	0.4211
yeast-0-3-5-9_vs_7-8	0.0414	0.0426	0.2926	0.4917	0.6392
yeast-0-2-5-6_vs_3-7-8-9	0.0414	0.0425	0.1502	0.4855	0.6331
yeast-0-2-5-7-9_vs_3-6-8	0.0079	0.0087	0.0086	0.1153	0.1652
ecoli-0-4-6_vs_5	0.0094	0.0096	0.0107	0.1315	0.1951
ecoli-0-1_vs_2-3-5	0.0083	0.0079	0.0098	0.1196	0.1703
ecoli-0-2-6-7_vs_3-5	0.0087	0.0084	0.0104	0.1203	0.1675
ecoli-0-6-7_vs_5	0.0096	0.0101	0.0170	0.1656	0.2852
ecoli-0-1-4-7_vs_2-3-5-6	0.4672	0.4701	1.8028	1.4985	4.6396
winequality-white-4_vs_6	0.0118	0.0116	0.0161	0.1804	0.2483
ecoli4	0.0244	0.0260	0.0441	0.3440	0.4347
abalone9-18	0.0164	0.0166	0.0197	0.2449	0.3425
page-blocks-1-3_vs_4	0.0047	0.0043	0.0052	0.0549	0.0883
zoo-3_vs_1-2-4-5-6-7	0.0246	0.0251	0.1233	0.3292	0.4372
yeast-1-4-5-8_vs_7	1.6210	1.6265	4.3666	3.2834	7.9625
winequality-white-4_vs_5-6	0.0169	0.0180	0.0231	0.2357	0.3083
yeast-2_vs_8	0.0718	0.0731	0.3700	0.6931	0.8994
yeast4	0.0417	0.0437	0.0780	0.4757	0.6139
yeast-1-2-8-9_vs_7	0.0751	0.0786	0.1363	0.7290	0.9489
yeast6	0.0296	0.0311	0.1305	0.3595	0.7476

From the results in Tables 6 and 7, it is not difficult to observe that in both contexts of SVM and ELM, the baseline algorithm is most time-saving, and then the empirical DTM strategy THR is more time-saving than two other optimal DTM strategies, that is, OTHR and ODOC, while the proposed CDTM algorithm is most time-consuming among all strategies. The results conform to the results of time complexity analysis in Section 3 as the CDTM introduces an extra DBSCAN clustering procedure, and meanwhile, it trains more sub-classifiers than several other algorithms. Fortunately, the proposed CDTM algorithm only slightly increases running time compared with two other optimization idea-based DTM algorithms, and thus it can satisfy the requirements of practical applications.

5. Conclusions

In this paper, a modified decision threshold moving algorithm called CDTM was proposed for classifying imbalanced data. Specifically, the CDTM algorithm introduces DBSCAN clustering technique to deeply capture detailed information about data distribution, and further provides individual and adaptive threshold in different feature regions. It can be seen as a combination of DTM technique and divide-and-conquer method. The proposed algorithm amends the drawbacks of traditional DTM algorithms to some extent. Experimental results on 40 benchmark class imbalance datasets show that the proposed CDTM algorithm is helpful for improving classification performance than several other state-of-the-art DTM algorithms, especially on data with highly class imbalance ratios, the superiority of CDTM is more significant. Additionally, the proposed CDTM algorithm is relatively time-saving, which meets the demands of practical applications in the real-world.

In future work, we wish to develop better individual DTM algorithms with stronger data distribution exploration ability. In addition, how to make individual DTM algorithms adapt multi-class imbalance data will be investigated, too.

Acknowledgments

This study was supported in part by the National Natural Science Foundation of China under grants No. 62176107 and No. 62076111.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. Y. C. Wang, C. H. Cheng, A multiple combined method for rebalancing medical data with class imbalances, *Comput. Biol. Med.*, **134** (2021), 104527. https://doi.org/10.1016/j.compbiomed.2021.104527
2. M. Zareapoor, P. Shamsolmoali, J. Yang, Oversampling adversarial network for class-imbalanced fault diagnosis, *Mech. Syst. Signal Process.*, **149** (2021), 107175. https://doi.org/10.1016/j.ymssp.2020.107175
3. S. Fan, X. Zhang, Z. Song, Imbalanced sample selection with deep reinforcement learning for fault diagnosis, *IEEE Trans. Ind. Inf.*, **18** (2021), 2518–2527. https://doi.org/10.1109/TII.2021.3100284
4. N. Gupta, V. Jindal, P. Bedi, LIO-IDS: Handling class imbalance using LSTM and improved one-vs-one technique in intrusion detection system, *Comput. Networks*, **192** (2021), 108076. https://doi.org/10.1016/j.comnet.2021.108076
5. Z. Li, M. Huang, G. Liu, C. Jiang, A hybrid method with dynamic weighted entropy for handling the problem of class imbalance with overlap in credit card fraud detection, *Expert Syst. Appl.*, **175** (2021), 114750. https://doi.org/10.1016/j.eswa.2021.114750
6. A. G. C. de Sá, A. C. M. Pereira, G. L. Pappa, A customized classification algorithm for credit card fraud detection, *Eng. Appl. Artif. Intell.*, **72** (2018), 21–29. https://doi.org/10.1016/j.engappai.2018.03.011
7. B. Guo, C. Zhang, J. Liu, X. Ma, Improving text classification with weighted word embeddings via a multi-channel TextCNN model, *Neurocomputing*, **363** (2019), 366–374. https://doi.org/10.1016/j.neucom.2019.07.052
8. Y. Li, H. Guo, Q. Zhang, M. Gu, J. Yang, Imbalanced text sentiment classification using universal and domain-specific knowledge, *Knowl. Based Syst.*, **160** (2018), 1–15. https://doi.org/10.1016/j.knosys.2018.06.019
9. L. Dou, F. Yang, L. Xu, Q. Zou, A comprehensive review of the imbalance classification of protein post-translational modifications, *Briefings Bioinf.*, **22** (2021), bbab089. https://doi.org/10.1093/bib/bbab089
10. M. Neyestani, F. Sarmadian, A. Jafari, A. Keshavarzi, A. Sharififar, Digital mapping of soil classes using spatial extrapolation with imbalanced data, *Geoderma Reg.*, **26** (2021), e00422. https://doi.org/10.1016/j.geodrs.2021.e00422
11. S. Ketu, P. K. Mishra, Scalable kernel-based SVM classification algorithm on imbalance air quality data for proficient healthcare, *Complex Intell. Syst.*, **7** (2021), 2597–2615. https://doi.org/10.1007/s40747-021-00435-5

12. Y. S. Li, H. Chi, X. Y. Shao, M. L. Qi, B. G. Xu, A novel random forest approach for imbalance problem in crime linkage, *Knowl. Based Syst.*, **195** (2020), 105738. https://doi.org/10.1016/j.knosys.2020.105738
13. P. Soltanzadeh, M. Hashemzadeh, RCSMOTE: Range-Controlled synthetic minority over-sampling technique for handling the class imbalance problem, *Inf. Sci*, **542** (2021), 92–111. https://doi.org/10.1016/j.ins.2020.07.014
14. S. Susan, A. Kumar, The balancing trick: optimized sampling of imbalanced datasets—a brief survey of the recent State of the Art, *Eng. Rep.*, **3** (2021), e12298. https://doi.org/10.1002/eng2.12298
15. H. Guan, Y. Zhang, M. Xian, H. D. Cheng, X. Tang, SMOTE-WENN: Solving class imbalance and small sample problems by oversampling and distance scaling, *Appl. Intell.*, **51** (2021), 1394–1409. https://doi.org/10.1007/s10489-020-01852-8
16. H. Yu, C. Sun, X. Yang, S. Zheng, H. Zou, Fuzzy support vector machine with relative density information for classifying imbalanced data, *IEEE Trans. Fuzzy Syst.*, **27** (2019), 2353–2367. https://doi.org/10.1109/TFUZZ.2019.2898371
17. H. Zhang, L. Jiang, C. Li, CS-ResNet: Cost-sensitive residual convolutional neural network for PCB cosmetic defect detection, *Expert Syst. Appl.*, **185** (2021), 115673. https://doi.org/10.1016/j.eswa.2021.115673
18. W. Pei, B. Xue, L. Shang, M. Zhang, Genetic programming for development of cost-sensitive classifiers for binary high-dimensional unbalanced classification, *Appl. Soft Comput.*, **101** (2021), 106989. https://doi.org/10.1016/j.asoc.2020.106989
19. Z. H. Zhou, X. Y. Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, *IEEE Trans. Knowl. Data Eng.*, **18** (2006), 63–77. https://doi.org/10.1109/TKDE.2006.17
20. W. J. Lin, J. J. Chen, Class-imbalanced classifiers for high-dimensional data, *Briefings Bioinf.*, **14** (2012), 13–26. https://doi.org/10.1093/bib/bbs006
21. H. Yu, C. Mu, C. Sun, W. Yang, X. Yang, X. Zuo, Support vector machine-based optimized decision threshold adjustment strategy for classifying imbalanced data, *Knowl. Based Syst.*, **76** (2015), 67–78. https://doi.org/10.1016/j.knosys.2014.12.007
22. H. Yu, C. Sun, X. Yang, W. Yang, J. Shen, Y. Qi, ODOC-ELM: Optimal decision outputs compensation-based extreme learning machine for classifying imbalanced data, *Knowl. Based Syst.*, **92** (2016), 55–70. https://doi.org/10.1016/j.knosys.2015.10.012
23. K. Yang, Z. Yu, C. L. P. Chen, W. Cao, J. You, H. S. Wong, Incremental weighted ensemble broad learning system for imbalanced data, *IEEE Trans. Knowl. Data Eng.*, **34** (2021), 5809–5824. https://doi.org/10.1109/TKDE.2021.3061428
24. Z. Qi, Z. Zhang, A hybrid cost-sensitive ensemble for heart disease prediction, *BMC Med. Inf. Decis. Making*, **21** (2021), 1–18. https://doi.org/10.21203/rs.2.22946/v1
25. H. Du, Y. Zhang, K. Gang, L. Zhang, Y. C. Chen, Online ensemble learning algorithm for imbalanced data stream, *Appl. Soft Comput.*, **107** (2021), 107378. https://doi.org/10.1016/j.asoc.2021.107378
26. T. Hayashi, H. Fujita, One-class ensemble classifier for data imbalance problems, *Appl. Intell.*, **52** (2022), 17073–17089. https://doi.org/10.1007/s10489-021-02671-1

27. E. Schubert, J. Sander, M. Ester, H. P. Kriegel, X. Xu, DBSCAN revisited, revisited: why and how you should (still) use DBSCAN, *ACM Trans. Database Syst.*, **42** (2017), 1–21. https://doi.org/10.1145/3068335
28. T. N. Tran, K. Drab, M. Daszykowski, Revised DBSCAN algorithm to cluster data with dense adjacent clusters, *Chemom. Intell. Lab. Syst.*, **120** (2013), 92–96. https://doi.org/10.1016/j.chemolab.2012.11.006
29. D. Birant, A. Kut, ST-DBSCAN: An algorithm for clustering spatial–temporal data, *Data Knowl. Eng.*, **60** (2007), 208–221. https://doi.org/10.1016/j.datak.2006.01.013
30. M. Ontivero-Ortega, A. Lage-Castellanos, G. Valente, R. Goebel, M. Valdes-Sosa, Fast Gaussian Naïve Bayes for searchlight classification analysis, *Neuroimage*, **163** (2017), 471–479. https://doi.org/10.1016/j.neuroimage.2017.09.001
31. R. D. Raizada, Y. S. Lee, Smoothness without smoothing: why Gaussian Naive Bayes is not naive for multi-subject searchlight studies, *PloS One*, **8** (2013), e69566. https://doi.org/10.1371/journal.pone.0069566
32. W. S. Noble, What is a support vector machine? *Nat. Biotechnol.*, **24** (2006), 1565–1567. https://doi.org/10.1038/nbt1206-1565
33. C. C. Chang, C. J. Lin, LIBSVM: A library for support vector machines, *ACM Trans. Intell. Syst. Technol.*, **2** (2011), 1–27. https://doi.org/10.1145/1961189.1961199
34. G. B. Huang, Q. Y. Zhu, C. K. Siew, Extreme learning machine: theory and applications, *Neurocomputing*, **70** (2006), 489–501. https://doi.org/10.1016/j.neucom.2005.12.126
35. G. B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, **42** (2011), 513–529. https://doi.org/10.1109/tsmcb.2011.2168604
36. H. Yu, J. Ni, S. Xu, B. Qin, H. Jv, Estimating harmfulness of class imbalance by scatter matrix based class separability measure, *Intell. Data Anal.*, **18** (2014), 203–216. https://doi.org/10.3233/IDA-140637
37. P. E. Gill, W. Murray, M. H. Wright, *Practical Optimization*, Academic Press, London, 1981. https://doi.org/10.2307/3616583
38. H. Guo, H. Liu, C. Wu, W. Zhi, Y. Xiao, W. She, Logistic discrimination based on G-mean and F-measure for imbalanced problem, *J. Intell. Fuzzy Syst.*, **31** (2016), 1155–1166. https://doi.org/10.3233/ifs-162150
39. I. Triguero, S. González, J. M. Moyano, S. G. López, J. A. Fernández, J. L. Martín, KEEL 3.0: an open source software for multi-stage analysis in data mining, *Int. J. Comput. Intell. Syst.*, **10** (2017), 1238–1249. https://doi.org/10.2991/ijcis.10.1.82
40. J. Demšar, Statistical comparisons of classifiers over multiple data sets, *J. Mach. Learn. Res.*, **7** (2006), 1–30. Available from: https://www.jmlr.org/papers/volume7/demsar06a/demsar06a.pdf.

AIMS Press