A Novel Model for Imbalanced Data Classification

Jian Yin,[1,4] **Chunjing Gan,**[1,4] **Kaiqi Zhao,**[2] **Xuan Lin,**[3] **Zhe Quan,**[3] **Zhi-Jie Wang**[1,4,5,*]
[1]School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China
[2]School of Computer Science, University of Auckland, Auckland, New Zealand
[3]College of Information Science and Engineering, Hunan University, Changsha, China
[4]Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
[5]National Engineering Laboratory for Big Data Analysis and Applications, Beijing, China
[*]Corresponding author: wangzhij5@mail.sysu.edu.cn

Abstract

Recently, imbalanced data classification has received much attention due to its wide applications. In the literature, existing researches have attempted to improve the classification performance by considering various factors such as the imbalanced distribution, cost-sensitive learning, data space improvement, and ensemble learning. Nevertheless, most of the existing methods focus on only part of these main aspects/factors. In this work, we propose a novel imbalanced data classification model that considers all these main aspects. To evaluate the performance of our proposed model, we have conducted experiments based on 14 public datasets. The results show that our model outperforms the state-of-the-art methods in terms of recall, G-mean, F-measure and AUC.

Introduction

In recent years, binary classification for *imbalanced data* has received much attention (Castro and Braga 2013; Lin et al.) due to its wide applications in various domains such as mortality prediction, and so on (Zhao et al. 2014; Bhattacharya, Rajan, and Shrivastava 2017; Liu et al. 2018; Nie et al. 2019; Liu et al. 2019; Lin et al. 2019; Xu et al.). A main characteristic of this problem is that, most of data samples belong to one class while the rest belong to the other. Typically, the class with most of samples is known as the *majority class*, while the other is known as the *minority class*, which is often of significant value (Weiss 2004; Alam et al. 2018; Quan et al. 2019).

Previous studies (Seiffert et al. 2010; Alam et al. 2018) have shown that conventional machine learning methods may fail to address the imbalanced data classification problem. To alleviate this, a lot of methods have been proposed (Barua et al. 2014; Castro and Braga 2013; Alam et al. 2018). Generally, the existing methods can be divided into three categories: sampling methods (including undersampling and oversampling methods) (Chawla et al. 2002; Liu, Wu, and Zhou 2009; Barua et al. 2014), cost-sensitive learning methods (Thai-Nghe, Gantner, and Schmidt-Thieme 2010; Castro and Braga 2013), and ensemble learning methods (Seiffert et al. 2010; Alam et al. 2018; Fang et al. 2019). Yet, there remains several issues for each of these categories. For example: (i) Undersampling methods remove samples from the majority class at the risk of losing important samples. Oversampling methods replicate minority samples for several times, and thus they often overfit the data (He and Garcia 2008). (ii) For cost-sensitive learning methods, it is hard to obtain misclassification cost parameter for minority and majority classes (Dumpala, Chakraborty, and Kopparapu 2018). (iii) For ensemble learning methods, the majority voting is commonly adopted, which may result in biased results due to the existence of "unstable samples" (Wu et al. 2017). Unstable samples have even chance of being assigned to each class label by different classifiers. Apart from the three categories, recent studies (Wang et al. 2018) indicate that, for imbalanced datasets, the limited training samples and their surrounding data space can improve the classification results.

In this paper, we propose a novel model to address the above limitations for the imbalanced binary classification problem. Our model integrates *sampling*, *data space construction*, *cost-sensitive learning*, and *ensemble learning* in a principle way. Specifically, it consists of four main components: (i) Data block construction component, which aims to divide the datasets into balanced data blocks using both undersampling and oversampling. (ii) Data space improvement component, which transforms the data space so that the data samples can be close to its k nearest neighbors with the same label, and separate the data samples from the other class by a large margin. (iii) Adaptive weight adjustment component, which obtains the class-wise weight for the ensemble learning component, so as to alleviate the trouble incurred by the "unstable samples". (iv) Ensemble learning components, which combines multiple base classifiers and obtains the final results by weighted voting. To sum up, we make two main contributions:

- We propose a novel model for imbalanced data classification. To the best of our knowledge, this is the first attempt to integrate the above four aspects together.
- We conduct extensive experiments to evaluate the performance of our model. The results consistently show that the proposed model achieves competitive performance compared against several state-of-the-art models.

Next, we review previous works most related to ours.

Related Work

As mentioned before, previous works can be generally classified into three categories: (i) sampling methods, (ii) cost-sensitive learning methods, and (iii) ensemble learning methods. In what follows, we review them briefly.

Sampling Methods

Sampling methods (Das, Krishnan, and Cook 2015) alleviate the class imbalance by adjusting the dataset, and so they are essentially a kind of data preprocessing methods. Particularly, they can be further categorized into two types: undersampling and oversampling. Undersampling is to remove majority class samples from the data, so that the degree of imbalance could be adjusted. The mainstream undersampling methods are *random undersampling* (He and Garcia 2008) and *informed undersampling* (Liu, Wu, and Zhou 2009). Random undersampling removes a set of majority samples from the original data in a random manner (He and Garcia 2008). This method is the simplest mechanism for data distribution adjustment. However, it often incurs significant information loss (Liu, Wu, and Zhou 2009). To alleviate the dilemma, *informed undersampling* methods (Liu, Wu, and Zhou 2009) combine other techniques such as training multiple classifiers to adjust the imbalanced data distribution. Oversampling duplicates or generates minority samples to compensate the lack of minority samples. The mainstream oversampling methods include *random oversampling* and *synthetic oversampling*. Random oversampling randomly replicates a set of minority samples from the original data. This sampling mechanism often overfits the data (Mease, Wyner, and Buja 2007). As for the synthetic oversampling, it generates synthetic minority samples using the existing samples. SMOTE (Chawla et al. 2002) and MWMOTE (Barua et al. 2014) are representative synthetic oversampling methods.

Compared to the works in this branch, our method utilizes both random undersampling and random oversampling. Specifically, we apply oversampling in the overall data block construction phase to replicate the minority samples. However, in each block the minority samples appear for just once, and so the overfitting phenomenon is alleviated to some extent. In addition, we use undersampling in the single data block construction phase to include a part of majority samples in each data block. Overall, our method absorbs the advantages of undersampling and oversampling, alleviating information loss and overfitting.

Cost-Sensitive Methods

Cost-sensitive methods improve the classifier by applying different costs for misclassifying different samples (Castro and Braga 2013; Krawczyk 2016). For example, Thai *et al.* (Thai-Nghe, Gantner, and Schmidt-Thieme 2010) propose sampling techniques with cost-sensitive learning method in support vector machine (SVM), and use cost-sensitive learning technique to optimize the cost ratio. Castro *et al.* (Castro and Braga 2013) incorporate prior knowledge into the cost parameter, and apply such a parameter to a joint objective function to enhance imbalanced data classification.

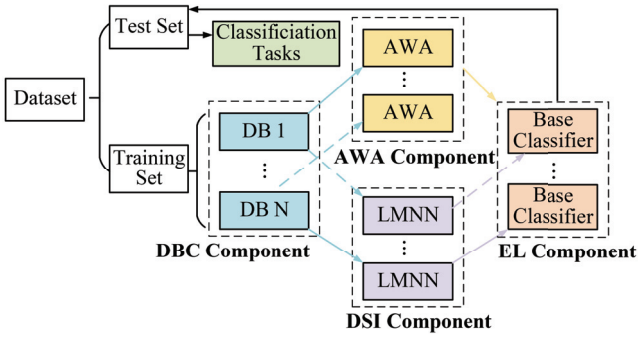


Figure 1: The architecture of our model.

Nikolaou *et al.* (Nikolaou et al. 2016) incorporate the shifted decision threshold and calibrated probability estimation for cost-sensitive learning in imbalanced data classification. Wu *et al.* (Wu et al. 2017) use cost-sensitive multi-set feature learning to learn discriminant features. Compared to the works in this branch, our method does not directly utilize different costs in the classification process. Instead, we use the cost for the "overall gain" calculation (detailed later in the paper). This allows us to obtain a weighting mechanism which maximizes the overall gain.

Ensemble Learning Methods

Ensemble learning methods utilize multiple weak classifiers, and combine them to obtain a strong classifier for classification (Hastie, Tibshirani, and Friedman 2009). There are also many ensemble learning methods for imbalanced data classification. For example, Seiffert *et al.* (Seiffert et al. 2010) propose a method that combines sampling techniques with boosting algorithm for imbalanced data classification. Alam *et al.* (Alam et al. 2018) use partitioning techniques to create balanced data, and then apply an ensemble classifier for classification and regression tasks. Compared to the works in this branch, instead of assigning equal importance to each classifier, our method incorporates class-wise weight into the ensemble learning framework, which allows us to handle incorrectly labelled data samples.

Besides conventional machine learning methods, there emerges several imbalanced classification methods based on deep learning (Khan et al. 2017; Huang et al. 2016). However, they focus on image data. Unlike image datasets, many imbalanced datasets are limited in size and features, and thus deep learning methods are not applicable in our setting.

Imbalance Data Classification Model

In this section, we first cover the architecture of our model, and then elaborate and discuss each component.

Overview

Figure 1 depicts the architecture of our proposed model named *DDAE*. As mentioned earlier, it contains four main components: (i) Data Block Construction (DBC) component, which is responsible for dividing the input data into nearly balanced data blocks; (ii) Data Space Improvement

(DSI) component, which is used to make the k nearest neighbors belong to the same class and separate the data samples from different classes by a large margin; (iii) Adaptive Weight Adjusting (AWA) component, which is used to adjust the weights of samples from different classes; and (iv) Ensemble Learning (EL) component, which is used to combine multiple base classifiers by weighted voting.

The work flow of our method is as follows. Generally, we partition a dataset into a training set and a test set. The first step is to put the data from the training set into the DBC component, obtaining a number of balanced data blocks. Then, each data block is fed into the DSI component for improving the data space. Meanwhile, the AWA component computes the class-wise weights based on the data in each block. After that, we apply a base classifier (kNN) to each data block processed by DSI component, and the EL component collaboratively use weights from AWA and the base classifiers to generate the class labels. Finally, the trained model is applied to the test set to perform the classification tasks. Next, we present each component in detail.

Data Block Construction

We divide the data into multiple data blocks, so as to make each data block balanced. In a nutshell, our method absorbs the advantages of undersampling and oversamping methods, alleviating information loss and overfitting to some extent. Specifically, we partition the majority samples into several subsets, each roughly of the same size as the set of minority samples. Then, we use oversampling to replicate the minority samples and combine each copy with each subset of majority samples. As we only use part of the majority samples in each data block, it can be considered as the undersampling of the majority samples.

Formally, let N_{min} (resp., N_{maj}) be the number of minority (resp., majority) samples. Let S_{min} (resp., S_{maj}) be the set of minority (resp., majority) samples. Let $\delta = N_{maj}/N_{min}$ be the imbalanced ratio. Algorithm 1 shows the pseudocodes of data block construction process. In this algorithm, δ^* can be computed by $\lfloor\delta\rfloor$ or $\lceil\delta\rceil$.

Algorithm 1: Data blocks construction

Input: Dataset D
Output: A set B of δ^* data blocks
1: Obtain N_{min}, N_{maj}, S_{min}, and S_{maj} based on D;
2: Divide S_{maj} into δ^* chunks $\{C_1, C_2, ..., C_{\delta^*}\}$;
3: **for** $i = 1$ to δ^* **do**
4: Put C_i and S_{min} into an empty data block B_i;
5: **end for**
6: **return** $B = \{B_1, B_2, ...\}$

Data Space Improvement

To improve the data space, our model incorporates the metric learning technique. Specifically, we adopt the *large margin nearest neighbor* (LMNN) algorithm (Weinberger and Saul 2009). This algorithm learns a transformation matrix L and incorporates a loss function $\varphi(L)$, so that it can push data samples of different classes away from the target sample, and pull data samples that have the same class label close to the target sample. Specifically, the loss function is as follows. $\varphi(L) = (1-\lambda)\varphi_{pull}(L)+\lambda\varphi_{push}(L)$ where λ is a positive real number used as the weight for these two items (i.e., pull and push). The first item $\varphi_{pull}(L)$ is computed as $\varphi_{pull}(L) = \sum_{i,j\in N(i)} \|L(x_i - x_j)\|^2$, where $N(i)$ is the k nearest neighbor of data i with the same class label as i. It penalizes large distances between the data sample and its k nearest neighbors with the same class label. The second item penalizes small distances between the data sample and others with different class label. It is computed as $\varphi_{push}(L) = \sum_{i,j,l}(1-\delta_{il})max\{0, 1+\|L(s_i - s_j)\|^2 - \|L(s_i - s_l)\|^2\}$, where δ_{il} is used to determine whether data samples s_i and s_l belong to the same class or not. More specifically, if they are from the same class, then $\delta_{il} = 1$; otherwise, $\delta_{il} = 0$.

Adaptive Weight Adjustment

We first introduce some concepts and then present our adaptive weight adjustment (AWA) component in detail. For each data sample s, its k nearest neighbors may be positive or negative. When applying kNN classifiers, the label of s may be ambiguous if the number of its positive and negative neighbors are roughly the same. More formally, we refer to the absolute difference between the number of negative neighbors and that of positive neighbors as positive-negative count difference (PNCD). If the PNCD is large than a threshold τ (it is conservatively set to 1 when k is odd; otherwise, it is set to 2 in our experiments), we refer to such a sample s as the stable sample. Otherwise, we refer to it as the unstable sample.

Most of the ensemble learning methods assign the same weight to all the classifiers, and obtain the final result by majority voting (Zhang and Ma 2012). Such a method may incur biased results, due to the existence of unstable samples. To address this issue, we suggest an adaptive weight adjustment mechanism to obtain better classification results. In particular, we first introduce the *unstable confusion matrix*, as shown in Table 1. In the matrix, $c_{1,0}$ (resp., $c_{1,1}$) denotes the number of unstable positive samples which are predicted as negative (resp., positive). In contrast, $c_{0,0}$ (resp., $c_{0,1}$) denotes the number of unstable negative samples which are predicted as negative (resp., positive). Let S_{pos} (resp., S_{neg}) be the set of samples whose real labels are positive (resp., negative). Let S_{grt} (resp., S_{sml}) be the number of unstable samples whose positive predictions is greater (resp., smaller) than negative predictions. One can obtain the values in the matrix as follows: $c_{1,0} = |S_{pos} \cap S_{sml}|$; $c_{1,1} = |S_{pos} \cap S_{grt}|$; $c_{0,0} = |S_{neg} \cap S_{sml}|$; and $c_{0,1} = |S_{neg} \cap S_{grt}|$.

Table 1: Unstable confusion matrix

Sample	Predict as negative	Predict as positive
Positive	$c_{1,0}$	$c_{1,1}$
Negative	$c_{0,0}$	$c_{0,1}$

In imbalanced data classification, it is usual that the cost (or importance) of the minority sample is larger than that of the majority sample. Without loss of generality, we assume the cost ratio between minority samples and majority samples is x. The basic idea of AWA is to adjust the weights for the positive and negative outputs, and to maximize the overall gain for the unstable confusion matrix.

In the initial stage, the default weight, denoted by W_d is set to 1 in this paper. That is, we set the weight pair (1,1) as the weights for negative and positive outputs, respectively. Then, we attempt to adjust the weight of positive (resp., negative) outputs. Let $gain_{mat}$ be the overall gain when we maintain the default weight. Let $gain_{pos}$ (resp., $gain_{neg}$) be the overall gain if we increase the weight of positive (resp., negative) output. We compute these gains as follows.

$$\begin{cases} gain_{mat} &= x*(c_{1,1}-c_{1,0})+(c_{0,0}-c_{0,1}) \\ gain_{pos} &= x*(c_{1,1}+c_{1,0})+(-c_{0,0}-c_{0,1}) \\ gain_{neg} &= x*(-c_{1,1}-c_{1,0})+(c_{0,0}+c_{0,1}) \end{cases} \quad (1)$$

After that, we adjust the weight according to the maximum gain. If the maximum gain is $gain_{mat}$, then we set the new weight $W_n = W_d$ for both negative and positive outputs. Otherwise, we choose the maximum gain $gain_{pos}$ (or $gain_{neg}$), and update its corresponding weight weight to $W_n = W_t + \Delta$, where Δ is a small positive real number, and W_t is the weight threshold, which is computed as

$$W_t = \begin{cases} (\frac{\delta^*}{2}+1)/(\frac{\delta^*}{2}-1), & if \quad \delta^* \bmod 2 = 0 \\ (\lfloor\frac{\delta^*}{2}\rfloor+1)/(\lfloor\frac{\delta^*}{2}\rfloor), & otherwise \end{cases} \quad (2)$$

For example, if $gain_{pos}$ is the maximum one, the weight pair is updated to (W_d, W_n), and vice versa.

Finally, given the weight pairs obtained from all the data blocks, the overall weight pair, denoted by $W_o = (W_o^0, W_o^1)$, is obtained based on the frequency of different weight pairs. Particularly, if the ratio between the number of non-default weight pairs and the number of all weight pairs is smaller than a threshold γ (it is empirically set to 0.2 in the experiments), then we set $W_o = (W_d, W_d)$. Otherwise, we further determine the number of "negative" non-default weight pairs, which is in the form of (W_n, W_d), and the number of "positive" non-default weight pairs, which is in the form of (W_d, W_n). If the former is larger than the latter, we set $W_o = (W_n, W_d)$; otherwise, we set $W_o = (W_d, W_n)$. Note that, we set $W_o = (W_d, W_n)$ when those two numbers are equal, since the cost of minority (i.e., positive) samples is larger than that of the majority (i.e., negative) samples.

Ensemble Learning

In ensemble learning, multiple classifiers together with the voting mechanism are usually used to obtain the final prediction results. As for binary classification problem, assume that there are m classifiers $\{f_0, f_2, ..., f_{m-1}\}$, and 2 classes $\{c_0, c_1\}$. If the ith classifier whose output is c_j for sample s, then ${f_i}^j(s) = 1$, where $i \in \{0, 1, ..., m-1\}$ and $j \in \{0, 1\}$; otherwise, it is 0. The voting mechanism can be described as follows.

$$F(s) = \begin{cases} c_0, & \sum_{i=0}^{m-1} {f_i}^0(s) \geq \sum_{i=0}^{m-1} {f_i}^1(s) \\ c_1, & otherwise \end{cases} \quad (3)$$

Unlike most of existing methods, which just count the number of positive and negative labels, we use the weight W_o obtained in the AWA component to enhance the final results. Specifically, our voting mechanism is as follows.

$$F(s) = \begin{cases} c_0, & W_o^0 * \sum_{i=0}^{m-1} {f_i}^0(s) \geq W_o^1 * \sum_{i=0}^{m-1} {f_i}^1(s) \\ c_1, & otherwise \end{cases} \quad (4)$$

Experiments

Experimental Setup

► *Datasets.* In our experiments, we employ 14 widely used datasets. Among them, six datasets (including cm1, kc3, mw1, pc1, pc3, pc4) are from OpenML (Vanschoren et al. 2013), and they are open datasets for software defect detection. The other eight datasets (including yeast1vs7, abalone9vs18, yeast6, abalone19, poker89vs6, wine3vs5, abalone20, and poker8vs6) are from KEEL repository (Fernández, del Jesus, and Herrera 2009). Among them, Yeast datasets are often used for predicting cellular localization sites of proteins. The abalone datasets are used for predicting the age of abalone. The two poker datasets are used for poker hands prediction while the wine dataset is used for wine quality prediction.

All the 14 datasets have various characteristics in terms of instances, features and IR (Imbalance Ratio). The detailed descriptions of these datasets are summarized in Table 2. In addition, similar to previous works, for all experiments we randomly split datasets into two parts: training set (70%) and test set (30%).

Table 2: Summary of the datasets. Note that, the names of the last two datasets are somewhat long, we use superscripts to mark them, where [a] is short for the "winequalityred3vs5" dataset, and [b] is short for the "abalone20vs8910" dataset.

Datasets	# instances	# features	IR
cm1	497	21	9.354
kc3	458	39	9.651
mw1	403	37	12
pc1	1109	21	13.4
pc3	1563	37	8.769
pc4	1458	37	7.191
yeast1vs7	459	7	14.3
abalone9vs18	731	8	16.405
yeast6	1484	8	41.4
abalone19	4174	8	129.438
poker89vs6	1485	10	58.4
poker8vs6	1477	10	85.882
wine3vs5[a]	691	11	68.1
abalone20[b]	1916	8	72.692

► *Baselines.* To evaluate the performance of our method, we compare it with the following state-of-the-art methods.

Table 3: G-mean, F-measure and AUC for IML, RP, CAdaMEC, MWMOTE and Our method (DDAE) on the 14 public datasets. Note that, NA denotes invalid result.

Dataset \ Method	IML			RP			CAdaMEC			MWMOTE			**DDAE**		
	G-mean	F-ms	AUC	G-mean	F-ms	AUC	G-mean	F-ms	AUC	G-mean	F-ms	AUC	G-mean	F-ms	AUC
cm1	0.52	0.287	0.589	0.758	0.567	0.762	0.591	0.38	0.654	0.623	0.407	0.663	**0.775**	**0.58**	**0.776**
kc3	0.805	**0.652**	0.814	0.81	0.604	0.811	0.772	0.625	0.792	0.749	0.563	0.764	**0.823**	0.625	**0.823**
mw1	0.635	0.345	0.653	0.701	0.39	0.702	0.721	0.446	0.728	0.721	0.446	0.728	**0.815**	**0.588**	**0.817**
pc1	0.657	0.408	0.679	0.803	0.556	0.807	0.7	0.504	0.731	0.767	**0.586**	0.782	**0.819**	0.573	**0.83**
pc3	0.578	0.342	0.582	0.726	0.513	0.726	0.721	0.519	0.731	0.646	0.426	0.671	**0.743**	**0.536**	**0.744**
pc4	0.725	0.574	0.73	**0.873**	**0.767**	**0.873**	0.826	0.699	0.828	0.772	0.621	0.778	0.804	0.676	0.813
yeast1vs7	0.716	0.471	0.718	0.701	0.449	0.702	0.755	0.603	0.78	0.745	0.574	0.768	**0.841**	**0.649**	**0.841**
abalone9vs18	0.709	0.375	0.719	0.695	0.345	0.702	0.697	0.49	0.736	0.7	0.51	0.74	**0.814**	**0.603**	**0.824**
yeast6	0.798	0.407	0.805	0.779	0.333	0.783	0.703	0.5	0.744	0.829	**0.636**	0.841	**0.883**	0.421	**0.883**
abalone19	0.626	0.037	0.628	0.771	0.065	0.773	0	NA	0.5	0.407	**0.143**	0.579	**0.839**	0.075	**0.852**
wine3vs5	0	NA	0.5	0.51	0.086	0.557	0	NA	0.5	0	NA	0.49	**0.55**	**0.156**	**0.62**
abalone20	0.802	0.252	0.802	0.898	0.314	0.904	0.628	0.385	0.693	0.446	0.227	0.598	**0.964**	**0.556**	**0.965**
poker89vs6	0.783	0.317	0.794	0.633	0.132	0.651	0.913	**0.862**	0.917	0.816	0.714	0.833	**0.968**	0.517	**0.968**
poker8vs6	0.615	0.08	0.628	0.764	0.128	0.764	0.707	**0.556**	0.75	0.707	**0.556**	0.75	**0.95**	0.317	**0.951**

- **IML** (Wang et al. 2018): It constructs a stable neighborhood data space by using the iterative metric learning technique, and it chooses the k-nearest neighbors (kNN) algorithm as the classifier.
- **RP** (Alam et al. 2018): It is an recursive based ensemble method for imbalanced data classification problem. It converts the imbalanced data classification problem into a variety of balanced data classification problems. In the meanwhile, it uses the majority voting as the ensemble rule to build an ensemble classifier.
- **CAdaMEC** (Nikolaou et al. 2016): It is based on a cost-sensitive learning method (AdaMEC (Ting 2000; Nikolaou and Brown 2015)) with proper calibration, and it uses decision tree as the classifier.
- **MWMOTE** (Barua et al. 2014): It is a synthetic oversampling method for imbalanced data classification problem. Each sample from the minority class is assigned a weight according to its Euclidean distance to the nearest majority class sample. Then, it generates synthetic samples from the weighted minority class.

► *Evaluation metrics.* For the imbalanced data classification problem, two widely accepted common senses for a good model are:

1. It should achieve high performance on comprehensive accuracy metrics (e.g., G-mean, F-measure, AUC) that consider samples from both the majority and the minority classes.
2. Without harming the comprehensive metrics, it is essential for the model to correctly predict the labels for as many samples from the minority class as possible. This is because applications such as software defect detection, the minority class (e.g., software defect) is considered to be much more important than the majority class.

A common setting in imbalanced classification is that the positive samples often refer to the minority class samples while the negative samples refer to the majority class (He and Garcia 2008). Based on this setting and the above common senses, we adopt the following metrics: $Recall$, $G\text{-}mean$, $F_\beta\text{-}measure$ and AUC, as do the prior works (He and Garcia 2008; Wang et al. 2018; Alam et al. 2018; Nikolaou et al. 2016). Let TP, FN, FP, TN be the true positives, false negatives, false positives and true negatives, respectively. The above four metrics are defined as follows: (i) $Recall = \frac{TP}{TP+FN}$, is motivated by the second common sense and measures how many positive (minority) samples are correctly classified. Another explanation of recall is the probability of detection for positive (minority) samples (Wang et al. 2018). In this paper, we use (ii) $G\text{-}mean = \sqrt{Recall * TNR}$, which combines the recall for both classes. Here, $TNR = \frac{TN}{TN+FP}$ denotes the recall for the negative (majority) class. (iii) $F_\beta - measure = (1+\beta^2)\frac{Precision*Recall}{\beta^2*Precision+Recall}$, which considers both precision ($\frac{TP}{TP+FP}$) and recall, and thus is a comprehensive metric of classification accuracy. Here, β trades precision for recall. The higher the β is, the more important the $Recall$ is. Because it is often more desirable to correctly classified the minority class in the imbalanced classification problem, we set $\beta = 2$. For simplicity, we use F-ms in the following experiments to denote $F_\beta - measure$ with $\beta = 2$. (iv) AUC refers to the area under the ROC curve. In the ROC curve, the x-axis corresponds to the false positive rate while the y-axis corresponds to the true positive rate. One attractive property of AUC is that it is not sensitive to class distributions (Fawcett 2004), thus it is appropriate for the imbalanced classification setting.

► *Parameter settings.* Since both IML (Wang et al. 2018) and MWMOTE (Barua et al. 2014) use kNN classifiers, we vary k from 1 to 10, and we adopt the values of k that achieve the best performance. Other parameters are the same as that in the original papers (Wang et al. 2018; Barua et al. 2014). For our model, we tune each parameter by fixing the others. Specifically, we vary λ within the range of $[0.1, 1]$ and select

the values that lead to the best performance. Similarly, the number of blocks can be either $\lfloor IR \rfloor$ or $\lceil IR \rceil$ and we set the value that achieves the best performance. Besides, we set γ to 0.2 for all datasets and the cost ratio to $\lceil IR \rceil$.

Overall Comparison against State-Of-The-Arts

We first report the performance of all the methods in terms of comprehensive accuracy metrics in Table 3. In general, our proposed method DDAE outperforms the state-of-the-art methods on most of the 14 datasets. Especially in half of the datasets (cm1, mw1, pc3, yeast1vs7, abalone9vs18, wine3vs5, abalone20), DDAE consistently performs better than its competitors. It improves the G-mean, F_2 and AUC by up to 18.6%, 17.1%, 18.7%, respectively. The worst case for our model is obtained on the pc4 dataset, in which the imbalance ratio of pc4 is the smallest among all the datasets as shown in Table 2. This phenomenon implies that our proposed method tends to find out as many positive (minority) samples as possible, and works better on highly skewed class distributions. Nevertheless, the DDAE is still compatible – it achieves close G-mean, F_2 and AUC scores to the best baselines and outperforms two of the other baselines. It is also worth noting that, on some of the datasets such as wine3vs5, the number of positive samples are extremely small. IML, CAdaMEC and MWMOTE classify all the samples as negative and fail to detect any of the positive samples. This results in a 0 G-mean and an invalid F-measure.

In order to showcase the capability of our model on detecting positive samples, we report the recall for all the methods on the 14 datasets in Table 4. DDAE performs the best in all cases compared to the baselines. This essentially demonstrates that our method is much more effective for identifying positive samples. Especially, on several datasets (e.g., abalone19, poker89vs6), DDAE achieves 100% recall, which significantly outperforms the competitors. Although some of the baselines (e.g., RP, CAdaMEC) perform well in the comprehensive accuracy metrics, they are not able to identify positive samples correctly.

Combining the observation from Table 3 and Table 4, the proposed method improves the recall significantly while maintaining competible performance in terms of comprehensive metrics such as G-mean, F_2 and AUC. In many of the cases, it achieves even higher G-mean and AUC than the state-of-the-art methods.

Table 4: Recall for IML, RP, CAdaMEC, MWMOTE and Our method (DDAE) on the 14 public datasets.

Datasets	IML	RP	CAdaMEC	MWMOTE	DDAE
cm1	0.313	0.688	0.375	0.438	**0.813**
kc3	0.692	**0.846**	0.615	0.615	**0.846**
mw1	0.5	**0.75**	0.625	0.625	**0.75**
pc1	0.852	0.889	0.519	0.63	**0.963**
pc3	0.51	**0.735**	0.612	0.49	**0.735**
pc4	0.814	0.881	0.78	0.678	**0.932**
yeast1vs7	0.667	0.667	0.583	0.583	**0.833**
abalone9vs18	0.6	0.6	0.5	0.5	**0.7**
yeast6	0.7	0.7	0.5	0.7	**0.9**
abalone19	0.667	0.833	0	0.167	**1**
wine3vs5	0	**0.333**	0	0	**0.333**
abalone20	0.8	**1**	0.4	0.2	**1**
poker89vs6	0.667	0.5	0.833	0.667	**1**
poker8vs6	0.5	0.75	0.5	0.5	**1**

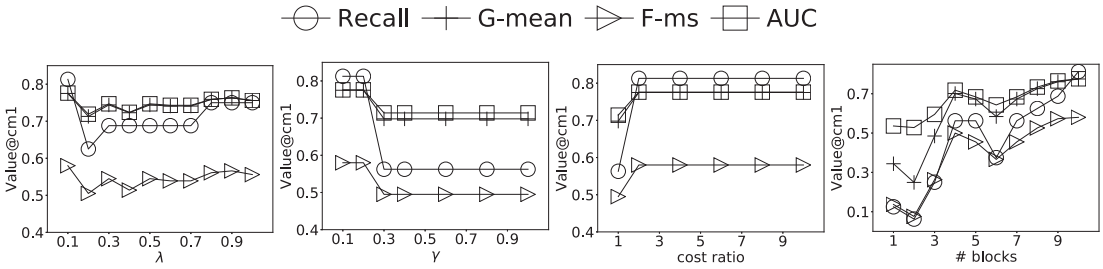
Impact of parameters

Impact of λ In the loss function $\phi(L)$, the parameter λ is used to determine the relative weight between the pull and push terms. It is important to investigate the impact of λ. To achieve this, we plot the results by varying λ from 0.1 to 1 on the cm1 and mw1 datasets in Figure 2(a) and 3(a). In general, the curves of different metrics are in similar trends in the same dataset, but the model acts differently on different datasets, i.e., it achieves the best performance when $\lambda = 0.1$ and $\lambda = 0.3$ on cm1 and mw1, respectively. This is because the data distribution of different datasets are quite different. If the samples from the two classes are highly overlapped in the original feature space, pushing the two classes away from each other is of great importance, i.e., we should set λ at a large value. In contrast, we should decrease λ to make samples from the same class closer to each other when many of the minority samples do not fall in the area dominated by the majority class.

Impact of γ The parameter γ is used as a threshold term to show the unstable ratio in the data. This parameter can be used to decide whether we adopt the AWA component or not. In this experiment, we vary γ from 0.1 to 1 on the cm1 and mw1 datasets. As we can see in Figure 2(b) and 3(b), the performance of the model becomes stable as γ is above 0.3, and $\gamma = 0.2$ achieves the best performance in both datasets. For the cm1 dataset, when γ is above 0.2, recall, G-mean, F-measure and AUC drop by 0.25, 0.077, 0.085 and 0.062, respectively. For mw1 dataset, although recall increases by moving γ from 0.2 to 0.3, it sacrifices the F-measure by around 0.058. If we take the overall performance into consideration, the best choice is $\gamma = 0.2$.

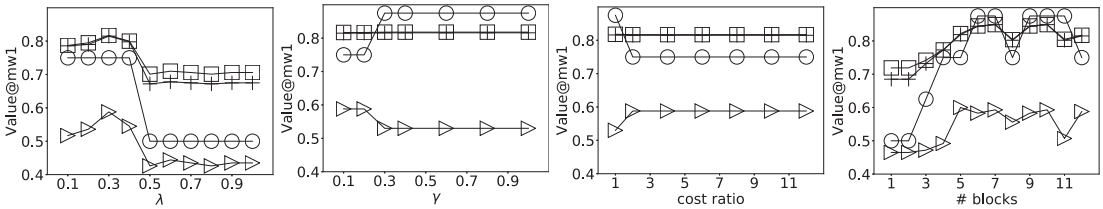
Impact of cost ratio In cost-sensitive learning methods, how to choose an appropriate cost ratio between the two classes remains a problem without any additional information. The cost ratio depicts the importance of the costs between the two classes. For example, when the cost ratio is set to 1, we treat the cost of samples from both classes are of the same importance. Therefore, assigning an arbitrary cost ratio to the data samples is controversial. However, as we can see in 2(c) and 3(c), due to our proposed weighting scheme the final results are insensitive to the choice of cost ratio when it is in a reasonable range, e.g., from 2 to the dataset's imbalance ratio. This experiment implies that our weighting scheme enhances the flexibility in choosing the weight without any prior information for the data.

Impact of # blocks In this part, we conduct experiments to illustrate the impact of the number of blocks on the cm1 and mw1 datasets. Blocks are used for adjusting the imbalanced data distribution. As we can see from figures 2(d) and 3(d), as the number of blocks grows, the four metrics share the same trend. Notably, in both datasets, the model achieves good performance when the number of blocks is close to



(a) Impact of λ (b) Impact of γ (c) Impact of cost ratio (d) Impact of # blocks

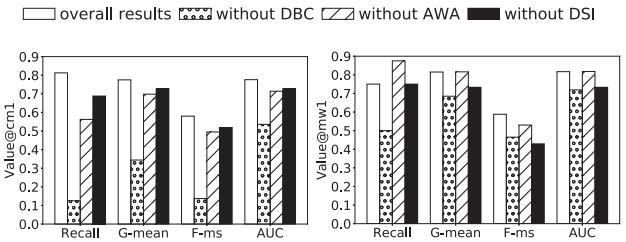
Figure 2: Impact of parameters on cm1.



(a) Impact of λ (b) Impact of γ (c) Impact of cost ratio (d) Impact of # blocks

Figure 3: Impact of parameters on mw1.

the imbalance ratio (10 for cm1, 12 for mw1). In general, with a small number of blocks, samples in each block are still imbalanced, and thus result in significant performance degradation. We can also notice that, the model achieves promising results when the number of blocks is 4 and 5 on the cm1 dataset, and 6, 7, 9, 10 and 11 on the mw1 dataset, respectively. These observations imply the possibility to use a smaller number of blocks rather than the imbalance ratio. Although a smaller number of blocks brings imbalance to some extent, the AWA component is effective to adjust the weights of the slightly imbalanced samples.



(a) cm1 dataset (b) mw1 dataset

Figure 4: Ablation Study.

Ablation Study

Our model contains four main components: Data Block Construction (DBC), Data Space Improvement (DSI), Adaptive Weight Adjustment (AWA), and Ensemble Learning (EL). To study the effectiveness of these components, we implement several variants of our model: (i) DDAE-DBC, which is obtained by removing the DBC component; in fact, the variant DDAE-DBC also removes AWA and EL, since AWA and EL depend on DBC; (ii) DDAE-DSI, which is obtained by removing the DSI component; and (iii) DDAE-AWA, which is obtained by removing the AWA component. Note that, when the DBC component is used, it is necessary to use the EL component. Therefore, there is no such a variant "DDAE-EL". Figure 4 plots the experimental results on the cm1 and mw1 datasets. We can observe that all these variants perform significantly worse than the full fledged version (i.e., DDAE) on both datasets, regardless of G-mean, F-measure and AUC. The recall of DDAE is lower than DDAE-AWA on the mw1 dataset. This is because recall only focuses on the minority class while the AWA component always considers the overall gain from both classes – improving the recall brings in more false positives, which may decrease overall metrics such as F-measure. Overall, the ablation study essentially demonstrates that the proposed method integrates these components in a principle way to take the advantages of each part. In addition, we also observe that DDAE-DBC performs the worst. This reflects that the DBC component plays a much more important role in terms of the overall performance, because it provides the flexibility to support AWA and EL.

Conclusions

We present a novel model that consists of four components, which collaboratively contribute to classifying imbalanced data more effectively. We have conducted experiments on 14 public datasets, showing that our model is effective and competitive, compared with state-of-the-art methods. This study opens several future research directions include 1) extending the proposed model to multi-class classification problems; 2) utilizing clustering methods for improving classification results; 3) incorporating other classifiers (e.g., SVM, neural

networks) into our model; 4) extending the proposed model to other kinds of data, for example, image data, time series data.

Acknowledgment

This work was supported by the National Key R&D Program of China (2018YFB1004400), NSFC (U1811264, 61972425, 61602166, U1611264, U1711261, U1711262).

References

Alam, T.; Ahmed, C. F.; Zahin, S. A.; Khan, M. A. H.; and Islam, M. T. 2018. An effective ensemble method for multi-class classification and regression for imbalanced data. In *ICDM*, 59–74.

Barua, S.; Islam, M. M.; Yao, X.; and Murase, K. 2014. Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. *TKDE* 26(2):405–425.

Bhattacharya, S.; Rajan, V.; and Shrivastava, H. 2017. Icu mortality prediction: A classification algorithm for imbalanced datasets. In *AAAI*.

Castro, C. L., and Braga, A. P. 2013. Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data. *TNNLS* 24(6):888–899.

Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; and Kegelmeyer, W. P. 2002. Smote: synthetic minority over-sampling technique. *JAIR* 16:321–357.

Das, B.; Krishnan, N. C.; and Cook, D. J. 2015. RACOG and wracog: Two probabilistic oversampling techniques. *TKDE* 27(1):222–234.

Dumpala, S. H.; Chakraborty, R.; and Kopparapu, S. K. 2018. A novel data representation for effective learning in class imbalanced scenarios. In *IJCAI*, 2100–2106.

Fang, L.; Luo, Y.; Feng, K.; Zhao, K.; and Hu, A. 2019. Knowledge-enhanced ensemble learning for word embeddings. In *WWW*.

Fawcett, T. 2004. Roc graphs: Notes and practical considerations for researchers. *Machine learning* 31(1):1–38.

Fernández, A.; del Jesus, M. J.; and Herrera, F. 2009. Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets. *IJAR* 50(3):561–577.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Berlin, Germany: Springer, 2 edition.

He, H., and Garcia, E. A. 2008. Learning from imbalanced data. *TKDE* (9):1263–1284.

Huang, C.; Li, Y.; Change Loy, C.; and Tang, X. 2016. Learning deep representation for imbalanced classification. In *CVPR*, 5375–5384.

Khan, S. H.; Hayat, M.; Bennamoun, M.; Sohel, F. A.; and Togneri, R. 2017. Cost-sensitive learning of deep feature representations from imbalanced data. *TNNLS* 29(8):3573–3587.

Krawczyk, B. 2016. Cost-sensitive one-vs-one ensemble for multi-class imbalanced data. In *IJCNN*, 2447–2452.

Lin, X.; Quan, Z.; Wang, Z.; Huang, H.; and Zeng, X. A novel molecular representation with bigru neural networks for learning atoms. *Breifings in Bioinformaitics*.

Lin, X.; Wang, Z.; Ma, L.; and Wu, X. 2019. Saliency detection via multi-scale global cues. *IEEE Trans. Multimedia* 21(7):1646–1659.

Liu, W.; Wang, Z.; Yao, B.; Nie, M.; Wang, J.; Mao, R.; and Yin, J. 2018. Geographical relevance model for long tail point-of-interest recommendation. In *DASFAA*.

Liu, W.; Wang, Z.; Yao, B.; and Yin, J. 2019. Geo-alm: Poi recommendation by fusing geographical information and adversarial learning mechanism. In *IJCAI*.

Liu, X.-Y.; Wu, J.; and Zhou, Z.-H. 2009. Exploratory undersampling for class-imbalance learning. *TSMCB* 39(2):539–550.

Mease, D.; Wyner, A. J.; and Buja, A. 2007. Boosted classification trees and class probability/quantile estimation. *JMLR* 8(Mar):409–439.

Nie, M.; Wang, Z.; Gan, C.; Quan, Z.; Yao, B.; and Yin, J. 2019. An improved hierarchical datastructure for nearest neighbor search. In *AAAI*.

Nikolaou, N., and Brown, G. 2015. Calibrating adaboost for asymmetric learning. In *International Workshop on Multiple Classifier Systems*, 112–124.

Nikolaou, N.; Edakunni, N.; Kull, M.; Flach, P.; and Brown, G. 2016. Cost-sensitive boosting algorithms: Do we really need them? *Machine Learning* 104(2-3):359–384.

Quan, Z.; Wang, Z.; Le, Y.; Yao, B.; Li, K.; and Yin, J. 2019. An efficient framework for sentence similarity modeling. *TASLP* 27(4):853–865.

Seiffert, C.; Khoshgoftaar, T. M.; Hulse, J. V.; and Napolitano, A. 2010. Rusboost: A hybrid approach to alleviating class imbalance. *TSMC-A* 40(1):185–197.

Thai-Nghe, N.; Gantner, Z.; and Schmidt-Thieme, L. 2010. Cost-sensitive learning methods for imbalanced data. In *IJCNN*, 1–8.

Ting, K. M. 2000. A comparative study of cost-sensitive boosting algorithms. In *ICML*.

Vanschoren, J.; van Rijn, J. N.; Bischl, B.; and Torgo, L. 2013. Openml: Networked science in machine learning. *SIGKDD Explorations* 15(2):49–60.

Wang, N.; Zhao, X.; Jiang, Y.; and Gao, Y. 2018. Iterative metric learning for imbalance data classification. In *IJCAI*, 2805–2811.

Weinberger, K. Q., and Saul, L. K. 2009. Distance metric learning for large margin nearest neighbor classification. *JMLR* 10(Feb):207–244.

Weiss, G. M. 2004. Mining with rarity: a unifying framework. *ACM Sigkdd Explorations Newsletter* 6(1):7–19.

Wu, F.; Jing, X.-Y.; Shan, S.; Zuo, W.; and Yang, J.-Y. 2017. Multiset feature learning for highly imbalanced data classification. In *AAAI*.

Xu, Y.; and Z. Wang, B. Y.; Gao, X.; Xie, J.; and Guo., M. Skia: Scalable and efficient in-memory analytics for big spatial-textual data. *TKDE*.

Zhang, C., and Ma, Y. 2012. Ensemble learning. In *Ensemble machine learning*. 1–34.

Zhao, K.; Cai, Z.; Sui, Q.; Wei, E.; and Zhu, K. Q. 2014. Clustering image search results by entity disambiguation. In *ECML-PKDD*.