Adaptive Multi-objective Swarm Crossover Optimization for Imbalanced Data Classification

Jinyan Li[1(✉)], Simon Fong[1(✉)], Meng Yuan[1], and Raymond K. Wong[2]

[1] Department of Computer Information Science, University of Macau, Macau SAR, China
{yb47432, ccfong, mb55525}@umac.mo

[2] School of Computer Science and Engineering, University of New South Wales, Kensington, Australia
wong@cse.unsw.edu.au

Abstract. Training a classifier with imbalanced dataset where there are more data from the majority class than the minority class is a known problem in data mining research community. The resultant classifier would become under-fitted in recognizing test instances of minority class and over-fitted with overwhelming mediocre samples from the majority class. Many existing techniques have been tried, ranging from artificially boosting the amount of the minority class training samples such as SMOTE, downsizing the volume of the majority class samples, to modifying the classification induction algorithm in favour of the minority class. However, finding the optimal ratio between the samples from the two majority/minority class for building a classifier that has the best accuracy is tricky, due to the non-linear relationships between the attributes and the class labels. Merely rebalancing the sample sizes of the two classes to exact portions will often not produce the best result. Brute-force attempt to search for the perfect combination of majority/minority class samples for the best classification result is NP-hard. In this paper, a unified preprocessing approach is proposed, using stochastic swarm heuristics to cooperatively optimize the mixtures from the two classes by progressively rebuilding the training dataset is proposed. Our novel approach is shown to outperform the existing popular methods.

Keywords: Class rebalancing · Swarm optimization · Classification

1 Introduction

Imbalance dataset is referred to the phenomenon where there are far more samples in one class than in the other. Some data mining applications that would have to deal with imbalance datasets are those typically would have to be trained with large amount of common samples but with limited rare samples. They include big data analytics and text mining [1], forecasting natural disasters [2], fraud detection in transactions [3], target identification from satellite radar images [4], classifying biological anomalies [5] as well as computer-assisted medical diagnosis and treatment [6], just to name a few.

Imbalanced data classification has long been an important and challenging problem in data mining and machine learning [7]. Conventional supervised learning algorithms by greedy search are usually designed to embrace the imbalanced dataset without regards to the class balance ratio by default. Most original classification model induction algorithms were designed without the consideration of imbalance issue initially. Those trained models suffer from overfitting from the sheer volume of majority training data; the recognition power for identifying rare test samples is limited due to the lack of sufficient training (known as underfitting) given few minority samples are available. Additional performance metrics which are used to assess or judge whether a classification model is incompetent owing to imbalance training go beyond just accuracy. Some useful metrics which are based on the counts of true-positive, false-positive etc. include G-mean [8], F1 measure [9], Kappa statistics [10], AUC/ROC [11], Matthews correlation coefficient [12] and Balance error rate [13] that have been used in the literature.

Pre-processing styled rebalancing schemes have been proposed in the past, mainly in the aspects of artificially inflating the minority class data, resampling down the volume of the majority class data, or a combination of the two. It was already shown [10] that merely matching the quantities of the majority and minority data to equal, does not yield the highest possible classification performance.

In this paper, an adaptive rebalancing model as a preprocessing tool is proposed, by considering the drawbacks of the current methods for solving imbalanced classification problem. What data mining users desire as the features of an ideal rebalancing tool, in observation of the above-mentioned limitations are: high performance that is not only in accuracy but in other reliability measures as well, free of parameter calibration, joint rebalancing actions by increasing and decreasing the minority and majority samples respectively, and be able to complete the dual actions till reaching the best possible performance within a reasonable time.

Given a potentially very large number of instances in the original dataset, finding the best ratio between two majority and minority classes of data is a challenging combinational optimization problem. Without resorting to brute-force, swarm optimization is applied on each aspect of rebalancing – one on searching for the appropriate amount of majority instances, and the other one on estimating the best combo of control parameters (the intensity and how far that the neighbors of the minority samples are to be fabricated) with respect to enlarging the minority population size. Our proposed rebalancing method couples these two optimizations as an unified iterative approach which will progressively enhance the mixtures of the optimized data from the two swarm optimizations by crossing over their optimized results generation after generation until a good quality dataset is produced. This unified rebalancing approach is called Adaptive Multi-Objective Swarm Crossover Optimization (AMSCO), within which the optimization of majority instances is called Swarm Instance Selection (SIS), the optimization of minority instances is called Optimized Synthetic Minority Oversampling Technique (OSMOTE). The overall design of AMSCO is shown in Fig. 1.

In this new approach, Particle Swarm Optimization (PSO) algorithm is chosen as the core optimizer whose searching particles represent the solution candidates. The original dataset, after it was loaded for the first time, will become the current dataset, and be checked with respect to its quality by inferring a candidate classifier from it. Until the

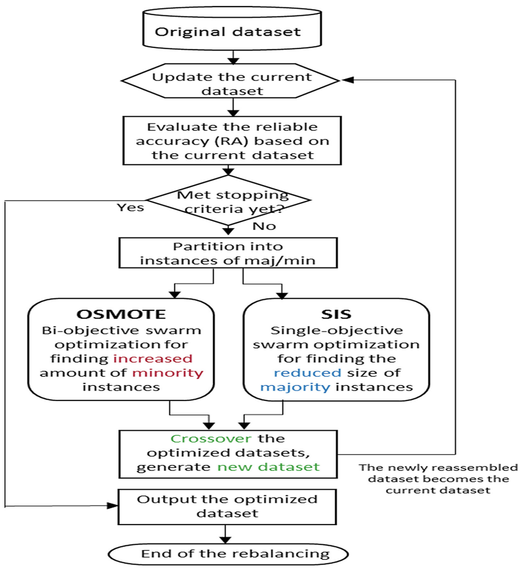
Original dataset
Update the current dataset
Evaluate the reliable accuracy (RA) based on the current dataset
Met stopping criteria yet?
Yes
No
Partition into instances of maj/min
OSMOTE
Bi-objective swarm optimization for finding increased amount of minority instances
SIS
Single-objective swarm optimization for finding the reduced size of majority instances
Crossover the optimized datasets, generate new dataset
The newly reassembled dataset becomes the current dataset
Output the optimized dataset
End of the rebalancing

Fig. 1. Flow chart of AMSCO

performance of the candidate classifier meets the expected requirement, the current dataset will be subject to two parallel swarm optimizations for optimally increasing the minority samples and decreasing the majority samples. The two swarms operate independently because their candidate solutions are different in nature. Their outputs, however, are crossed over by selectively merging instances from the most competent optimized datasets into one that is framed by the size of the original dataset. The selected dataset in turn becomes the current dataset when the optimization cycle iterates. The dataset is checked by the criteria and passed to two swarm operations again, if it is still not good enough. Decision tree is used as the classification algorithm here which works like a wrapper approach in testing the goodness of the current dataset.

The advantages of AMSCO are: (1) progressively refining the dataset by rebalancing the instances through swarm optimizations to its best; (2) the imbalanced dataset becomes balanced in such a way that both accuracy and reliability are maximized; (3) the size of the resultant balanced dataset is controlled so the original dataset size is approximately preserved; (4) as a by-product, the base-learner used in the wrapper approach is trained upon finding the well balanced dataset. The classifier trained by the final dataset (which supposed to offer the best performance) can be instantly put into use; (5) compared to the tightly coupled swarm optimization techniques and other conventional rebalancing methods by linear search, AMSCO has superior in fast computational speed, accuracy and reliability.

The remainder of this paper is structured as follows. Section 2 reviews popular approaches that have been successfully employed in imbalanced dataset classification to certain extent. In Sect. 3, we elaborate the design of AMSCO. The benchmark datasets, experiment and its results are described in Sect. 4. Section 5 concludes this paper.

2 Related Works

Class rebalancing techniques can be summed up as two major categories. The first group concerns about data level, which re-sampling the two classes samples [14]. The second level pertains to alter the classification algorithms for imbalanced data classification. Researches proposed various resample techniques in data level. Random over-sampling [15] and under-sampling [16] are the simplest method. The former augments minority scale through copying its samples, and the latter randomly delete majority class samples to realize a balance. Typical under-sampling and over-sampling techniques may subside the disadvantage of information vacancy and over learning [17] respectively. One-side selection [18] techniques categorized majority class samples and eventually find out the safety samples to fulfill the under-sampling. The most popular sampling method is SMOTE (Synthetic Minority Oversampling Technique) [19] that usually achieve effective performance. The principle is letting the algorithm fabricate extra minority data into the dataset through observing and analyzing the characteristics of minority class sample' spatial structure. Assuming the oversampling rate is N (Eq. (1) synthesizes N times new minority class samples) and each minority class sample $x_i \in S_{minority}$. The other parameter k is used by the algorithm to examine k neighbors of x_i in minority class samples, then to randomly select x_t from the k neighbors using Eq. (1) to generate the synthetic data $x_{new,N}$:

$$x_{new,N} = x_i + rand[0,1] \times (x_t - x_i) \tag{1}$$

In Eq. (1) *rand* [0, 1] generates a random number between 0 and 1. N and k influence SMOTE to generate a suitable number of characteristic minority class samples. The second level of approaches solves the class imbalance problem during the training stage. It contains ensemble based techniques [20] and cost-sensitive learning approaches [21]. The basic idea of ensemble learning is that, a strong classifier will be voted and integrated by a series of weak classifiers after several rounds of iteration. The commonly used ensemble learning methods are bagging [22], boosting [23], and random forest [24]. SMOTEBoost [25] approach combines Adaboosting and SMOTE, and it offers high performances. Cost-sensitive learning assigns a different weight to each part of confusion matrix by the cost matrix. In general, the cost of misclassified minority class is the largest, thus in order to obtain a results with minimum cost, the classifier will be bias to minority class. The distinguished cost-sensitive learning algorithms are associated with boosting [26, 27] or Support Vector Machines (SVM) [28] to tackle class imbalance problem.

3 Design of AMSCO

The proposed AMSCO uses mainly two swarm optimization processes that run independently, for fixing the exceeding majority data instances and shortage of minority data instances respectively. The overall idea of AMSCO is to smash up the original dataset and to resemble it back again using only the qualified instances selected by the two swarm optimization processes. This is done by first dividing the original dataset into two groups – one group contains instances which are purely of majority class and the other group purely of minority class. Through the swarm search processes, the instances are randomly encoded into search particles which represent some candidate solutions, and the particles move iteratively from random positions towards some global optimum. The instances being selected from the swarm processes are being heuristically enhanced, leaving only some fittest ones at the end.

At the end of each iteration, the qualified instances which are represented by the fittest particles so far are gathered in buffers, in preparation of packing them into a new dataset by crossover operation. The new dataset after packed, will be taken as the current dataset, and subject to the optimization in the new iteration. By this way, the current dataset evolves in improving its quality, offering increasingly higher fitness iteration after iteration. Eventually the most refined dataset is outputted as the best dataset at the end of the whole preprocessing operation, by which the imbalance problem is solved and the final dataset is ready to induce a classifier that will have the best possible performance.

The intermediate dataset which is supposed to be the best thus far at each round of optimization, is assembled from four possible datasets. The compositions of the datasets are shown in Fig. 2.

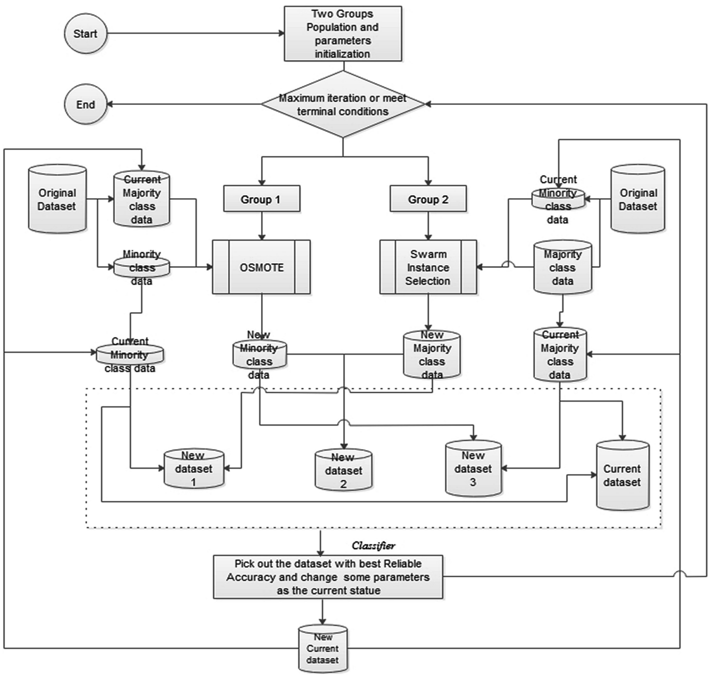
Start
Two Groups Population and parameters initialization
End
Maximum iteration or meet terminal conditions
Original Dataset
Current Majority class data
Minority class data
Group 1
OSMOTE
Group 2
Swarm Instance Selection
Current Minority class data
Original Dataset
Majority class data
Current Minority class data
New Minority class data
New Majority class data
Current Majority class data
New dataset 1
New dataset 2
New dataset 3
Current dataset
Classifier
Pick out the dataset with best Reliable Accuracy and change some parameters as the current statue
New Current dataset

Fig. 2. Components and data around the AMSCO

The four possible datasets, which are candidates to be crossed over and packed into a new generation of current dataset are the three combinations of optimized majority dataset, optimized minority datasets, and the current dataset. From Fig. 2, they are mixed and enumerated as follow: (1) New dataset 1 = New Majority class data (after optimized by SIS on the majority instances) + Current Minority class data; (2) New dataset 2 = New Majority class data (after optimized by SIS on the majority instances) + New Minority class data (after optimized by OSMOTE on minority instances); (3) New dataset 3 = New Minority class data (after optimized by OSMOTE on minority instances) + Current Minority class data; and, (4) New dataset 4 = Current Majority class data + Current Minority class data.

The rationale behind crossing over different majority and minority portions of the optimized data is hoping to generate the new dataset that may have the perfect balance/ratio by fusing the best from the two optimized portions. Since the perfect ratio between the majority and minority instances is not known in advance, and it is short of a deterministic way (other rather brute-force) to compute the ratio, we resort to iterative heuristics in summing up the best instances from both majority and minority classes found so far for the improved version of dataset. It is worth noting a few remarks that: (1) An ideal balanced dataset is required to have both portions of majority and minority although the exact ratio is elusive; so in the four combos of new datasets, there must be certain instances that come from majority and minority classes; (2) The swarm optimization is purely probabilistic, there may be chances that both OSMOTE and SIS yield no better quality instances (none or even negative improvement), so New dataset 4 is needed as a backup; (3) The terminal conditions are the result doesn't change in the past defined iterations (convergence);The length of new minority class samples cannot too bigger than the new majority class samples (2 times are used in the experiment); or it achieves the maximum iteration (4) The sizes of the optimized datasets do vary from time to time; there are chances that they are shorter than the original dataset. When this happens the full dataset of the highest fitness will fill in, followed by all of the second best, and so forth until the new generation of current dataset matches the size of the original dataset. This is to prevent the assembled dataset overly shrinking or enlarging along the way. (5) OSMOTE and SIS are separate processes since the ways how the instances are selected for reduction and replication are different, though they share the same objectives and similar fitness evaluation functions.

OSMOTE and SIS which are the core of the proposed rebalancing model are based on Swarm intelligence optimization algorithm [29] which iteratively evolves a solution from randomly picked values to some global optimal result. Specifically, Particle Swarm Optimization (PSO) algorithm [30] is used in OSMOTE and SIS whose search agents imitate the flying patterns of birds. Assuming there is a population $X = (X_1, X_2, \ldots, X_n)$ which is grouped by n particles in D dimensional search space, the ith particle in this space is expressed as a vector X_i with D dimension, $X_i = (x_{i1,} x_{i2}, \ldots, x_{iD})^T$, and the position of the ith particle in the search space represents a potential solution. As the objective function, the program can calculate the corresponding fitness of position X_i of each particle, where the speed of the ith particle is $V_i = (V_{i1}\ ,V_{i2}, \ldots, V_{iD})^T$, the extremum value of each agent is $P_i = (P_{i1}, P_{i2}, \ldots, P_{iD})^T$ and the extremum of the population is $P_g = (P_{g1}, P_{g2}, \ldots, P_{gD})^T$. In the process of iteration, the extremum value of each agent and the population will update their position and speed [31]. Equations (2) and (3) show the mathematical process as follows:

$$V_{id}^{k+1} = \omega * V_{id}^{k} + c_1 r_1 \left(P_{id}^{k} - X_{id}^{k}\right) + c_2 r_2 \left(P_{id}^{k} - X_{id}^{k}\right) \tag{2}$$

$$X_{id}^{k+1} = X_{id}^{k} + V_{id}^{k+1} \tag{3}$$

In the Eq. (2), ω is inertia weight; d = 1, 2, …, D; i = 1, 2, …, n; k is the current iteration time; c_1 and c_2 are non-negative constants as the velocity factor, r_1 and r_2 are random values between 0 to 1 and V_{id} is the particle speed.

Cooperation evolution draws thought from population coordination theory in the ecology. It simulates the mutual influence and mutual restriction between various populations in nature to strengthen performances of each population and global [32]. Generally, in multiple swarm collaboration algorithm, particles randomly be divided into M sub-groups, S_i, $1 \leq i \leq M$, collaboration through the information interaction between populations. There are three kinds of evaluation rules:

$$\text{ER1} : V_i = \omega * V_i + c_1 r_1 \left(P_i^{s} - X_i\right) + c_2 r_2 (P_i^{sg} - X_i) \tag{4}$$

$$\text{ER2} : V_i = \omega * V_i + c_1 r_1 \left(P_i^{s} - X_i\right) + c_2 r_2 (P_i^{sg} - X_i) + c_3 r_3 (P_i^{g} - X_i) \tag{5}$$

$$\text{ER3} : V_i = \omega * V_i + c_1 r_1 \left(P_i^{s} - X_i\right) + c_2 r_2 (P_i^{g} - X_i) \tag{6}$$

In Eqs. (4) to (6) P_i^{s}, P_i^{sg} and P_i^{g} are respectively stands for the best value of particle X_i, the best fitness of particle X_i's sub-group and the best fitness of all groups. Thus, multiple swarms collaboration method founded on reducing a problem into several or more sub-problems. The sub-groups demarcate and search the searching space in parallel and though communicating and sharing the best information between each sub-groups to find out the final global best solution for a short time. In the case of AMSCO, two sub-groups are constructed, for cooperatively optimizing two classes of data in OSMOTE and SIS, the best information are crossed over, until a final global best solution is obtained as a well-balanced dataset. More information regarding OSMOTE and SIS will be explained in the following sub-sections.

The fitness is defined as a product of accuracy and Kappa or Kappa statistics [33]. Kappa is chosen to estimate the credibility of a classification model. When a classifier suffers from imbalanced dataset, it has a sign of high accuracy but a low value (zero or even negative) of Kappa. Kappa is an efficient indicator to be fairly reflect the consistency of test data and the dependability of the classification model, so as to investigate whether the performances fall into a secure area. There are six degrees of interpretation for Kappa outcome ranging between −1 and 1 in mathematics [33]. Subzero part denotes that this model is worthless, and each of the other five levels is segmented by a 0.2 interval. These areas respectively stand for the strength of agreement in poor, slight, fair, moderate, substantial and almost prefect. The model typically has some credibility when its Kappa value exceeds over 0.4 [10, 34], and the credibility will increase with the improvement of Kappa statistics. For the fitness which is used to represent the goodness of the classification model, a composite performance criterion called Reliable Accuracy (RA) is defined, where $RA \propto (Accuracy \times Kappa)$, of an induced model. The fitness function therefore is the performance evaluation of a decision tree, generated from a given dataset. Other classification algorithms could be optionally used in lieu of decision tree, though it is used because of its popularity. The fitness is *RA* that depends on the accuracy and Kappa which are defined as follow:

$$RA = \frac{P_o^2 - P_o P_c}{1 - P_c} \tag{7}$$

$$Accuracy = \frac{TP + TN}{P + N} \tag{8}$$

$$Kappa = \frac{P_o - P_c}{1 - P_c} \tag{9}$$

$$P_o = Accuracy = \frac{TP + TN}{c^+ + c^-} \tag{10}$$

$$P_c = \frac{(TP + FP) \times (TP + FN) + (FN + TN)(FP + TN)}{(c^+ + c^-)^2} \tag{10}$$

where *TP*, *TN*, *FP*, *FN*, C^+ and C^- are the counts of true-positive, true-negative, false-positive, false-negative, instances of positive/majority class and instances of negative/minority class respectively. P_o is the measure of the percentage of agreement, and P_c is the chance of agreement.

3.1 Optimized SMOTE for Over-Sampling Minority Instances

OSMOTE is extended from Synthetic Minority Oversampling Technique (SMOTE) [19], which is one of the most popular methods to over-sample the minority instances for rebalancing an imbalanced dataset. Its basic idea is to fabricate extra minority data into the dataset by inserting synthetic samples along the line segments connecting any or all of the *k* minority class nearest neighbors in the data space. In SMOTE, two parameters are required and they need to be manually set. One is the over-sampling rate of *N* which tells the algorithm to synthesize *N* times new minority class samples, e.g. $N = 2$ means the minority class data are going to be doubled. The other parameter *k* is used by the algorithm to inspect *k* nearest neighbours of each minority data to generate the synthetic data.

This method can effectively create synthetic examples increasing the population of minority instances, rather than by over-sampling with replacement. Depending upon the extent of over-sampling required by *N*, certain neighbors from the *k* nearest neighbors are chosen randomly. One obvious drawback is the lack of guideline recommending what the values of $N \in [1 \ldots \infty]$ and $k \in [1 \ldots (C^+ \text{ and } C^-)]$ should be used, for generating a rebalanced dataset that gives the highest classification performance. In many cases, these two parameters are arbitrarily chosen; though the resultant classification performance is improved, it is not maximized.

In our OSMOTE approach, the two parameters of the system, N which indicates the target amount of dataset after over-sampled, and k the range of reference neighbors for duplicating the minority data vectors, are to be optimized using PSO. Each particle of PSO swarm, p, is coded as a candidate solution with a pair of values within the ranges of N and k. So $p = \langle v, \kappa \rangle$, where $v \in [1 \ldots \infty]$ and $\kappa \in [1 \ldots (C^{+}$ and $C^{-})]$. The fitness function is the resultant decision tree built on a dataset after over-sampled by SMOTE with $N = v$ and $k = \kappa$. The RA performance is treated as fitness in this case, for evaluating the goodness of the chosen p such that *fitness* = *RA* = *fitness_evaluation*(*DT*(*p*)).

OSMOTE is aimed at improving both accuracy and Kappa where these two values fluctuate dynamically during the search process. As such, it is a dynamic multi-objective algorithm. Without considering complex situations like Pareto, this dual objective algorithm actually just tries to maintain a certain high accuracy level while maximizing the Kappa which is deemed more importantly as credibility to its highest. Figure 3 shows a snapshot of the fluctuation patterns of accuracy and Kappa as the optimization progresses. In general, accuracy and Kappa follow similar trends but of different intensities in swinging ups and downs; and the overall trends are on the rise. One can see that in this example, accuracy and Kappa have both reached a very high value of approximately 1, at the 310th cycle of iteration. Since the two objectives are not opposing each other, a special type of optimization called the non-inferior set tactics [35] is adopted here and customized for this specific rebalancing task.

It first collects all the possible solutions of this multi-objective problem. These solutions in non-inferior set are satisfied with several update criteria. Three update criteria are used here regulating the OSMOTE for the particles evolution, optimization and convergence.

(1) Both accuracy and Kappa of the new particle must be better than the existing one;
(2) Either one of the accuracy or Kappa of the new particle must be better than the existing one, as well as the defined tolerance is larger than the absolute value of difference of the other measures such as F1, ROC and BER;
(3) The current threshold value of Kappa of the new particle must be greater than the older particle's Kappa value.

If a new particle meets any of the above three criteria, it will replace the current one into the next generation. Otherwise, this particle will follow its trajectory and move to a neighbor position. This algorithm progressively lifts up Kappa through updating by the

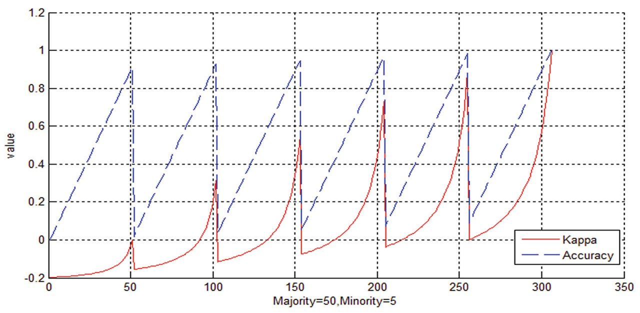

Fig. 3. Snapshot of fluctuating values of accuracy and Kappa during OSMOTE of a imbalanced dataset

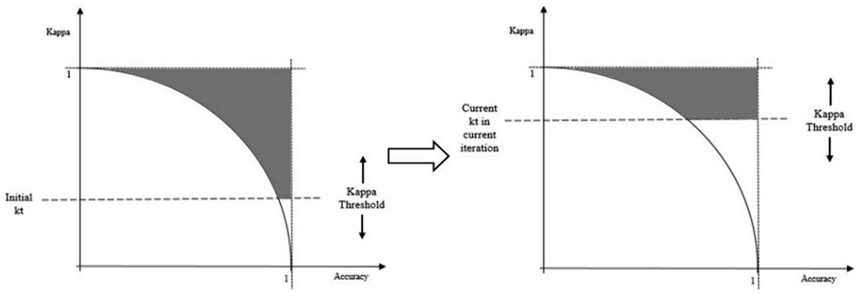

Fig. 4. Illustration of non-inferior region in the search space

constraint conditions. At the start, the initial threshold of Kappa, k_t, is set at 0.4 that is the second bottom confidence level of Kappa. In the last phase of each iteration, the average Kappa value in current non-inferior set will compare with the latest threshold value, the threshold will increase further if the average value increases, and vice versa. By doing so, the non-inferior region will progressively be reduced as the Kappa threshold is lifting up. Figure 4 illustrates this concept; the shaded region is the search space within which the particles of PSO move and scout for the highest values of accuracy and Kappa. The search time is significantly reduced when the non-inferior region (search space) is closing up by the variable, threshold Kappa. The ultimate objective of SMOTE is to achieve finding the maximum value of the reliable accuracy (*RA*) which is dependent both on Kappa and accuracy. It is noted that the constraints in preparing the search over the current dataset are based on these conditions: $100 \leq N \leq r \times size(majority\, data,\ C^{+})$, and $2 \leq k \leq size(minority\, data,\ C^{-})$ where r is the ratio of majority and minority classes in the original data.

The pseudo code of OSMOTE is listed in Algorithm 1.

Algorithm 1: Optimized SMOTE (OSMOTE)

```
Specify a Swarm intelligence algorithm $A$ and a Classifier $C$
// Initialize the population of the $A$ algorithm and the other related parameters
Define the floor value of Kappa: $k_t$
Define the scope of $K$ and $N$
 1: Load dataset
 2: Initialize the particle's position and speed // through the SMOTE and $C$ to get the Current Kappa and
    Current Accuracy as the initial population fitness
 3: for $i$= 1: Maximum number of iteration
 4:       Radom select the global best ($g\_b$) from the non-inferior solution
 5:       Update the particles' speed and position //as the update condition, the new local best replace the older
 6:       if particle satisfied with the conditions
 7:      change the positions
 8:       else randomly select the position
 9:       end if
10:     Combine the new and old non-inferior solution set
11:     Filter the final results set by the update condition
12:       if there is no Kappa value bigger than $k_t$ // update Kappa threshold
13:           reduce $k_t$
14:       elseif the mean of Kappa in the solution set is bigger than
15:             $k_t$+step length
16:             increase $k_t$
17:     endif
18: end for
19: Output: Get the final global best non-inferior solution
```

3.2 Swarm Instance Selection (SIS) for Reducing Majority Instances

Along with OSMOTE that syntheses just the correct amount of minority data, Swarm Instance Selection (SIS) is there for reducing majority instances to the appropriate amount. The outputs of these optimizations which have been rebalanced at the best efforts with respect to its majority/minority class, would be used as potential candidates for restructuring a new output data in the crossover.

The SIS approach is an evolutionary version of under-sampling method [36] which could effectively select the useful majority samples. It is known that there is no rigid ratio or partitioning rule for labelling which instances are useful or otherwise. The relations between different combinations of majority and minority class data with the predicted class are non-linear. Therefore, a wrapper approach, using the classifier (DT in our case) to tell which combinations of majority instances and minority instances can find us the best combination which offers the highest classification performance from the base learner.

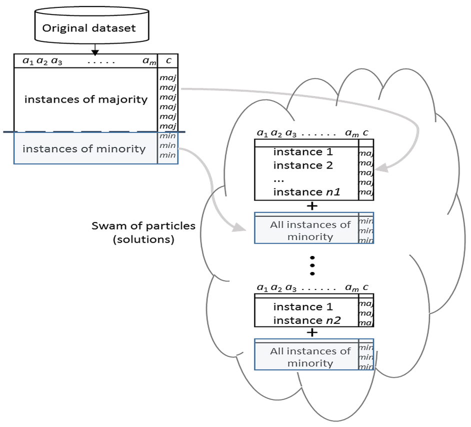

Fig. 5. Majority instances are selected into swarm particles in SIS

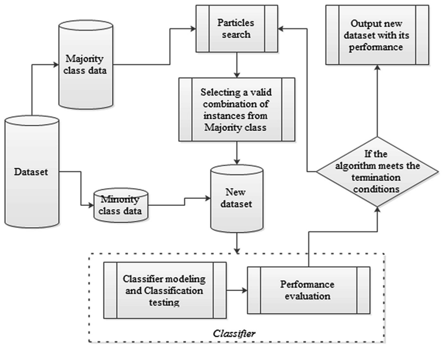
Majority class data
Particles search
Output new dataset with its performance
Selecting a valid combination of instances from Majority class
Dataset
If the algorithm meets the termination conditions
Minority class data
New dataset
Classifier modeling and Classification testing
Performance evaluation
Classifier

Fig. 6. Flow chart of Swarm Instance Selection

Given the sheer volume of instance, brute-force is not feasible. That is the reason why the majority data selection process is devised to be optimized by stochastic swarm search. In SIS each PSO particles encodes a candidate solution as two parts, one of which is a collection of instances randomly selected from the majority instances of the current dataset (or original dataset if it is the first cycle). Figure 5 shows the random selection concept. The other part of the particle is the whole of the minority group of instances from the current (or original) dataset. Like chromosomes in genetic algorithm, the particles will change in their collections towards a global best solution represented by the maximum performance from the base learner.

The size of the PSO particle is $Size(p) = Random\_selection(Data_{maj}) + Data_{min}$. The size of the swarm particle is also constrained by the minimum and maximum limits as follow: $Min \geq r \times Size(Data_{min})$, and $Max < r \times Size(Data_{maj})$, where r is the ratio of the minority data over majority data of the current dataset. Figure 6 shows the working procedure of Swarm Instance Selection approach.

In SIS, each randomly selected group from the majority class data will combine with all of the minority class data to create a new candidate dataset, which will be used to build a decision tree for performance testing. As SIS runs, the particles will eventually choose the best combination of majority class samples through the comparison of each particle's fitness over some time. Compared with other under-sampling methods, SIS incurs certain overhead in computational speed. However, SIS often can achieve a better solution (globally best) at the end, without using the naïve brute-force.

Both OSMOTE and SIS have the advantage of staying focused in tackling the imbalance problem with respect to one of their classes, through cooperation hand-in-hand via the crossover operation at the end of each iteration. Their respective class data are rebalanced while considering the best sampled data that were generated thus far. Conventional over-sampling or under-sampling techniques however focus only on their individual class size, neglecting about how the other class size and instances within might have been evolved or improved. The conventional over-sampling and under-sampling are conducted independently without regards of their counter-part. This is the prime advantage and unique difference between AMSCO and the traditional rebalancing algorithms. The pseudo codes of SIS is listed as follow:

```
Algorithm 2: Swarm Instance selection (SIS)
Specify a Swarm intelligence algorithm $A$ and a Classifier $C$
 1: Load dataset
 2: Initialize the population of the $A$ algorithm and the other
      related parameters
 3: Using dataset to initialize the particle's position and speed
      //through classifier $C$ get the Current Reliable Accuracy as the initial population fitness
 4: for $i$= 1: Maximum number of iterations
 5:       select the global best
 6:       if the current best less than the new
 7:       replace
 8:       else
 9:             randomly select the best
10:       end if
11:       if the terminal condition is met
12:             break
13:       end if
14: end for
15: Output: get the best combination of the instances to build a new dataset with its performances
```

4 Experiment

Eight rebalancing methods are used in the experimentation to evaluate the performance of the proposed methods in this paper versus the traditional ones. The first to compare is basic classification algorithm, decision tree, without any pre-processing of rebalancing, three methods are commonly used in algorithm level to the bias of classifier, and the other four are sampling methods which include the traditional over-sampling methods with the three swarm rebalancing algorithms.

- Decision Tree (DT): one of the most popular classifiers. It often shows good performance in imbalanced dataset classification. Many papers in the workshop of ICML 2013 investigated C4.5 with imbalanced dataset and it effectively increases the performance of sampling techniques from imbalanced dataset [37].
- Bagging: Bagging method + DT.
- AdaBoost.M1 (AdaBM1): AdaBM1 + DT.AdaBoost.M1 stands for Discrete AdaBoost [38], which is classical boosting method.
- Cost-sensitive (CTS): CTS + DT. The values of cost matrix respectively matching to the elements of confusion matrix, *TP* and *TN*'s cost are zero. *FN* denotes the misclassified minority class samples; its cost is 10. Misclassified majority class samples are *FP* which cost half of *FN*.
- Synthetic Minority Over-Sampling Technique (SMOTE): SMOTE + DT. The two parameters are manually selected taking the default values. The average value of its ten times operation is used as the final performance.
- Swarm Instance Selection algorithm (SIS): SIS + DT.
- Optimized SMOTE Algorithm (OSMOTE): OSMOTE + DT.
- SIS-OSMOTE: SIS-OSMOTE + DT. The two optimization processes are placed sequentially. OSMOTE will load the new dataset from SIS as the Kappa of SIS is greater than 0.3 or it reaches the maximum iteration. In this case, SIS and OSMOTE work independently without crossing over their optimized datasets.
- AMSCO: Adaptive Multi-Objective Swarm Crossover Optimization. Its logics are in Fig. 1.

For fair comparison, the maximum iteration of bagging, AdaBM1, SIS, OSMOTE, SIS-OSMOTE and AMSCO is standardized at 100. The amount of base classifiers of Bagging, the population of SIS, OSMOTE, SIS-OSMOTE and AMSCO is 20. Stratified 10-cross-validation is used as the verification and testing method. There are 30 imbalanced datasets being used for benchmarking as they are selected from 100 binary class imbalanced dataset from KEEL [39]. These datasets have low Kappa statistics. Table 1 lists the characteristics of these datasets, Maj and Min respectively denotes majority class and minority class, Imb.r is the imbalance ratio (majority/minority). The imbalance ratio ranges from 1.87 to 129.44. The simulation software is programmed by Matlab 2014b. The simulation computing platform is CPU: E5-1650 V2 @ 3.50 GHz, RAM: 32 GB.

For easy comparison, the average performance values across the 30 benchmark datasets are charted as histograms featuring vis-a-vis all the rebalancing algorithms, the swarm and the traditional. The results unanimously point to an observation that AMSCO has an edge over the performances outperforming the rest of them. Figure 7 shows the overall performance comparison chart.

Observing from the results, it is apparent that, there are rooms for improvement for decision tree that classifies the original datasets without any rebalancing, both the accuracy and Kappa are the lowest among all. In the four typical methods, SMOTE performs relatively well. In the ensemble learning, AdaBM1 is better than Bagging. Cost-sensitive learning is more effective than Bagging. Our individual class optimization methods using Swarms are much better than the traditional methods, demonstrating the usefulness of swarm optimization, except for SIS is worse than

Table 1. Characteristic of datasets used in experiment

Dataset	#Samples	Maj	Min	Imb.r	Dataset	#Samples	Maj	Min	Imb.r
abalone-17_vs_7-8-9-10	2338	2280	58	39.31	poker-8-9_vs_6	1485	1460	25	58.4
abalone-19_vs_10-11-12-13	1622	1590	32	49.69	poker-8_vs_6	1477	1460	17	85.88
abalone-20_vs_8-9-10	1916	1890	26	72.69	poker-9_vs_7	244	236	8	29.5
abalone-21_vs_8	581	567	14	40.5	vehicle1	846	629	217	2.9
abalone19	4174	4142	32	129.44	vehicle3	846	634	212	2.99
abalone9-18	731	689	42	16.4	winequality-red-3_vs_5	691	681	10	68.1
cleveland-0_vs_4	177	164	13	12.62	winequality-red-8_vs_x6-7	855	837	18	46.5
flare-F	1066	1023	43	23.79	winequality-red-8_vs_6	656	638	18	35.44
glass-0-1-4-6_vs_2	205	188	17	11.06	winequality-white-3-9_vs_5	1482	1457	25	58.28
glass-0-1-5_vs_2	172	155	17	9.12	winequality-white-9_vs_4	168	163	5	32.6
glass-0-1-6_vs_2	192	175	17	10.29	yeast-0-3-5-9_vs_7-8	506	456	50	9.12
glass2	214	197	17	11.59	yeast-0-5-6-7-9_vs_4	528	477	51	9.35
haberman	306	225	81	2.78	yeast-1-2-8-9_vs_7	947	917	30	30.57
pima	768	500	268	1.87	yeast-1-4-5-8_vs_7	693	663	30	22.1
poker-8-9_vs_5	2075	2050	25	82	yeast-1_vs_7	459	429	30	14.3

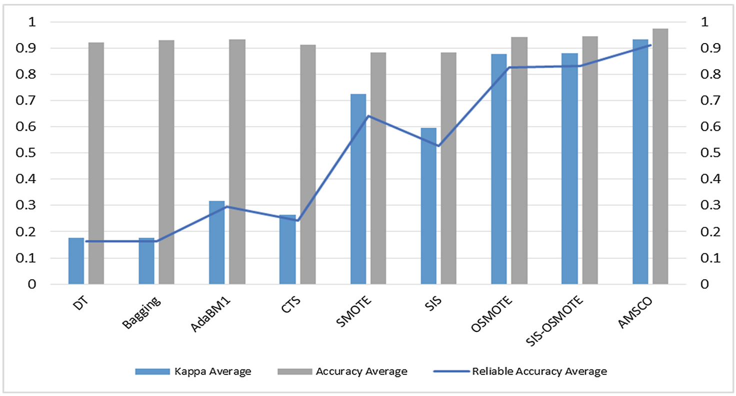

Fig. 7. Overall comparison of the rebalancing methods in terms of Kappa, accuracy and Reliable Accuracy

SMOTE. However, OSMOTE which focuses on minority data is more effective than SIS. Compared to the conventional methods, AMSCO's Kappa values stay above 0.9 in 26 out of 30 cases, achieve perfect performance at 1 in 3 out of 30 cases. AMSCO is relatively more stable too comparing to other methods. The average values of accuracy and Kappa are the highest, and the standard deviations of the two performance measures are the lowest for AMSCO.

5 Conclusions

This paper proposed AMSCO algorithm to tackle class imbalanced dataset in sampling two classes in parallel. AMSCO aims at rebalancing the dataset, improving classification model's credibility and preserving the high accuracy, within reasonable time.

It implements two swarm optimization algorithm to progressively find out the best performance for a specific classifier. One optimization process called OSMOTE is extended from SMOTE. It focuses on inflating the minority data to an appropriate amount. The other is called SIS which selects only the useful instances for filtering the majority data. Experimental results show that, AMSCO can significantly outperform a number of rebalancing methods in different categories. In our future works, AMSCO will be extended to solving imbalanced problem on multi-class classification.

Acknowledgement. The authors are thankful for the financial support from the Research Grant Temporal Data Stream Mining by Using Incrementally Optimized Very Fast Decision Forest (iOVFDF), Grant no. MYRG2015-00128-FST, offered by the University of Macau, FST, and RDAO.

References

1. Sun, A., Ee-Peng, L., Liu, Y.: On strategies for imbalanced text classification using SVM: a comparative study. Decis. Support Syst. **48**(1), 191–201 (2009)
2. Cao, H., Li, X.L., Woon, D.Y.K., Ng, S.K.: Integrated oversampling for imbalanced time series classification. IEEE Trans. Knowl. Data Eng. **25**(12), 2809–2822 (2013)
3. Chan, P.K., Stolfo, S.J.: Toward scalable learning with non-uniform class and cost distributions: a case study in credit card fraud detection. In: KDD, vol. 1998 (1998)
4. Kubat, M., Holte, R.C., Matwin, S.: Machine learning for the detection of oil spills in satellite radar images. Mach. Learn. **30**(2-3), 195–215 (1998)
5. Choe, W., Ersoy, O.K., Bina, M.: Neural network schemes for detecting rare events in human genomic DNA. Bioinformatics **16**(12), 1062–1072 (2000)
6. Mazurowski, M.A., et al.: Training neural network classifiers for medical decision making: the effects of imbalanced datasets on classification performance. Neural Netw. **21**(2), 427–436 (2008)
7. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. **21**(9), 1263–1284 (2009)
8. Tang, Y., Zhang, Y.Q., Chawla, N.V., Krasser, S.: SVMs modeling for highly imbalanced classification. IEEE Trans. Syst. Man Cybern. Part B Cybern. **39**(1), 281–288 (2009)
9. Guo, H., Viktor, H.L.: Learning from imbalanced data sets with boosting and data generation: the DataBoost-IM approach. ACM SIGKDD Explor. Newslett. **6**(1), 30–39 (2004)
10. Li, J., Fong, S., Mohammed, S., et al.: Improving the classification performance of biological imbalanced datasets by swarm optimization algorithms. J. Supercomput. **72**, 3708 (2016). doi:10.1007/s11227-015-1541-6
11. Chawla, N.V.: C4. 5 and imbalanced data sets: investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In: Proceedings of the ICML, vol. 3 (2003)
12. Stone, E.A.: Predictor performance with stratified data and imbalanced classes. Nat. Methods **11**(8), 782–783 (2014)
13. Chen, Y.-W., Lin, C.-J.: Combining SVMs with various feature selection strategies. In: Guyon, I., Nikravesh, M., Gunn, S., Zadeh, L.A. (eds.) Feature Extraction: Foundations and Applications. Studies in Fuzziness and Soft Computing, pp. 315–324. Springer, Heidelberg (2006)

14. Wallace, B.C., et al.: Class imbalance, redux. In: 2011 IEEE 11th International Conference on Data Mining (ICDM). IEEE (2011)
15. Liu, A., Ghosh, J., Martin, C.E.: Generative oversampling for mining imbalanced datasets. In: DMIN (2007)
16. Batuwita, R., Palade, V: Efficient resampling methods for training support vector machines with imbalanced datasets. In: The 2010 International Joint Conference on Neural Networks (IJCNN). IEEE (2010)
17. Drummond, C., Holte, R.C.: C4. 5 class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In: Workshop on learning from imbalanced datasets II, vol. 11 (2003)
18. Kubat, M., Matwin, S: Addressing the curse of imbalanced training sets: one-sided selection. In: ICML, vol. 97 (1997)
19. Chawla, N.V., Bowyer, K.W.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 341–378 (2002)
20. Galar, M., et al.: A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **42**(4), 463–484 (2012)
21. Thai-Nghe, N., Gantner, Z., Schmidt-Thieme, L: Cost-sensitive learning methods for imbalanced data. In: The 2010 International Joint Conference on Neural Networks (IJCNN). IEEE (2010)
22. Zhu, X.: Lazy bagging for classifying imbalanced data. In: IEEE ICDM 2007, pp. 763–768 (2007)
23. Sun, Y., Kamel, M.S., Wang, Y.: Boosting for learning multiple classes with imbalanced class distribution. In: IEEE ICDM 2006, pp. 592–602 (2006)
24. del Río, S., et al.: On the use of MapReduce for imbalanced big data using random forest. Inf. Sci. **285**, 112–137 (2014)
25. Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W.: SMOTEBoost: improving prediction of the minority class in boosting. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 107–119. Springer, Heidelberg (2003). doi:10.1007/978-3-540-39804-2_12
26. Fan, W., et al.: AdaCost: misclassification cost-sensitive boosting. In: ICML, vol. 99 (1999)
27. Sun, Y., et al.: Cost-sensitive boosting for classification of imbalanced data. Pattern Recognit. **40**(12), 3358–3378 (2007)
28. Zadrozny, B., Langford, J., Abe, N.: Cost-sensitive learning by cost-proportionate example weighting. In: IEEE ICDM 2003, pp. 435–442 (2003)
29. Kennedy, J., et al.: Swarm Intelligence. Morgan Kaufmann, San Francisco (2001)
30. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Swarm Intell. **1**(1), 33–57 (2007)
31. Li, J., et al.: Adaptive swarm balancing algorithms for rare-event prediction in imbalanced healthcare data. Comput. Med. Imaging Graph (2016). http://dx.doi.org/10.1016/j.compmedimag.2016.05.001
32. Van den Bergh, F., Engelbrecht, A.P.: A cooperative approach to particle swarm optimization. IEEE Trans. Evol. Comput. **8**(3), 225–239 (2004)
33. Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. Biometrics **33**, 159–174 (1977)
34. Viera, A.J., Garrett, J.M.: Understanding interobserver agreement: the kappa statistic. Fam. Med. **37**(5), 360–363 (2005)
35. Fonseca, C.M., Fleming, P.J.: Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I: a unified formulation. IEEE Trans. Syst. Man Cybern. Part A Syst. Hum. **28**(1), 26–37 (1998)

36. García, S., Herrera, F.: Evolutionary undersampling for classification with imbalanced datasets: proposals and taxonomy. Evol. Comput. **17**(3), 275–306 (2009)
37. Cieslak, D.A., Chawla, N.V.: Learning decision trees for unbalanced data. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008. LNCS (LNAI), vol. 5211, pp. 241–256. Springer, Heidelberg (2008). doi:10.1007/978-3-540-87479-9_34
38. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: ICML, vol. 96 (1996)
39. Alcalá, J., et al.: Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J. Multiple Valued Logic Soft Comput. **17** (255-287), 11 (2010)