Neural Computing and Applications
https://doi.org/10.1007/s00521-023-08217-5

ORIGINAL ARTICLE

A self-organizing incremental neural network for imbalance learning

Yue Shao[1,2] · Baile Xu[1,3] · Furao Shen[1,2] · Jian Zhao[4]

Received: 26 January 2022 / Accepted: 6 January 2023

Abstract
Class imbalance learning deals with data that have very skewed class distributions, and commonly exists in real-world applications. Incremental learning has the ability to train a model continually using new data, which requires the model to learn the new information without forgetting the old one. While these two issues have been independently discussed, their joint treatment has not been studied thoroughly. This paper studies the combined challenges and proposes a balanced self-organizing incremental neural network (Balanced SOINN). First, we introduce Balanced SOINN, which can be trained incrementally with the resampling method. Then, we compare Balanced SOINN with other methods with artificial and real-world data. Our proposed method is competitive in artificial datasets in non-incremental scenarios and achieves the best performance with real-world datasets in incremental scenarios.

Keywords Class imbalance · Incremental learning · Oversampling · Undersampling

1 Introduction

Class imbalance problems see increasing demand from real-world applications. For example, when detecting fraud in online transactions, most of the transactions are real, while only a small part of them are fraudulent. In medical diagnosis, healthy people outnumber those affected. However, most algorithms expect balanced class distributions, which result in limited performances when presented with complex imbalanced datasets.

Incremental learning can train a model continually using new data, instead of being trained once on a whole dataset. Under these circumstances, catastrophic forgetting [1] is a big challenge. In recent years, incremental learning faced with continuous data streams during daily operation have been studied, but it is underexplored where certain types of data distributions over-dominate the instance space compared to other data distributions. With imbalanced class distribution, how a learning system can learn new data without forgetting previously learned information becomes a crucial problem.

This paper presents a resampling method based on self-organizing incremental neural network (SOINN) [2] to solve the class imbalance problem. We call the proposed method the balanced self-organizing incremental neural network (Balanced SOINN). Balanced SOINN inherits the ability of incremental learning of SOINN, and uses an undersampling and oversampling method to improve the performance when faced with imbalanced datasets. The main contributions of our work are:

- We propose a sampling method to solve class imbalance problems, where different minority class samples have different weights according to their level of node importance.

✉ Furao Shen
frshen@nju.edu.cn

✉ Jian Zhao
jianzhao@nju.edu.cn

Yue Shao
yueshao@smail.nju.edu.cn

Baile Xu
blxu@smail.nju.edu.cn

[1] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210046, China

[2] School of Artificial Intelligence, Nanjing University, Nanjing 210046, China

[3] Department of Computer Science and Technology, Nanjing University, Nanjing 210046, China

[4] School of Electronic Science and Engineering, Nanjing University, Nanjing 210046, China

Published online: 29 January 2023

- The proposed method can be used in incremental learning to learn new information without forgetting old data and outperforms other incremental methods on classification tasks with imbalanced class distribution.

The rest of the paper is organized as follows. Section 2 introduces some related work about imbalanced learning and incremental learning. Section 3 highlights the Balanced SOINN algorithm. Section 4 discusses the experimental results on both artificial and real-world data. Finally, Sect. 5 draws conclusions from the experiments.

2 Related work

When the number of instances of each class is not approximately equal, class imbalance occurs. Specifically, in binary classification problems, the majority class contains more examples, while the minority class contains fewer examples [3].

Sampling methods use oversampling and undersampling to re-balance the class distribution. These techniques are independent of the underlying classifier. Oversampling generates artificial data for the minority class. SMOTE [4] randomly generates new samples between a tuple of nearest neighbors of the minority class. SMOTE boderline1 &2 [5], SMOTE SVM [6] and SMOTE-IPF [7] are variants of this oversampling algorithm. ADASYN [8] adaptively generate synthetic data samples for the minority class. Undersampling methods, such as NearMiss [9], OSS [10] and IHT [11] reduce the number of samples for the majority class. However, those sampling methods can only re-balance the class distribution to a pre-defined level, which means the number of instances to be generated or reduced must be manually settled.

Cost-sensitive methods, which include cost-sensitive SVM [12] and the AdaCost family [13], assign different costs to instances based on the original data distribution and modify the learning algorithm to accept the costs. Those approaches make the algorithm spend more effort into learning the minority class better, but the costs are delicate to design and they are more difficult to implement than sampling [14].

Other methods such as active learning, one class learning, ensemble methods and deep learning are well-studied to solve the imbalance learning problem [15–20]. Active learning aims to label unlabeled data to increases the number of instances in the minority class. One class learning, such as One-class SVM [21], only learns one class at a time to avoid the bad influence of skewed data distribution. EasyEnsemble [22], BalanceCascade [22] and Self-paced Ensemble [23] are ensemble methods, which ensemble different classifier together to improve the performance. Deep learnig methods solve the imbalance problem through loss function engineering [24–29], meta-learning [30–33] or other approaches [34–37].

However, the above algorithms only focus on the imbalance problem and they cannot handle data stream or the incremental problem. SEA [38] is a streaming ensemble algorithm designed for large-scale classification. UOB and OOB [18] use undersampling and oversampling bagging techniques to handle both online learning and imbalance learning. They can learn from streaming imbalanced data with a changing underlying distribution, but they are not efficient enough due to the nature of ensemble.

3 The proposed approach

3.1 Overview of balanced SOINN

Balanced SOINN is an extension of the existing SOINN model. SOINN can represent the topology structure of input data in an unsupervised and incremental way. To better illustrate the whole algorithm, Fig. 1 shows the simplified learning procedure of the original SOINN. The algorithm's inputs are unlabeled data, shown as yellow circles and defined as "input signals." The outputs are blue circles, called "nodes," and edges between them. We feed the network with data one by one incrementally. At each time, one input signal is given. The first two input signals are added as nodes directly since the topology has not been formed. After that, once a new signal comes, its nearest (winner) and the second-nearest node (second winner) will be searched, and similarity thresholds are calculated. If the distance between the signal and the winner or second winner is greater than the corresponding similarity threshold, the signal will be inserted as a node. Otherwise, the winner and its neighbors will adjust their weights to record the information from the new signal. If the winner and the second winner are not connected, an edge will be formed between them to set up the topological representation. As the signals continue to be fed into the network, the algorithm goes to step 2 and repeats the whole procedure to form the topology structure. If the number of input signals is an integer multiple of parameter λ, the algorithm will do within-class insertion to erase accumulating error and noise reduction to eliminate the influence of noise. The learning process suggests that SOINN can realize incremental learning. Old knowledge can be well preserved using nodes, and new information can be learned by adding new nodes or updating old nodes.

Balanced SOINN is developed based on the original SOINN, and three main improvements have been made:

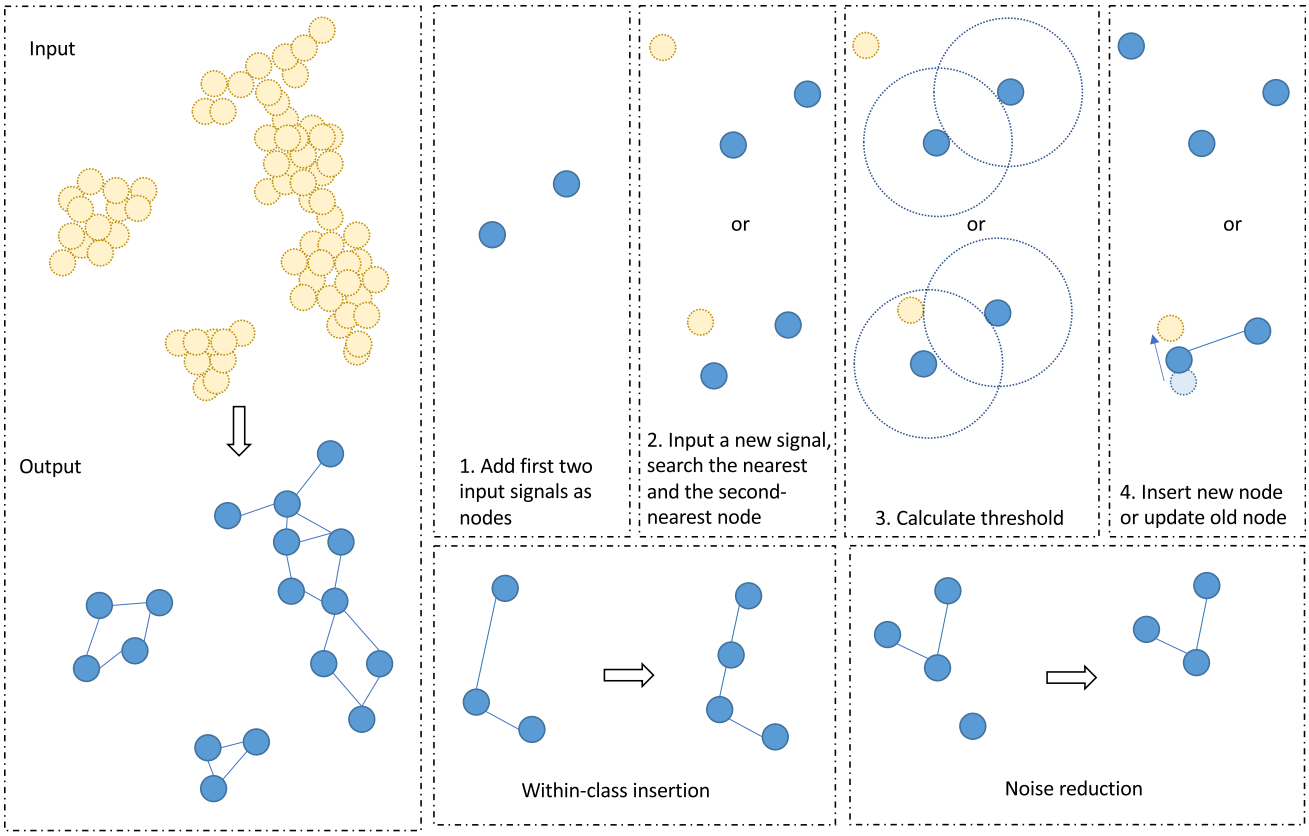

Fig. 1 Simplified learning procedure of the original SOINN

- Labels are introduced in Balanced SOINN to make the algorithm applicable for supervised learning. The addition and update of nodes depend not only on distance and threshold but also on labels. The noise reduction part has also been adjusted to make nodes from the minority class less likely to be removed.
- Within-class insertion is removed because it has little contribution to inserting new nodes but needs too many parameters.
- An oversampling method is adopted based on node importance. The learned output nodes are the topology representation of the input data and can be treated as output after undersampling. Thus, the data are well re-balanced.

The simplified learning procedure of Balanced SOINN is shown in Fig. 2. Like the original one, Balanced SOINN can realize incremental and topology learning. The hollow circles of different colors are input data with different labels. The outputs of Balanced SOINN are solid circles and edges. Solid circles of the same color have the same labels, and circles of lighter colors are the result of over-sampling. Compared with the output of original SOINN in Fig. 1, the output of Balanced SOINN tends to have more nodes of the minority class. The first two input signals are added as nodes, and their labels are recorded. On arrival of a new signal, the algorithm will search for the nearest and second nearest node, then calculate the similarity threshold like the existing SOINN. If both winner and second winner are from the same class as the input data, the insertion and update are the same as the existing SOINN. If none of them are of the same label as the input signal, a new node will be added. When there is a node in the winner or the second winner, which is from the same class as the input signal and belongs to the same cluster according to the threshold criterion, the node will update its weight to get closer to the new signal, as shown in step 3 and step 4 of Fig. 2. Otherwise, the signal will cause the creation of a new node. Since no insertion has been done during the update, the algorithm will oversample the node if the node is from the minority class and has high node importance. After λ learning iterations, the Balanced SOINN executes a noise-reduction algorithm to make the learned prototype more robust. The learning process goes on until the network finish learning.

3.2 Complete algorithm of balanced SOINN

Balanced SOINN uses undersampling and oversampling methods to re-balance data distribution to overcome the class imbalance problem. It inherits the original SOINN

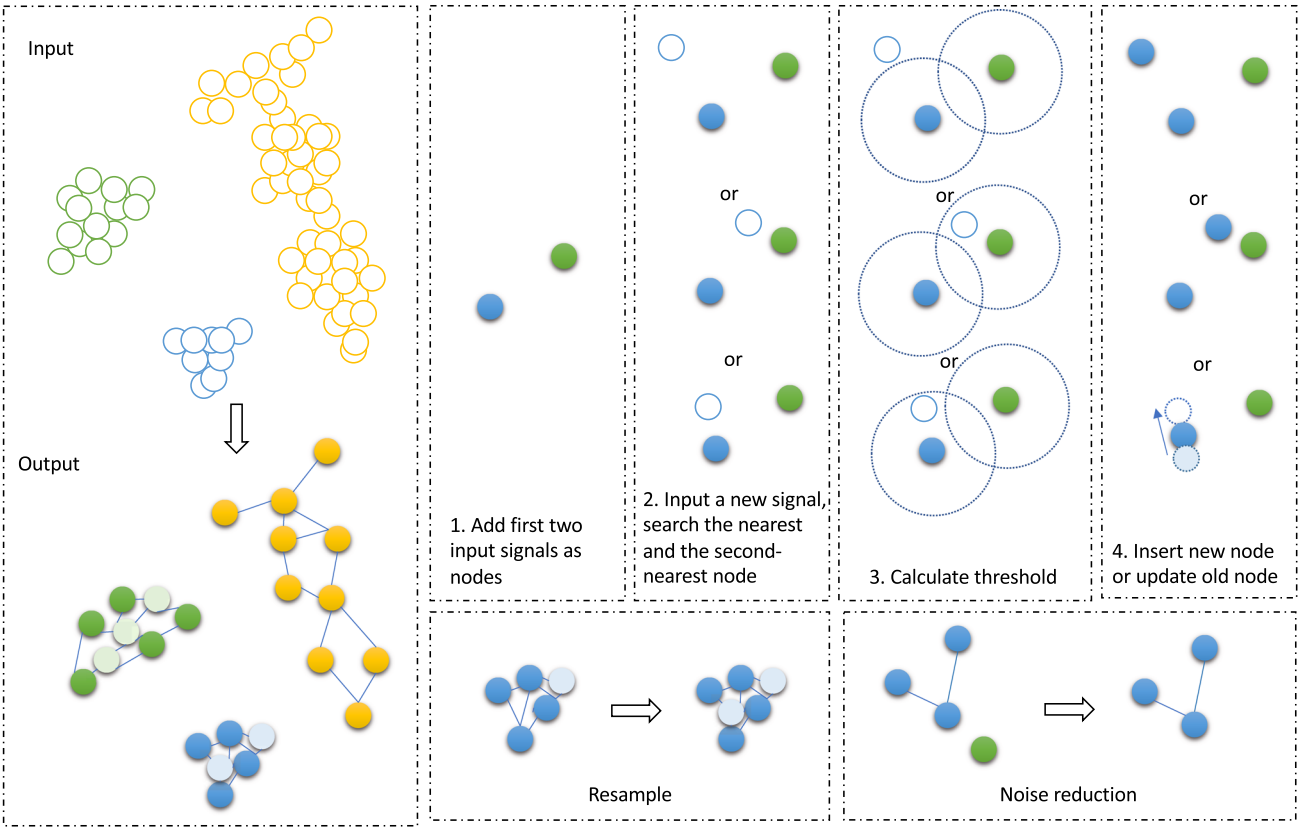

Fig. 2 Simplified learning procedure of Balanced SOINN

and can be formatted as $<N, E>$, where N is the neuron set and E is the edge set, which contains prototypes of the input data. Unlike the original SOINN, the Balanced SOINN is suitable for supervised learning. Each node n_i in the neuron set can be denoted as $<W_i, label_i>$, where W_i is a prototype node of the same dimensions of the input data, and $label_i$ is the label of W_i.

The sampling method works according to the node importance, which is evaluated when a input signal is coming. It re-balances the class distribution without manually setting the number of sampling instances. If a signal comes from the minority class, it will have more information than that from the majority class and is more likely to be oversampled, which leads to higher node importance. In our algorithm, higher node importance means that the signal is from the minority class, lies in high-density areas, or is on the boundary of different classes. Furthermore, signals of high node importances tend to be oversampled. As for undersampling, it uses SOINN to study the topological structure of the input. After that, redundant nodes are discarded, and prototypes are left. Figure 3 shows the flowchart of Balanced SOINN. Algorithm 1 is the detailed algorithm of Balanced SOINN. When an input signal and its label are given to Balanced SOINN, the algorithm updates the counter of the label, which is used for judging the imbalance level when sampling. We add the first two signals as new nodes directly and continue to process the third signal. The algorithm searches for the winner and the second winner of the third signal and updates their border probability. If the input signal belongs to a different cluster of the winner or second winner according to the similarity threshold criterion, or if none of the labels of the winner or second winner is the same as the signal, between-class insertion is done. A new node is created and added to the network. After between-class insertion, the algorithm continues to process the next signal. Otherwise, the algorithm decides whether edge creation should be done. If those two nodes belong to the same class and are not connected, Balanced SOINN creates an edge between the winner and the second winner. The nearest node and their neighbors with the same label are updated to remember the new knowledge from the input signal. Edges that are too old are removed to forget old-fashioned information. The density of the nearest node is updated. After that, the oversampling algorithm generates some synthetic nodes from the nearest node with the same label. For every λ learning iterations, the network deletes the nodes caused by noise. The whole process repeats until all signals are processed.

Algorithm 1 The complete algorithm of Balanced SOINN

1: Initialize neuron set N=∅, edge set E=∅, counter map C=∅.
2: Input new signal $< \xi, label >$, where $\xi \in R^n$,$label \in N$.
3: if label not in C, then initialize $C[label] = 1$, else $C[label] = C[label] + 1$.
4: If it is the first two signals, directly add it to N according to the formula below and go to step (2) to continue learning.

$$N = N \cup \{< \xi, label >\} \tag{1}$$

5: Search the nearest node (winner) s_1, and the second-nearest node (second winner) s_2 by,

$$s_1 = \arg\min_{n \in N} \|\xi - W_n\| \tag{2}$$

$$s_2 = \arg\min n \in N \setminus \{s_1\} \|\xi - W_n\| \tag{3}$$

6: Update s_1 and s_2's probabilities of being border nodes.
7: If the distance between ξ and $s1$ or $s2$ is greater than similarity threshold T_{s_1} or T_{s_2}, or none of $label_{s1}$ and $label_{s1}$ are the same with label of the input signal, the input signal is a new node, add it to N according to formula 1 and go to step (2) to process the next signal. T_s of node s is calculated as follows:

$$T_s = \begin{cases} \arg\max_{(s,t) \in E} \|W_s - W_t\|, & \text{if node } s \text{ has topological neighbors} \\ \arg\min_{t \in N \setminus \{s\}} \|W_s - W_t\|, & \text{if node } s \text{ is not connected to any node} \end{cases} \tag{4}$$

8: If a connection between $s1$ and $s2$ does not exist and both of $label_{s1}$ and $label_{s1}$ are same with label, create it. Set the age of the connection between $s1$ and $s2$ to zero.
9: Find the nearest node s with the same label.
10: Increment the age of all edges emanating from s by 1.
11: Adapt the weight vectors of s and its direct topological neighbors by fraction $\epsilon_1(t)$ and $\epsilon_2(t)$ of the total distance to the input signal,

$$\Delta W_s = \epsilon_1(t)(\xi - W_s) \tag{5}$$

and for all direct neighbors i of s,

$$\Delta W_i = \epsilon_2(t)(\xi - W_i) \tag{6}$$

We adopt a scheme to adapt the learning rate over time by

$$\epsilon_1(t) = \frac{1}{t} \tag{7}$$

$$\epsilon_2(t) = \frac{1}{100t} \tag{8}$$

12: Update density vector of node s.
13: Remove edges with age greater than a pre-defined threshold age_{max}. If this results in nodes having no more emanating edges, remove them as well.
14: Oversample node s by probability $NodeImportance_i$. The probability $NodeImportance_i$ is calculated by Figure 4.
15: If the number of input signals generated so far is an integer multiple of parameter λ, delete noise nodes.
16: Go to Step (2) to continue the learning until the learning time is satisfied.

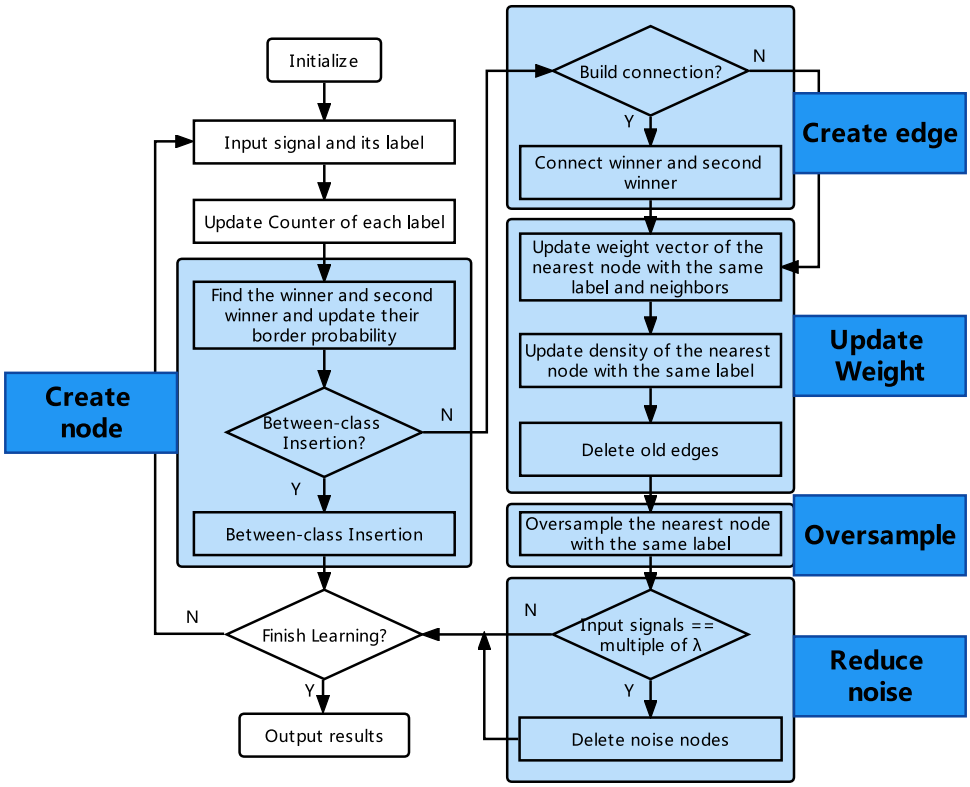

Fig. 3 Flowchart of Balanced SOINN

3.3 Analyses of algorithm

3.3.1 Evaluation of node importance

Signals from the minority class, lie in high-density areas or far from the center, have higher node importance. When a new signal $<\xi, label>$ is coming, whether it is from the minority class, it is judged by the counter map C in Algorithm 1. Moreover, the density of a node in our algorithm is the same as the concept "node density" defined in [39]. The basic idea is to calculate the local number of samples, which is defined as "point". First, the mean distance $\bar{d}_i$ of node i is calculated from its neighbors in Eq. (9), m is the number of neighbours of node i and W_i is the weight vector of node i.

$$\bar{d}_i = \frac{1}{m}\sum_{j=1}^{m} \|W_i - W_j\| \tag{9}$$

The point of node i is calculated as

$$p_i = \begin{cases} \dfrac{1}{(1+\bar{d}_i)^2}, & \text{if node } i \text{ is winner} \\ 0, & \text{if node } i \text{ is not winner} \end{cases} \tag{10}$$

The mean of the accumulated point of a node describes the density of that node. If the mean distance of node i to its neighbors is large, the point of nodes in this area is low and thus the density in this area will be low.

Furthermore, we define the "border node" if the node lies on the border of different classes, which is more helpful in finding the decision boundary than common nodes. Since nodes on the border have more neighbors of different labels than nodes that lie in the center of the class distribution, we use the number of different labeled nodes, which is defined as $BorderCount_i$, to describe node i's probability of being a border node. When a new signal comes, if the winner has a different label, the BorderCount of the winner node i will be increased by 1. The probability of node i being a border node is defined as

$$BorderProb_i = BorderCount_i / WinningTimes_i \tag{11}$$

WinningTimes count the number of times a node becomes the winner, which reflects the likelihood of a node lies in high-density areas. *BorderCount* of a node in high-density areas is higher than that in low-density one, so we divide the *BorderCount* by *WinningTimes* to reduce the impact.

3.3.2 Denoising

Noise often exists in real-world datasets. Balanced SOINN will generate some node from noise during training. So we adopt a noise-reduction algorithm from [40] and do some justification to reduce the removal of nodes from the minority class. Whether a node comes from the majority class is judged by its imbalance index. The imbalance index is the proportion of samples in one class to the

number of all the classes, and a high imbalance index means the node is from the majority class. The imbalance index of node i with $label_n$ is defined as follows,

$$imbalanceIndex_i = \frac{counter\ map\ C[label_i]}{sum(counter\ map\ C)} \tag{12}$$

Nodes lie in regions with low density have fewer edges connected to other nodes, which are likely to be generated from noisy data. However, for nodes from the minority class, they often have fewer edges than others due to their small quantity. So we should consider the imbalance index of each node when executing the noise-reduction algorithm described in Algorithm. 2.

3.3.3 Sampling algorithm

To re-balance the dataset, we adopt oversampling and undersampling methods in our algorithm, which is the most crucial part of the Balanced SOINN algorithm. SOINN is useful in handling online non-stationary data and gives typical prototype nodes of every cluster. After processed by SOINN, the input signals are undersampled because the number of new nodes added to the neuron set N are less than the number of the signals. When all signals have been processed, the nodes in N are excellent representatives of the data's original distribution. So we focus on the oversample algorithm based on SMOTE, which is detailed in Fig. 4. When a signal i comes and causes the algorithm to

Algorithm 2 Noise-reduction algorithm of Balanced SOINN

1: For a node i in the neuron set N, find k nearest nodes of node i.
2: For node i has less than min_{edge} edges, delete it from the neuron set N if the major voting of k nearest nodes has a different class label with node i, or the following inequality is satisfied.

$$imbalanceIndex_i > \frac{1}{C} - 0.1 \tag{13}$$

In that equation, C is the number of classes. If node i satisfies the inequality, $C[label_i]$ is large, and the node is from the majority class.
3: For node i has more than min_{edge} edges, delete it from the neuron set N if none of k nearest nodes have the same class label with node i.
4: Repeat this process until all nodes are processed.

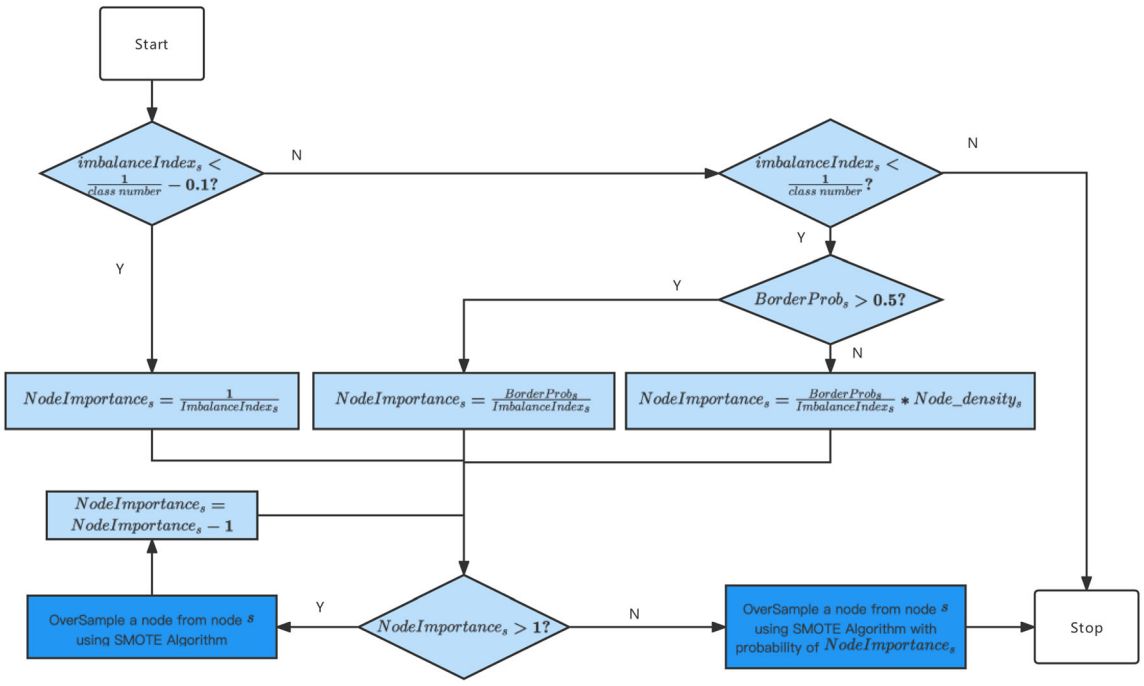

Fig. 4 OverSample algorithm of Balanced SOINN. The original SMOTE is marked in blue and our modifications are marked in lighter blue

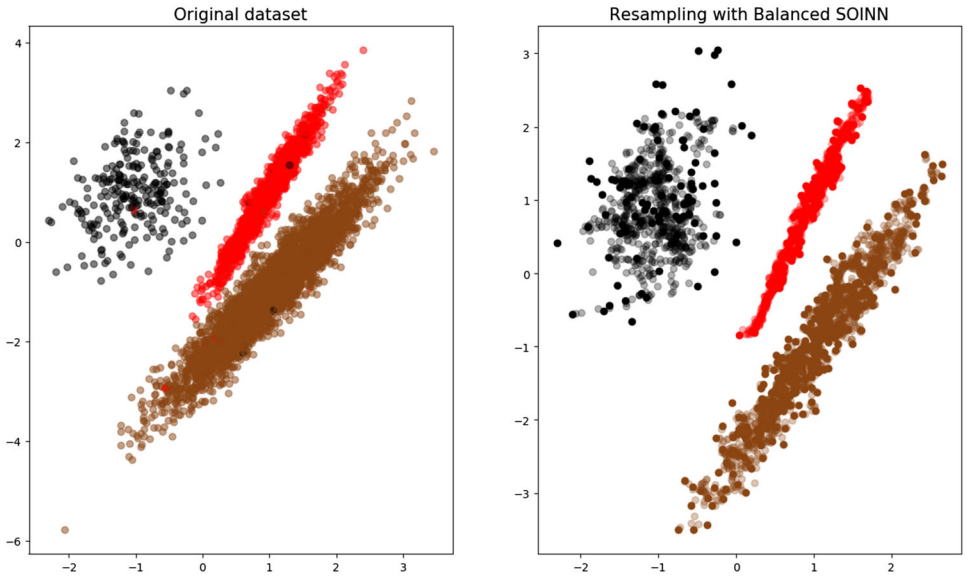

Fig. 5 Sampling result of Balanced SOINN

oversample the nearest node s with the same label, we will calculate $NodeImportance_s$ using the algorithm. The imbalance index is the proportion of samples in one single class to the number of all signals. The node from the minority class has a small imbalance index, so we use it to decide whether to oversample. Higher node importance means that the signal is from the minority class, located in a high-density region or close to the boundary. We use node importance to decide how many samples should be generated by oversampling.

Node importance is inversely proportional to the imbalance index, which allows the algorithm to adapt to different levels of data imbalance. For a node with a low imbalance index, which is less than $\frac{1}{class\ number} - 0.1$, we assume it is from the minority class and oversample it more. For a node with a higher imbalance index, greater than $\frac{1}{class\ number} - 0.1$ but less than $\frac{1}{class\ number}$, we oversample it according to its probability of being a border node and its density. The *BorderProb* and density of a node are defined in Sect. 3.3.1. A node with an imbalance index higher than $\frac{1}{class\ number}$ is likely from the majority class, so we do not oversample it. When we decide to oversample a node a, SMOTE algorithm selects one of its k nearest minority class neighbors b at random and connects a and b to form a line in the feature space. Then a new instance is randomly picked from the line, which is a convex combination of the two chosen instances a and b. The synthetic instance will be added to neuron set N as a new node. If the Node importance is greater than 1, the oversampling process will be repeated more than once.

The Sampling result of Balanced SOINN applied on synthetic data is shown in Fig. 5. The figure on the left shows the input data. There are 5000 samples drawn in different colors, 262 of them are black, 1253 of them are red, and the rest 3485 dots are brown. After processed by Balanced SOINN, there are 747 black dots, 1564 red dots, and 2346 brown dots. The total number of samples becomes 4678. The black dots lie in regions of red dots and brown dots have been removed, which indicates that the algorithm is robust to noise. The dataset is well re-balanced without losing too much information.

4 Experimental results

In this section, we use several algorithms to compare their performance on classifying imbalanced data, especially when it is incremental. Balanced SOINN and SOINN are self incremental neural networks. Their outputs are the prototypes of the input data. To illustrate Balanced SOINN's ability to re-balance imbalanced data incrementally, we train a Gaussian Naive Bayes classifier on the prototypes and take the classifier's result as the final output on binary classification task.

The Test-Then-Train method is used in our experiment in the context of incremental learning, which input data blocks in sequence. The classifier is trained with the first chuck. For the following chucks, we test the classifier on the chunk first and then train the classifier with it. After all chucks have been tested, the average scores of their performance from the second chuck to the last are compared. We choose the balanced accuracy, F1, and G_mean score to evaluate how well the performance is without the influence of the imbalanced dataset. Balanced accuracy equals to the average of true positive rate and true negative rate [41]. It can serve as an overall performance metric for a model whose data is imbalanced.

Table 1 Comparison results of Balanced SOINN and other methods: balanced accuracy, boldface balanced accuracy means the best classifier

	0.05 vs 0.95	0.075 vs 0.925	0.1 vs 0.9	0.2 vs 0.8	0.4 vs 0.6	Average
SMOTE [4]	0.8620	0.8878	0.9057	0.9277	0.9421	0.9051
SMOTE Borderline [5]	0.8404	0.8590	0.8702	0.8949	0.9265	0.8782
ADASYN [8]	0.8020	0.8359	0.8576	0.8937	0.9250	0.8628
SMOTE ENN [42]	**0.8737**	**0.8984**	**0.9122**	0.9317	0.9438	**0.9120**
SMOTE Tomek [42]	0.8636	0.8912	0.9070	0.9279	0.9438	0.9067
EasyEnsemble [22]	0.8595	0.8877	0.9029	0.9271	0.9431	0.9041
BalancedBagging [43]	0.8611	0.8871	0.9054	0.9273	0.9428	0.9047
RUSBoost [44]	0.8531	0.8939	0.9092	0.9321	**0.9468**	0.9070
SOINN [2]	0.8213	0.8648	0.8727	0.9306	0.9434	0.8866
Balanced SOINN	0.8512	0.8947	0.9118	**0.9329**	0.9451	0.9071

Table 2 Comparison results of Balanced SOINN and other methods: F1, boldface F1 means the best classifier

	0.05 vs 0.95	0.075 vs 0.925	0.1 vs 0.9	0.2 vs 0.8	0.4 vs 0.6	Average
SMOTE [4]	0.9553	0.9561	0.9608	0.9543	0.9500	0.9553
SMOTE Borderline [5]	0.9132	0.8991	0.9006	0.9015	0.9256	0.9080
ADASYN [8]	0.8498	0.8693	0.8823	0.8999	0.9243	0.8851
SMOTE ENN [42]	0.9694	0.9717	0.9728	0.9618	0.9528	0.9657
SMOTE Tomek [42]	0.9587	0.9597	0.9639	0.9565	0.9524	0.9582
EasyEnsemble [22]	0.9526	0.9548	0.9570	0.9553	0.9519	0.9543
BalancedBagging [43]	0.9560	0.9542	0.9596	0.9556	0.9515	0.9554
RUSBoost [44]	0.9296	0.9671	0.9645	0.9598	0.9547	0.9552
SOINN [2]	0.9886	0.9871	0.9842	**0.9796**	0.9580	**0.9795**
Balanced SOINN	**0.9901**	**0.9875**	**0.9855**	0.9730	**0.9611**	0.9794

Table 3 Comparison results of Balanced SOINN and other methods: G_mean, boldface G_mean means the best classifier

	0.05 vs 0.95	0.075 vs 0.925	0.1 vs 0.9	0.2 vs 0.8	0.4 vs 0.6	Average
SMOTE [4]	0.8596	0.8868	0.9051	0.9276	0.9421	0.9043
SMOTE Borderline [5]	0.8404	0.8583	0.8692	0.8925	0.9250	0.8771
ADASYN [8]	0.8000	0.8337	0.8555	0.8911	0.9235	0.8608
SMOTE ENN [42]	**0.8701**	**0.8963**	**0.9108**	0.9316	0.9438	**0.9105**
SMOTE Tomek [42]	0.8608	0.8901	0.9062	0.9278	0.9438	0.9057
EasyEnsemble [22]	0.8573	0.8868	0.9024	0.9271	0.9431	0.9034
BalancedBagging [43]	0.8585	0.8863	0.9048	0.9273	0.9428	0.9039
RUSBoost [44]	0.8528	0.8921	0.9084	**0.9320**	**0.9468**	0.9064
SOINN [2]	0.8018	0.8543	0.8635	0.9286	0.9430	0.8782
Balanced SOINN	0.8383	0.8892	0.9083	**0.9320**	0.9444	0.9024

The imbalance ratio (IR) is the proportion the number of instances in the majority class to the instances in the minority one, reflecting how skewed the data distribution is. To test the proposed algorithm on binary classification, we conduct experiments on both artificial and real-world datasets with different imbalance ratios and give an in-depth analysis of their performances.

4.1 Artificial data

The artificial data have 2 features. We generate 10,000 samples with a fixed imbalance ratio, half of which are used for training and the other half for testing. We compare Balanced SOINN with other non-incremental techniques under a non-incremental environment. Data with dynamic imbalance ratios are also generated to show that the proposed algorithm performs better than the original SOINN in imbalance learning under an incremental environment. The data are split into 30 chunks for incremental learning.

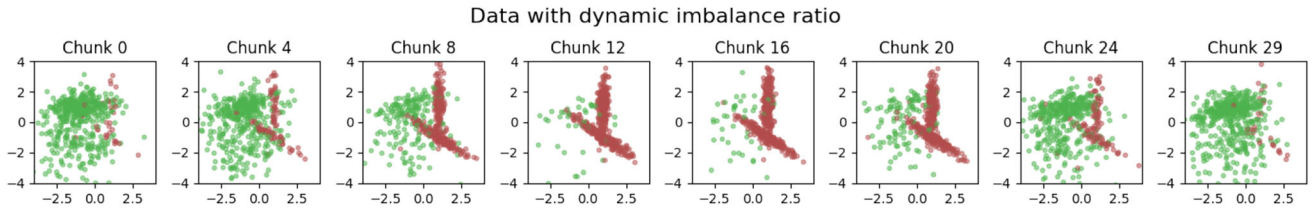

Fig. 6 Data with dynamic imbalance ratio

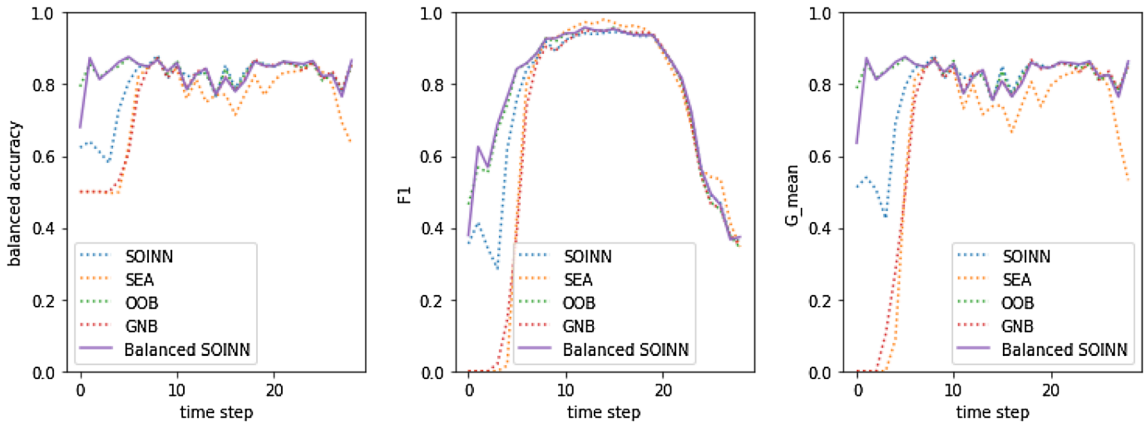

Fig. 7 Comparison results of Balanced SOINN and other classifiers on data with dynamic imbalance ratio

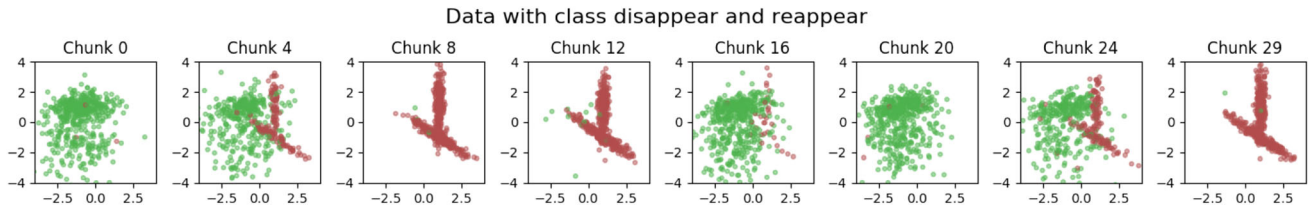

Fig. 8 Data with class disappearance and reappearance

Each chuck has 500 samples. An emerging class and a disappearing class are taken into consideration to illustrate its performance on incremental learning.

4.1.1 Static imbalanced ratio

This section compares Balanced SOINN to a series of sampling and ensemble techniques with a static imbalance ratio in non-incremental scenarios. Sampling methods, such as SMOTE, SMOTE Borderline, ADASYN, SMOTE ENN [42], and SMOTE Tomek [42], cannot be used for binary classification tasks directly. Like SOINN and Balanced SOINN, We train Gaussian Naive Bayes classifiers on their outputs and take the classifiers' results as the final outputs. The base models of ensemble methods are Gaussian Naive Bayes classifiers. With a fixed imbalance ratio between the minority and the majority classes, we can observe Balanced SOINN's performance with different imbalance ratio. We set parameters of SOINN and Balanced SOINN as $\lambda = 200$, $\text{age}_{\max} = 30, k = 3$ and $\min_{\text{edge}} = 1$. Table 1 shows the average balanced accuracy of Balanced SOINN and other methods when samples from the minority class are 5%, 7.5%, 10%, 20%, and 40%. The comparison results of F1 and G_mean are shown in Tables 2 and 3, the best performances are highlighted in boldface.

Experimental results of the proposed Balanced SOINN and SOINN on imbalanced data with static imbalanced ratios show that although for F1, Balanced SOINN is not as good as the original SOINN, the difference between the two methods is not evident. Balanced SOINN improves performance on balanced accuracy and G_mean significantly. Usually, the improvement is more noticeable when the imbalance ratio is greater. A conclusion can be drawn that combining SOINN with a sampling method to rebalance the skewed data distribution can significantly improve the classification. In addition, the proposed method is also very competitive compared to other sampling and ensemble techniques under a non-incremental environment. These results suggest that Balanced SOINN

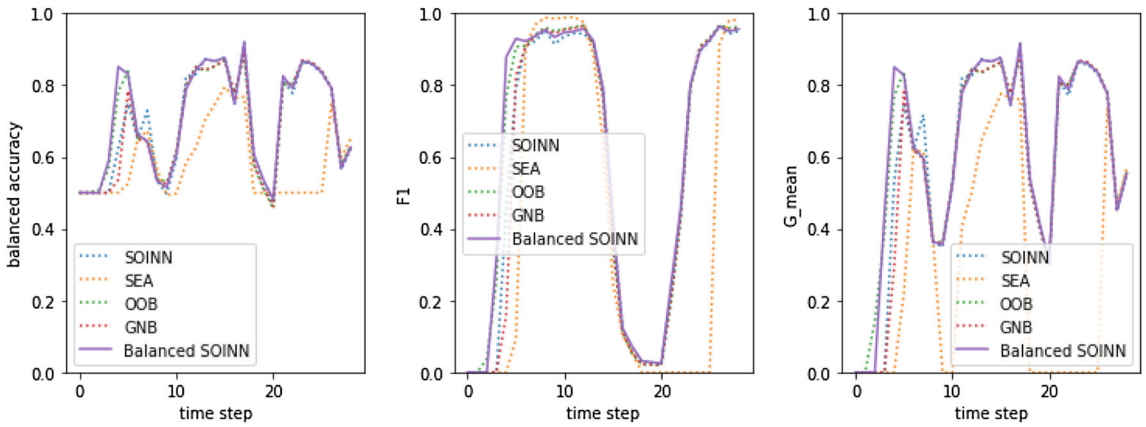

Fig. 9 Comparison results of Balanced SOINN and other classifiers on data with class disappearance and reappearance

Table 4 Comparison results of Balanced SOINN and other classifiers on data with class disappear and reappear

	Balanced accuracy	F1	G_mean
SEA	0.5822	0.4035	0.2816
OOB	0.6988	0.5862	0.6078
GNB	0.6858	0.5521	0.5714
SOINN	0.6909	0.5619	0.5838
Balanced SOINN	**0.7046**	**0.5979**	**0.6102**

Table 5 Real-word datasets chosen from the KEEL dataset repository

Name	#Atts	#Min	#Maj	IR	λ	$age_{\max}$
glass1	9	76	138	1.82	100	100
wisconsin	9	239	444	1.86	300	50
glass0	9	70	144	2.06	200	50
glass-0-1-2-3_vs_4-5-6	9	51	163	3.2	100	100
vehicle0	18	199	647	3.25	300	20
segment0	19	329	1976	6.02	500	200
yeast-0-3-5-9_vs_7-8	8	50	456	9	200	20
yeast-0-5-6-7-9_vs_4	8	51	477	9.35	500	20
vowel0	13	90	898	9.98	300	300

can achieve better results than the original SOINN on imbalanced datasets and perform well under a non-incremental environment.

4.1.2 Dynamic imbalanced ratio

When the model is trained on data blocks incrementally, we change the imbalance ratio dynamically to see if Balanced SOINN can quickly adapt to the changing imbalance ratio. The data is split into 30 chunks. Each chuck has 500 samples. We test it on several other classifiers to compare their results. UOB is not robust enough against dynamic changes in class imbalance, so we do not compare it in this situation. We compare Balanced SOINN with SOINN, SEA, and OOB with Gaussian Naive Bayes as a baseline to prove their efficiency in imbalance learning. We set parameters as $\lambda = 300$, $\text{age}_{\max} = 300$, $k = 3$ and $\min_{\text{edge}} = 1$. Figure 6 shows the synthetic data of binary classification. In the beginning, the first data chuck contains two classes, and the green nodes are from the majority class. With the changing of the imbalance ratio, the number of green nodes is decreasing, and the green-labeled class becomes the minority class. These artificial data are designed to show the performance of various methods in the situation of dynamic imbalance ratio and the changing roles of majority and minority class.

Figure 7 shows the result of Balanced SOINN and other classifiers on data with a dynamic imbalance ratio. Balanced SOINN performs well at the beginning with skewed data distribution and can rapidly learn the new information when data changes.

Figure 8 shows the synthetic data with the changing imbalance ratio, which results in class disappearance and reappearance. In incremental learning, the classifier should have the ability to remember the old signal while adapting to new signals quickly. So the classifier must remember the old class even if no samples of this class are fed into it in the current block and output a good result when the class reappears. We compare the proposed Balanced SOINN with other classifiers in Fig. 9. Table 4 shows the average balanced accuracy, F1 score, and G_mean on various classifiers.

In Fig. 9, Balanced SOINN outperforms others in incremental learning and learns new data without forgetting previously learned information. Table 4 shows that our method achieves the best performance on balanced accuracy, F1, and G_mean. Balanced SOINN can store input

Table 6 Balanced accuracy of Balanced SOINN and other classifiers on real-world data

	SEA	OOB	UOB	SOINN	Balanced SOINN	GNB
glass1	0.6316	0.5891	0.6205	0.6337	**0.6579**	0.6316
wisconsin	0.9645	**0.9714**	0.9679	0.9445	0.9341	0.9634
glass0	0.6690	0.6601	0.6690	0.6763	**0.6907**	0.6690
glass-0-1-2-3_vs_4-5-6	0.8371	**0.8653**	0.8309	0.8436	**0.8653**	0.8371
vehicle0	0.6852	0.6924	0.7026	**0.7278**	0.7157	0.6822
segment0	0.9220	0.8966	0.8870	0.9164	**0.9406**	0.9014
yeast-0-3-5-9_vs_7-8	0.5699	0.5621	**0.5750**	0.5539	0.5743	0.5454
yeast-0-5-6-7-9_vs_4	0.5004	0.5125	0.6287	0.5433	**0.6596**	0.4992
vowel0	0.8057	0.8707	**0.8756**	0.6467	0.8701	0.8621
average Balanced accuracy	0.6995	0.7105	0.7416	0.6651	**0.7612**	0.7020
average rank	3.6667	3.56667	3.2222	3.3333	**2.0000**	4.4444

Table 7 F1 of Balanced SOINN and other classifiers on real-world data

	SEA	OOB	UOB	SOINN	Balanced SOINN	GNB
glass1	0.5918	0.5510	0.5882	0.6019	**0.6200**	0.5918
wisconsin	0.9417	**0.9487**	0.9416	0.9072	0.8910	0.9413
glass0	0.5714	0.5625	0.5714	0.5773	**0.5895**	0.5714
glass-0-1-2-3_vs_4-5-6	0.7556	**0.8000**	0.6780	0.7727	**0.8000**	0.7556
vehicle0	0.4874	0.4948	0.5037	**0.5382**	0.5291	0.4846
segment0	0.7125	0.6285	0.6165	0.7407	**0.7694**	0.6350
yeast-0-3-5-9_vs_7-8	0.1929	0.2154	0.2337	0.1992	**0.2405**	0.2021
yeast-0-5-6-7-9_vs_4	0.1498	0.1551	0.2240	0.1068	**0.2857**	0.1509
vowel0	0.6764	0.5920	0.5865	0.3002	**0.6787**	0.6671
average F1	0.4329	0.3977	0.4152	0.3367	**0.4936**	0.4138
average rank	3.6667	3.6667	3.8889	3.5556	**1.6667**	3.8889

Table 8 G_mean of Balanced SOINN and other classifiers on real-world data

	SEA	OOB	UOB	SOINN	Balanced SOINN	GNB
glass1	0.6177	0.5765	0.5971	0.6070	**0.6387**	0.6177
wisconsin	0.9643	**0.9710**	0.9675	0.9437	0.9318	0.9633
glass0	0.6267	0.6255	0.6267	0.6370	**0.6573**	0.6267
glass-0-1-2-3_vs_4-5-6	0.8313	**0.8614**	0.8300	0.8371	**0.8614**	0.8313
vehicle0	0.6792	0.6839	0.6917	**0.7190**	0.7135	0.6749
segment0	0.9200	0.8920	0.8825	0.9097	**0.9395**	0.8966
yeast-0-3-5-9_vs_7-8	**0.3999**	0.3700	0.3422	0.3113	0.3712	0.3124
yeast-0-5-6-7-9_vs_4	0.4062	0.3686	0.5351	0.2436	**0.6132**	0.3167
vowel0	0.7757	0.8622	**0.8694**	0.3818	0.8633	0.8490
average G_mean	0.6254	0.6232	0.6573	0.4616	**0.6968**	0.5937
average rank	3.1111	3.6667	3.5556	4.0000	**1.8889**	4.1111

data as prototypes. When the class disappears, it will not forget the information and shows better balanced accuracy, F1, and G_mean than other algorithms.

4.2 Real-word data

We compare Balanced SOINN with SOINN, SEA, OOB, UOB, and Gaussian Naive Bayes on nine real-world datasets. We chose those datasets from the KEEL dataset [45], which were gathered from different well-known sources such as UCI repository [46]. The datasets are divided into chunks of 100 samples each. The imbalance ratio between minority and majority of those datasets range from 1.5 to 10. Table 5 summarizes the properties of the datasets, including the number of attributes, the number of minority class examples, the number of majority class

examples, and the imbalance ratio in the dataset. Values of λ and age_{max}, shown in Table 5, are tuned by a 10-fold cross-validation scheme for every dataset. Other parameters are set as $k = 3$ and $\min_{\text{edge}} = 1$.

The final results are shown in Tables 6, 7 and 8. The metrics are then ranked to make comparisons among algorithms, where rank 1 is the best performing algorithm. Since no single method can outperform all others in all measures and all datasets, we use average rank to evaluate the performance. The best results are highlighted.

Balanced SOINN does not perform the best on every dataset but is competitive on these nine datasets and gives the best average score and average rank on balanced accuracy, F1, and G-mean.

5 Conclusion

In this paper, we proposed the Balanced SOINN, a new method based on self-organizing incremental neural network (SOINN), to solve the class imbalance problem. It automatically learns the prototypes of input data to undersample all the data and oversamples nodes of the minority class, on boundary areas, or in high-density areas. By doing so, Balanced SOINN can re-balance the data distribution to overcome the bad influence of skewed data distribution. With the ability of incremental learning, Balanced SOINN can handle the situation where class disappearance and emergence happen. The experimental results on artificial datasets and real-world datasets show that our proposed method performs well with class imbalance problem and outperforms other incremental learning methods.

Funding This work was supported in part by the National Key R &D Program of China under Grant STI 2030-Major Projects 2021ZD0201300, and by the National Science Foundation of China under Grant 62276127.

Data availability The datasets generated during and analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest No potential conflict of interest was reported by the authors.

References

1. Mccloskey M, Cohen NJ (1989) Catastrophic interference in connectionist networks: the sequential learning problem. Psychol Learn Motiv 24:109–165
2. Furao S, Hasegawa O (2006) An incremental network for on-line unsupervised classification and topology learning. Neural Netw 19(1):90–106
3. He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284
4. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357
5. Han H, Wang W-Y, Mao B-H (2005) Borderline-smote: a new over-sampling method in imbalanced data sets learning. In: International conference on intelligent computing, pp 878–887. Springer
6. Nguyen HM, Cooper EW, Kamei K (2009) Borderline over-sampling for imbalanced data classification. In: Proceedings: 5th international workshop on computational intelligence and applications, vol 2009, pp 24–29 (2009). IEEE SMC Hiroshima Chapter
7. Sáez JA, Luengo J, Stefanowski J, Herrera F (2015) SMOTE-IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. Inf Sci 291:184–203
8. He H, Bai Y, Garcia EA, Li S (2008) Adasyn: adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE international joint conference on neural networks (ieee world congress on computational intelligence), pp 1322–1328. IEEE
9. Mani I, Zhang I (2003) kNN approach to unbalanced data distributions: a case study involving information extraction. In: Proceedings of workshop on learning from imbalanced datasets, vol 126
10. Kubat M, Matwin S, et al. (1997) Addressing the curse of imbalanced training sets: one-sided selection. In: Icml, vol 97, pp 179–186. Citeseer
11. Smith MR, Martinez T, Giraud-Carrier C (2014) An instance level analysis of data complexity. Mach Learn 95(2):225–256
12. Akbani R, Kwek S, Japkowicz N (2004) Applying support vector machines to imbalanced datasets. In: European conference on machine learning, pp 39–50. Springer
13. Alejo R, García V, Sotoca JM, Mollineda RA, Sánchez JS Improving the performance of the RBF neural networks trained with imbalanced samples. In: International work-conference on artificial neural networks, pp 162–169 (2007). Springer
14. Branco P, Torgo L, Ribeiro RP (2016) A survey of predictive modeling on imbalanced domains. ACM Comput Surv (CSUR) 49(2):1–50
15. Ertekin S, Huang J, Bottou L, Giles L (2007) Learning on the border: active learning in imbalanced data classification. In: Proceedings of the 16th ACM conference on conference on information and knowledge management, pp 127–136
16. Ertekin S, Huang J, Giles CL (2007) Active learning for class imbalance problem. In: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval, pp 823–824
17. Zhuang L, Dai H (2006) Parameter optimization of kernel-based one-class classifier on imbalance learning. J Comput 1(7):32–40
18. Wang S, Minku LL, Yao X (2014) Resampling-based ensemble methods for online class imbalance learning. IEEE Trans Knowl Data Eng 27(5):1356–1368
19. Krawczyk B, Minku LL, Gama J, Stefanowski J, Woźniak M (2017) Ensemble learning for data stream analysis: a survey. Inf Fusion 37:132–156
20. Johnson JM, Khoshgoftaar TM (2019) Survey on deep learning with class imbalance. J Big Data 6(1):27
21. Schölkopf B, Platt JC, Shawe-Taylor J, Smola AJ, Williamson RC (2001) Estimating the support of a high-dimensional distribution. Neural Comput 13(7):1443–1471

22. Liu X-Y, Wu J, Zhou Z-H (2008) Exploratory undersampling for class-imbalance learning. IEEE Trans Syst Man Cybern Part B (Cybern) 39(2):539–550
23. Liu Z, Cao W, Gao Z, Bian J, Chen H, Chang Y, Liu T-Y (2020) Self-paced ensemble for highly imbalanced massive data classification. In: 2020 IEEE 36th international conference on data engineering (ICDE), pp 841–852. IEEE
24. Lin T-Y, Goyal P, Girshick R, He K, Dollár P (2017) Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp 2980–2988
25. Dong Q, Gong S, Zhu X (2018) Imbalanced deep learning by minority class incremental rectification. IEEE Trans Pattern Anal Mach Intell 41(6):1367–1381
26. Cui Y, Jia M, Lin T-Y, Song Y, Belongie S (2019) Class-balanced loss based on effective number of samples. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 9268–9277
27. Cao K, Wei C, Gaidon A, Arechiga N, Ma T (2019)Learning imbalanced datasets with label-distribution-aware margin loss. In: Advances in neural information processing systems, pp 1567–1578
28. Li M, Zhang X, Thrampoulidis C, Chen J, Oymak S (2021) Autobalance: optimized loss functions for imbalanced data. Adv Neural Inf Process Syst 34:3163–3177
29. Kini GR, Paraskevas O, Oymak S, Thrampoulidis C (2021) Label-imbalanced and group-sensitive classification under overparameterization. arXiv:2103.01550
30. Ren M, Zeng W, Yang B, Urtasun R (2018) Learning to reweight examples for robust deep learning. arXiv:1803.09050
31. Shu J, Xie Q, Yi L, Zhao Q, Zhou S, Xu Z, Meng D (2019) Meta-weight-net: learning an explicit mapping for sample weighting. In: Advances in neural information processing systems, pp 1919–1930
32. Lee HB, Lee H, Na D, Kim S, Park M, Yang E, Hwang SJ (2019) Learning to balance: bayesian meta-learning for imbalanced and out-of-distribution tasks. arXiv:1905.12917
33. Liu Z, Wei P, Jiang J, Cao W, Bian J, Chang Y (2020) Mesa: boost ensemble imbalanced learning with meta-sampler. Adv Neural Inf Process Syst 33:14463–14474
34. Huang C, Li Y, Loy CC, Tang X (2016) Learning deep representation for imbalanced classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5375–5384
35. Wang Y, Gan W, Yang J, Wu W, Yan J (2019) Dynamic curriculum learning for imbalanced data classification. In: Proceedings of the IEEE international conference on computer vision, pp 5017–5026
36. Kang B, Xie S, Rohrbach M, Yan Z, Gordo A, Feng J, Kalantidis Y (2019) Decoupling representation and classifier for long-tailed recognition. arXiv:1910.09217
37. Yang Y, Zha K, Chen Y-C, Wang H, Katabi D (2021) Delving into deep imbalanced regression. arXiv:2102.09554
38. Street WN, Kim Y (2001) A streaming ensemble algorithm (sea) for large-scale classification. In: Proceedings of the 7th ACM SIGKDD international conference on knowledge discovery and data mining, pp 377–382
39. Furao S, Ogura T, Hasegawa O (2007) An enhanced self-organizing incremental neural network for online unsupervised learning. Neural Netw 20(8):893–903
40. Shen F, Hasegawa O (2008) A fast nearest neighbor classifier based on self-organizing incremental neural network. Neural Netw 21(10):1537–1547
41. Brodersen KH, Ong CS, Stephan KE, Buhmann JM (2010) The balanced accuracy and its posterior distribution. In: 2010 20th international conference on pattern recognition, pp 3121–3124. IEEE
42. Batista GE, Prati RC, Monard MC (2004) A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explor Newslett 6(1):20–29
43. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140
44. Seiffert C, Khoshgoftaar TM, Van Hulse J, Napolitano A (2009) Rusboost: a hybrid approach to alleviating class imbalance. IEEE Trans Syst Man Cybern-Part A Syst Hum 40(1):185–197
45. Alcala-Fdez J, Fernandez A, Luengo J, Derrac J, Garcia S, Sanchez L, Herrera F (2011) Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Mult Valued Logic Soft Comput 17(2–3):255–287
46. Dua D, Graff C (2017) UCI machine learning repository. http://archive.ics.uci.edu/ml