A New Under-Sampling Method Using Genetic Algorithm for Imbalanced Data Classification

Jihyun Ha
Department of Industrial Engineering
Sungkyunkwan University
2066 Seobu-ro, Suwon 16419, Republic of Korea
haaforever@skku.edu

Jong-Seok Lee
Department of Industrial Engineering
Sungkyunkwan University
2066 Seobu-ro, Suwon 16419, Republic of Korea
jongseok@skku.edu

ABSTRACT
The class imbalance problem is frequently found in many real-world domains, where many of traditional classifiers often fail to detect minority class objects due to paying less attention to those. In an effort to address this class imbalance problem, a new under-sampling technique GAUS (genetic algorithm based under-sampling) is proposed in this paper. GAUS is designed to overcome several limitations of existing methods such as performance instability and information loss of data distribution. To select informative majority objects, GAUS tries to maximize the performance of a prototype classifier such that the prototypes minimize the loss between distributions of original and undersampled majority objects. We confirmed the effectiveness of the proposed GAUS based on real-world datasets.

Categories and Subject Descriptors
I.5.2 [**Pattern Recognition**]: Design Methodology – *pattern analysis*

General Terms
Algorithms, Measurement, Performance, Design, Reliability, Experimentation

Keywords
Imbalanced classification, under-sampling, genetic algorithm

1. INTRODUCTION
Classification is to construct a classifier based on a given training dataset, which makes a classification decision of an unlabeled object. Classifiers tend to show poor performances in many real-world data mining applications such as medical diagnosis [1], fraud detection [2], video surveillance [3], vehicle diagnostics [4] and so on. The common characteristic of these applications is that each of class contains different numbers of objects. Since traditional classifiers were developed to maximize an overall accuracy, which is independent of the class distribution, they show a predictive bias towards the majority (negative) class by paying less attention to the minority (positive) class [5]. Therefore, classification of imbalanced datasets is one of the challenging problems, which has consistently attracted a significant amount of interest from researchers in various fields of academia and industries [5].

A large number of techniques have been developed to deal with the problem of data imbalance. They can be roughly divided into two categories; the internal approaches and external approaches [6]. Methods belonging to the internal approaches, also referred to as algorithm level approaches, are based on the modification of existing learning algorithms or combine a classifier into meta-learning algorithms such as bagging and boosting [7]. On the other hand, the external approaches, also known as data level approaches or resampling methods, aim at balancing the class distribution before learning a classifier [8]. Techniques in the external approaches are categorized into two main types [9]. The first one is an oversampling technique magnifying the size of minority class, and the other is an undersampling technique which makes the number of major objects smaller. In this work, since the proposed method is included in the undersampling technique, we provide an overview of the methods in this category only.

There is a lack of research on undersampling methods compared with oversampling methods. In addition, existing undersampling methods suffer from the instability of performance. In this paper, we propose GAUS (Genetic Algorithm based Under-Sampling) to deal with the limitations of existing methods. Using GA, instance selection of majority objects is utilized to undersample the original set of majority objects. To avoid the distortion of decision boundary, classification performance of the prototype classifier is employed as the fitness function in GA.

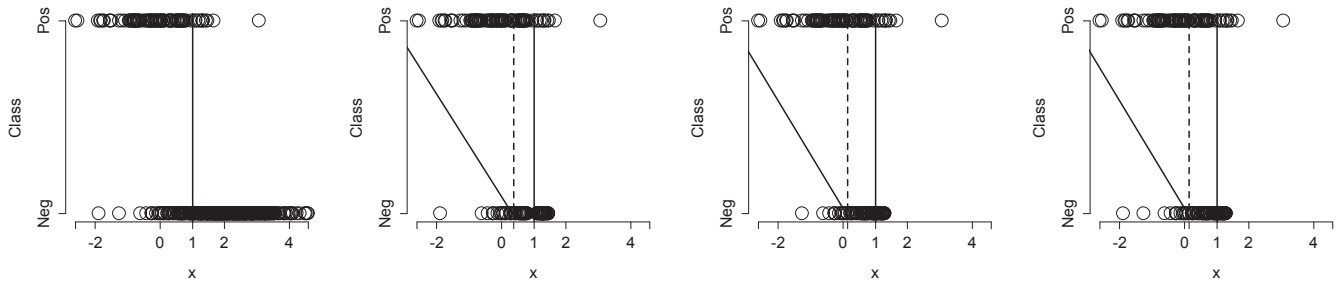
The rest of this paper is organized as follows. In section 2, we briefly review existing undersampling methods, and their major drawbacks are discussed in section 3. Section 4 describes the proposed method in detail. Design of numerical experiments and the results will be introduced in section 5. In section 6, we conclude our research and provide some future works.

2. RELATED WORK
As mentioned earlier, the internal and external approaches are main solutions for dealing with imbalance data classification. In this section, since the proposed method is related to the undersampling technique, we focus on presenting a brief overview about the existing undersampling methods.

Random undersampling (RUS) is by far the simplest method for balancing class distribution. A set of major objects is randomly selected and included into a training dataset. However, RUS has a major drawback that it can eliminate potentially useful objects or contain noisy objects, which causes a problem of the distortion of decision boundary [10].

Mani and Zhang [11] proposed four different undersampling methods, namely, NearMiss-1, NearMiss-2, NearMiss-3 and

IMCOM '16, January 04–06, 2016, Danang, Vietnam.
 ISBN 978-1-4503-4142-4/16/01…$15.00
DOI: http://dx.doi.org/10.1145/2857546.2857643



(a) Original dataset (b) NearMiss-1 (c) NearMiss-2 (d) NearMiss-3

Figure 1. Results of the NearMiss family.

MostDistant, using the k-nearest neighbor (kNN) approach. The main idea of the NearMiss family is that major objects which are close to minor objects are selected to better learn the decision boundary [12]. NearMiss-1 selects the major objects whose individual average distances to the three closest minor objects are the smallest, while MostDistant chooses the ones whose individual average distances to the three closest minor objects are the largest. NearMiss-2 selects the major objects whose average distances to the three farthest minor objects are the smallest, and NearMiss-3 selects a predefined number of the closest major objects for each minor objects.

Cateni et al. [13] proposed the similarity-based undersampling (SBU) approach. To minimize the loss of information and preserve the original distribution, SBU tends to remove major objects in a dense data space. A measure of dissimilarity such as Euclidean distance between each pair of major objects is first calculated. The smaller value of dissimilarity, the more similar the two major objects. After sorting values of dissimilarity for all possible pairs in an ascending order, one of the pairs in the n (input parameter) smallest values of dissimilarity is randomly eliminated from the original dataset.

3. PROBLEMS OF EXISTING METHODS

However, applying existing undersampling techniques to an imbalanced dataset is not a good strategy to improve the ability of a classifier to learn a better decision rule. In this section, we present a brief explanation why existing methods are not eligible to improve the learnability of a classifier in terms of the area under the receiver operating characteristic curve (AUC-ROC) [14].

Lemma 1: Let h be a simple classification rule $h: X \rightarrow Y$ such that if $x > k$ then $h(x) = 0$, or $x \leq k$ then $h(x) = 1$, where $y = 0$ is for the majority class and $y = 1$ represents the minority class. Let D be a given one-dimensional dataset for imbalanced binary classification. Let n_p minority objects follow $N(\mu_p, \sigma)$ and n_n majority objects follow $N(\mu_n, \sigma)$ where $\mu_p < \mu_n$. Then, the classifier h has the following optimal decision boundary which maximizes the value of AUC-ROC.

$$k^* = \frac{(\mu_p + \mu_n)}{2} \tag{1}$$

Proof: When, the decision boundary is $x = k$, the true positive rate (TPR) and false positive rate (FPR) are as follows.

$$\mathrm{TPR}(k) = \int_{-\infty}^{k} f_p(x)dx \tag{2}$$

where f_p is the probability density function of $N(\mu_p, \sigma^2)$.

$$\mathrm{FPR}(k) = \int_{-\infty}^{k} f_n(x)dx \tag{3}$$

where f_n is the probability density function of $N(\mu_n, \sigma^2)$.

Therefore, the value of AUC-ROC at $x = k$ is:

$$\mathrm{AUC}(k) = \frac{1}{2}\{1 + \mathrm{TPR}(k) - \mathrm{FPR}(k)\} \tag{4}$$

$$= \frac{1}{2}\left\{1 + \Phi\left(\frac{k - \mu_p}{\sigma}\right) - \Phi\left(\frac{k - \mu_n}{\sigma}\right)\right\} \tag{5}$$

where Φ is the cumulative density function (CDF) of the standard normal distribution. By setting the derivative of $\mathrm{AUC}(k)$ equal to zero and solving for k, the optimal decision boundary k^* (Eq. 1) is obtained. ■

Proposition 1: For the dataset D, the performance of RUS becomes more unstable as the number of minority objects decreases.

Proof: Since RUS is applied to the dataset D to balance the class distribution, it holds that

$$\bar{X} \sim N\left(\mu_n, \frac{\sigma^2}{n_p}\right) \tag{6}$$

where $\bar{X}$ is a sample mean of n_p undersampled majority objects. Then, the variance of the optimal decision boundary K^* after applying RUS to D is as follows.

$$V(K^*) = V\left(\frac{\mu_p + \bar{X}}{2}\right) = \frac{\sigma^2}{4n_p} \tag{7}$$

Therefore, as the number of minority objects in the dataset D decreases, the variance of the estimated decision boundary through the RUS goes higher. ■

It is obvious that undersampled majority objects by NearMiss family and SBU do not follow the original normal distribution. They therefore tend to build a biased decision boundary. Figure 1 shows several results of those methods. In Figure 1(a), there are 100 minority objects generated from $N(0,1)$, and 500 majority objects follow $N(2,1)$. NearMiss-1, NearMiss-2 and NearMiss-3 are applied to the set of majority objects to select 100 objects, and Figure 1(b), (c) and (d) show their results, respectively. The solid line is the optimal decision boundary, $x = 1$, by Equation (1), and dotted line is the estimated decision boundary by NearMiss

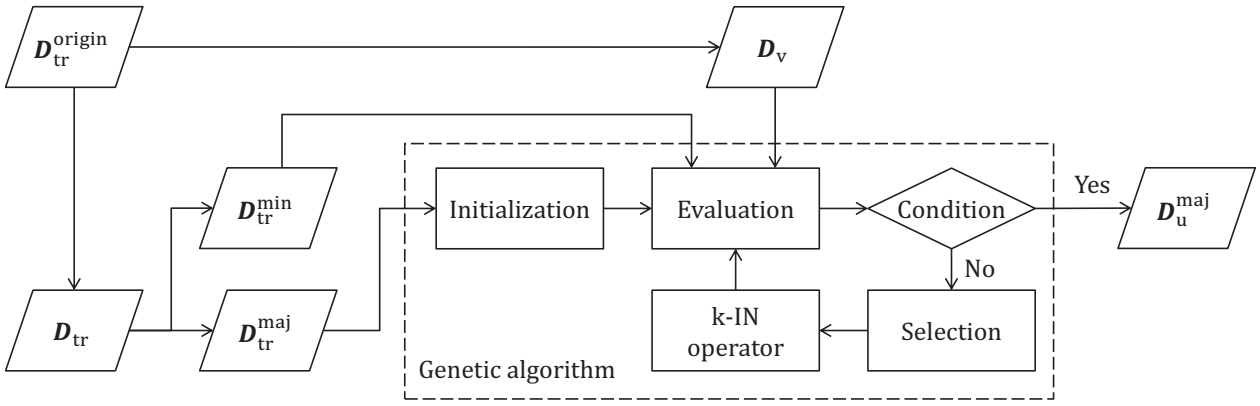


Figure 2. Flowchart of the proposed method (GAUS).

for the balanced dataset. As can be seen from the plots, the bias between estimated and true optimal decision boundary exists due o the tendency to sample majority objects near the decision boundary.

4. PROPOSED METHOD

In this study, to overcome the limitations of the existing methods discussed in the previous section, we propose a new under-sampling method called GAUS (Genetic Algorithm based Under-Sampling). Figure 2 shows the big picture of the proposed method. It is on the basis of majority instance selection maximizing the performance of a prototype classifier over a validation dataset. Genetic algorithm (GA) is used for finding majority prototypes. To minimize the loss of original distribution usually caused by undersampling, the performance of the prototype classifiier is employed as a fitness value of the chromosome.

First, a given training dataset $\boldsymbol{D}_{\mathrm{tr}}^{\mathrm{origin}}$ is split into a training dataset $\boldsymbol{D}_{\mathrm{tr}}$ and a validation dataset $\boldsymbol{D}_{\mathrm{v}}$ using stratified random partitioning. Some parts of $\boldsymbol{D}_{\mathrm{tr}}$ are directly encoded to serve as a solution space in GA, and $\boldsymbol{D}_{\mathrm{v}}$ is used for evaluating the fitness of chromosomes. The size of $\boldsymbol{D}_{\mathrm{v}}$ is determined by a user. In this work, we set the size of $\boldsymbol{D}_{\mathrm{v}}$ at $\beta = 20\%$ of the size of $\boldsymbol{D}_{\mathrm{tr}}^{\mathrm{origin}}$. Since the output of undersampling is a set of majority objects, only majority objects $\boldsymbol{D}_{\mathrm{tr}}^{\mathrm{maj}}$ belonging to $\boldsymbol{D}_{\mathrm{tr}}$ are entered into the GA framework. It means that a single chromosome indicates which objects are sampled from $\boldsymbol{D}_{\mathrm{tr}}^{\mathrm{maj}}$ to adjust the unbalanced distribution of class.

The length of a chromosome (the number of gene) is the same as the number of majority objects to be sampled. To make $\boldsymbol{D}_{\mathrm{tr}}^{\mathrm{origin}}$ totally balanced, the number of minority objects in $\boldsymbol{D}_{\mathrm{tr}}^{\mathrm{origin}}$ is same with the length of the chromosome. For example, let $\boldsymbol{D}_{\mathrm{tr}}^{\mathrm{origin}}$ contains 5 minority and 50 majority objects, and $\beta = 20\%$. If any user wants to make a totally balanced class distribution of $\boldsymbol{D}_{\mathrm{tr}}^{\mathrm{origin}}$, the length of a chromosome will be 5 which is the number of minority objects in $\boldsymbol{D}_{\mathrm{tr}}^{\mathrm{origin}}$. The index of objects in $\boldsymbol{D}_{\mathrm{tr}}^{\mathrm{maj}}$ is encoded into a chromosome. The values of each gene indicate the sampled objects from $\boldsymbol{D}_{\mathrm{tr}}^{\mathrm{maj}}$. For example, $\boldsymbol{D}_{\mathrm{tr}}^{\mathrm{maj}}$ consists of majority objects $\{\boldsymbol{x}_1^{\mathrm{n}}, \boldsymbol{x}_2^{\mathrm{n}} \cdots, \boldsymbol{x}_{39}^{\mathrm{n}}, \boldsymbol{x}_{40}^{\mathrm{n}}\}$. A chromosome (16, 24, 37, 7, 34) means that object $\boldsymbol{x}_7^{\mathrm{n}}$, $\boldsymbol{x}_{16}^{\mathrm{n}}$, $\boldsymbol{x}_{24}^{\mathrm{n}}$, $\boldsymbol{x}_{34}^{\mathrm{n}}$ and $\boldsymbol{x}_{37}^{\mathrm{n}}$ are selected from $\boldsymbol{D}_{\mathrm{tr}}^{\mathrm{maj}}$ to balance $\boldsymbol{D}_{\mathrm{tr}}^{\mathrm{origin}}$.

As previously mentioned, each chromosome indicating a candidate set of undersampled majority objects is evaluated in terms of classification performance (AUC-ROC) over $\boldsymbol{D}_{\mathrm{v}}$. Let $\boldsymbol{D}_{\mathrm{u}}^{\mathrm{maj}}$ denote a set of sampled majority objects by decoding a chromosome. Using $\boldsymbol{D}_{\mathrm{u}}^{\mathrm{maj}} \cup \boldsymbol{D}_{\mathrm{tr}}^{\mathrm{min}}$ as prototypes, each object in $\boldsymbol{D}_{\mathrm{v}}$ is classified by taking the class of its first nearest prototype (1NN). Since $\boldsymbol{D}_{\mathrm{v}}$ is also an imbalanced dataset generated from $\boldsymbol{D}_{\mathrm{tr}}^{\mathrm{origin}}$, the fitness of the chromosome is measure by AUC-ROC.

Algorithm 1: FitnessEvaluation

Input:
[1] c_i : a chromosome i
[2] $\boldsymbol{D}_{\mathrm{v}}$: validation dataset
[3] $\boldsymbol{D}_{\mathrm{tr}}^{\mathrm{min}}$: a set of minority objects in $\boldsymbol{D}_{\mathrm{tr}}$

Procedure:
[1] $\boldsymbol{D}_{\mathrm{u}}^{\mathrm{maj}} \leftarrow$ decode c_i
[2] $\boldsymbol{D}_{\mathrm{prt}} \leftarrow \boldsymbol{D}_{\mathrm{tr}}^{\mathrm{min}} \cup \boldsymbol{D}_{\mathrm{u}}^{\mathrm{maj}}$
[3] **For each** $\boldsymbol{x}_j$ in $\boldsymbol{D}_{\mathrm{v}}$ **do**
[4] $\quad \boldsymbol{p}_j \leftarrow$ find an object in $\boldsymbol{D}_{\mathrm{prt}}$ closest in distance to $\boldsymbol{x}_j$
[5] $\quad$ **If** $\boldsymbol{p}_j$ is an minority object **then**
[6] $\quad\quad \hat{y}_j \leftarrow 1$ # predicted class
[7] $\quad$ **Else**
[8] $\quad\quad \hat{y}_j \leftarrow 0$
[9] $\quad$ **End if**
[10] **End for**
[11] TPR $\leftarrow \left(\sum_{\boldsymbol{x}_j \in \boldsymbol{D}_{\mathrm{v}}} I(y_j = 1 \wedge \hat{y}_j = 1)\right) / \left(\sum_{\boldsymbol{x}_j \in \boldsymbol{D}_{\mathrm{v}}} I(y_j = 1)\right)$
[12] FPR $\leftarrow \left(\sum_{\boldsymbol{x}_j \in \boldsymbol{D}_{\mathrm{v}}} I(y_j = 0 \wedge \hat{y}_j = 1)\right) / \left(\sum_{\boldsymbol{x}_j \in \boldsymbol{D}_{\mathrm{v}}} I(y_j = 0)\right)$
[13] $f_i \leftarrow (1 + \mathrm{TPR} - \mathrm{FPR})/2$

Output:
[1] f_i : fitness value of chromosome i

Algorithm 1 shows how the fitness value of a chromosome is computed. Lines 1 to 10 explain how to predict the class of objects in $\boldsymbol{D}_{\mathrm{v}}$ using a chromosome and the first nearest neighbor classification. In lines 11 to 13, TPR (true positive rate), FPR (false positive rate) and AUC-ROC are computed. Since encoding the chromosome is based on the indices of objects, general crossover and mutation operator are not an efficient option. In this work, k-influential neighborhood genetic operator is proposed, which is based on the concept of nearest and reverse nearest

neighbor. Using the following definitions and the example, we explain how to perform crossover and mutation operators.

Definition 1 (k reverse nearest neighbors of $\boldsymbol{x}$): The k reverse nearest neighbors of $\boldsymbol{x}$, denoted as $RN_k(\boldsymbol{x})$, is a set of objects $\boldsymbol{q}$ that have $\boldsymbol{x}$ as one of their k nearest neighbors, namely,

$$RN_k(\boldsymbol{x}) = \{\boldsymbol{q} | \boldsymbol{q} \in \boldsymbol{D}, \boldsymbol{x} \in N_k(\boldsymbol{q})\} \tag{8}$$

where $N_k(\boldsymbol{q})$ is a set of k nearest neighbors of $\boldsymbol{q}$. For any object $\boldsymbol{x}$, $RN_k(\boldsymbol{x})$ can be an empty set, or have one or more elements, while $N_k(\boldsymbol{x})$ always contains at least k objects.

Definition 2 (k-influential neighborhood of $\boldsymbol{x}$): The k-influential neighborhood of $\boldsymbol{x}$, denoted as $IN_k(\boldsymbol{x})$, is a union of $N_k(\boldsymbol{x})$, $RN_k(\boldsymbol{x})$ and $\{\boldsymbol{x}\}$ which can be defined as below.

$$IN_k(\boldsymbol{x}) = N_k(\boldsymbol{x}) \cup RN_k(\boldsymbol{x}) \cup \{\boldsymbol{x}\} \tag{9}$$

Example 1 (k-influential neighborhood of $\boldsymbol{x}$): Figure 3 shows a simple graphical description of how to obtain the influential neighborhood when $k = 1$. Here, $N_1(\boldsymbol{x}_1) = \{\boldsymbol{x}_3\}$, $N_1(\boldsymbol{x}_2) = \{\boldsymbol{x}_3\}$, $N_1(\boldsymbol{x}_3) = \{\boldsymbol{x}_4\}$, $N_1(\boldsymbol{x}_4) = \{\boldsymbol{x}_5\}$ and $N_1(\boldsymbol{x}_5) = \{\boldsymbol{x}_4\}$. During the search of the first nearest neighbor for each object, the first reverse nearest neighbor can be simultaneously built. $RN_1(\boldsymbol{x}_1) = \emptyset$, $RN_1(\boldsymbol{x}_2) = \emptyset$, $RN_1(\boldsymbol{x}_3) = \{\boldsymbol{x}_1, \boldsymbol{x}_2\}$, $RN_1(\boldsymbol{x}_4) = \{\boldsymbol{x}_3, \boldsymbol{x}_5\}$ and $RN_1(\boldsymbol{x}_5) = \{\boldsymbol{x}_4\}$. Finally, by combining $N_1(\boldsymbol{x})$, $RN_1(\boldsymbol{x})$ and $\{\boldsymbol{x}\}$ together,
$IN_1(\boldsymbol{x}_1) = \{\boldsymbol{x}_1, \boldsymbol{x}_3\}$, $IN_1(\boldsymbol{x}_2) = \{\boldsymbol{x}_2, \boldsymbol{x}_3\}$,
$IN_1(\boldsymbol{x}_3) = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4\}$, $IN_1(\boldsymbol{x}_4) = \{\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\}$
and $IN_1(\boldsymbol{x}_5) = \{\boldsymbol{x}_4, \boldsymbol{x}_5\}$ are obtained.

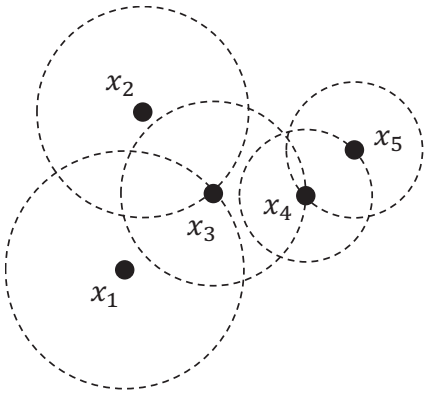


Figure 3. Influential neighborhood.

In general, an unbalanced dataset contains a huge number of majority objects. Therefore, the encoding way and genetic operators that are usually used construct an excessively large solution space. As a result, the convergence speed of the fitness value becomes very slow, and such a time-consuming task is inefficient for data preprocessing. To find the best subset of majority prototypes more effectively, the k-influential neighborhood is used for searching better descendants in the process of genetic evolution. The procedure for crossover using k-influential neighborhood is as follows. Let a_i be the i-th gene of its mother chromosome and b_i be the i-th gene the father chromosome.

(1) Find the union of $IN_k(a_i)$ and $IN_k(b_i)$.
(2) Draw a random sample from the union set, and put it into i-th gene of child chromosome.
(3) Repeat the above procedure for all of genes.

The mutation process is as follows. Let g_i be the i-th gene of a chromosome.

(1) Find $IN_k(g_i)$
(2) Draw a random sample from $D - IN_k(g_i)$, and put it into i-th gene.
(3) Repeat the above procedure for all of genes.

Offspring chromosomes are generated by sampling objects from the set of influential neighborhood of their parents. Genetic diversity is guaranteed by sampling the objects which do not belong to k-influential neighborhood. The whole procedure of the proposed GAUS method is described below.

Algorithm 2: GAUS

Input:
[1] $\boldsymbol{D}_{\text{tr}}$: training dataset
[2] $\boldsymbol{D}_{\text{v}}$: validation dataset
[3] k : size of influential neighborhood
[4] prC : crossover probability
[5] prM : mutation probability
[6] $maxG$: maximum number of generations

Procedure:
[1] $\boldsymbol{D}_{\text{tr}}^{\text{min}} \leftarrow$ a set of minority objects in $\boldsymbol{D}_{\text{tr}}$
[2] $\boldsymbol{D}_{\text{tr}}^{\text{maj}} \leftarrow$ a set of majority objects in $\boldsymbol{D}_{\text{tr}}$
[3] $g \leftarrow 1$
[4] $\boldsymbol{P}[g] \leftarrow \text{Initialization}\left(\boldsymbol{D}_{\text{tr}}^{\text{maj}}\right)$
[5] $\boldsymbol{F} \leftarrow \text{FitnessEvaluation}\left(popul[g], \boldsymbol{D}_{\text{v}}, \boldsymbol{D}_{\text{tr}}^{\text{min}}\right)$
[6] $\boldsymbol{D}_{\text{u}}^{\text{maj}} \leftarrow$ update the current best chromosome
[7] **While** $g < maxG$ **do**
[8] $\quad \boldsymbol{P}[g+1] \leftarrow \emptyset$
[9] $\quad$ **For each** chromosome i in $\boldsymbol{P}[g+1]$ **do**
[10] $\quad\quad r \leftarrow$ random number from [0,1]
[11] $\quad\quad$ **If** $r < prC$ **then**
[12] $\quad\quad\quad par \leftarrow \text{Selection}(\boldsymbol{P}[g], \boldsymbol{F})$
[13] $\quad\quad\quad \boldsymbol{P}[g+1][i] \leftarrow \text{Crossover}\left(par, k, \boldsymbol{D}_{\text{tr}}^{\text{maj}}\right)$
[14] $\quad\quad$ **Else**
[15] $\quad\quad\quad \boldsymbol{P}[g+1][i] \leftarrow \text{Initialization}\left(\boldsymbol{D}_{\text{tr}}^{\text{maj}}\right)$
[16] $\quad\quad$ **End if**
[17] $\quad\quad r \leftarrow$ random number from [0,1]
[18] $\quad\quad$ **If** $r < prM$ **then**
[19] $\quad\quad\quad \boldsymbol{P}[g+1][i] \leftarrow \text{Mutation}\left(\boldsymbol{P}[g+1][i], k, \boldsymbol{D}_{\text{tr}}^{\text{maj}}\right)$
[20] $\quad\quad$ **End if**
[21] $\quad$ **End for**
[22] $\quad \boldsymbol{D}_{\text{u}}^{\text{maj}} \leftarrow$ update the current best chromosome
[23] $\quad g \leftarrow g + 1$
[24] **End while**

Output:
[1] $\boldsymbol{D}_{\text{u}}^{\text{maj}}$: undersampled majority dataset

The input parameter of GAUS is prC for a crossover probability, prM for a mutation probability, k for constructing k-influential neighborhood and $maxG$ which is the maximum number of generations. In line 4, random initialization of population is performed. Line 5 indicates the evaluation of population. In lines 10 to 20, crossover and mutation operators using k-influential neighborhood are conducted to generate offspring chromosomes.

Table 2. Performance comparison for 7 sampling methods, 6 classifiers and 4 measures.

1NN								3NN						
Method	GAUS	MD	NM1	NM2	NM3	RUS	SBU	GAUS	MD	NM1	NM2	NM3	RUS	SBU
AUC PR	**36 (54%)**	0 (0%)	7 (10%)	5 (7%)	0 (0%)	12 (18%)	7 (10%)	**29 (43%)**	0 (0%)	8 (12%)	4 (6%)	0 (0%)	18 (27%)	8 (12%)
AUC ROC	**35 (52%)**	0 (0%)	10 (15%)	2 (3%)	0 (0%)	19 (28%)	1 (1%)	**29 (43%)**	0 (0%)	10 (15%)	1 (1%)	0 (0%)	22 (33%)	5 (7%)
F1	**40 (60%)**	0 (0%)	7 (10%)	5 (7%)	0 (0%)	8 (12%)	7 (10%)	**29 (43%)**	0 (0%)	8 (12%)	4 (6%)	4 (6%)	14 (21%)	8 (12%)
G-mean	**37 (55%)**	0 (0%)	11 (16%)	1 (1%)	0 (0%)	17 (25%)	1 (1%)	**31 (46%)**	0 (0%)	9 (13%)	1 (1%)	0 (0%)	22 (33%)	4 (6%)
5NN								LOGR						
Method	GAUS	MD	NM1	NM2	NM3	RUS	SBU	GAUS	MD	NM1	NM2	NM3	RUS	SBU
AUC PR	**29 (43%)**	0 (0%)	8 (12%)	4 (6%)	0 (0%)	19 (28%)	8 (12%)	**17 (25%)**	12 (18%)	1 (1%)	5 (7%)	10 (15%)	7 (10%)	15 (22%)
AUC ROC	**31 (46%)**	0 (0%)	6 (9%)	1 (1%)	0 (0%)	25 (37%)	4 (6%)	**22 (33%)**	10 (15%)	1 (1%)	6 (9%)	0 (0%)	19 (28%)	9 (13%)
F1	**32 (48%)**	0 (0%)	6 (9%)	4 (6%)	4 (6%)	13 (19%)	8 (12%)	**21 (31%)**	5 (7%)	1 (1%)	7 (10%)	9 (13%)	8 (12%)	16 (24%)
G-mean	**32 (48%)**	1 (1%)	5 (7%)	1 (1%)	0 (0%)	24 (36%)	4 (6%)	**23 (34%)**	11 (16%)	1 (1%)	7 (10%)	0 (0%)	17 (25%)	8 (12%)
CART								NB						
Method	GAUS	MD	NM1	NM2	NM3	RUS	SBU	GAUS	MD	NM1	NM2	NM3	RUS	SBU
AUC PR	14 (21%)	1 (1%)	11 (16%)	7 (10%)	2 (3%)	**18 (27%)**	14 (21%)	**19 (28%)**	8 (12%)	1 (1%)	8 (12%)	8 (12%)	6 (9%)	17 (25%)
AUC ROC	16 (24%)	1 (1%)	7 (10%)	6 (9%)	1 (1%)	**26 (39%)**	10 (15%)	**22 (33%)**	9 (13%)	1 (1%)	7 (10%)	8 (12%)	9 (13%)	11 (16%)
F1	15 (22%)	0 (0%)	10 (15%)	6 (9%)	6 (9%)	**19 (28%)**	11 (16%)	**23 (34%)**	6 (9%)	1 (1%)	8 (12%)	8 (12%)	11 (16%)	10 (15%)
G-mean	17 (25%)	1 (1%)	7 (10%)	6 (9%)	1 (1%)	**26 (39%)**	9 (13%)	**18 (27%)**	7 (10%)	4 (6%)	7 (10%)	8 (12%)	12 (18%)	11 (16%)

Maximum generation number is the termination condition of the genetic algorithm.

5. EXPERIMENTAL EVALUATION

A numerical experiment was conducted to confirm the effectiveness of the propose GAUS using 67 datasets from KEEL dataset repository [15]. Table 1 shows the parameter setting for the proposed method.

Table 1. Parameter setting for GAUS.

Parameter	value
k	5
prC	0.5
prM	0.05
$maxG$	50
Population size	100
Selection	Roulette wheel selection

We used 1NN, 3NN, 5NN, logistic regression, decision tree (CART) and naïve Bayes to confirm the robustness of the proposed method. Their performance is measured by AUC-ROC, AUC-PR (area under the precision-recall curve), G-mean and F1 score that are the most fundamental metrics to evaluate classification performance in imbalanced data. All experiments were conducted using 10 replications with 5-fold cross validation, and the average of each metrics is computed to compare the performances. Table 2 shows the results of our experiments. Among 67 datasets, we counted the number of winning for each sampling methods. Except for the case of CART, the proposed method outperformed the existing undersampling methods in terms of all evaluation metrics.

6. CONCLUSION

The problem of the existing undersampling methods is that they make an unreliable decision boundary having high variance or a bias. In this paper, we present a new undersampling method that we named GAUS (genetic algorithm based undersamplnig). We comprehensively studied the effectiveness of the proposed GAUS method based on a quite large set of real-world datasets using several well-known classification techniques. Except for the case of using CART, the proposed method outperformed the existing undersampling methods. For our future research, we plan to analyze why GAUS is not suited to the CART algorithm, and improve the robustness of the proposed method so that it works for any classifier. Additionally, using the k-influential neighborhood genetic operator proposed in this study, we will further develop a better prototype classifier for imbalanced data classifition.

7. ACKNOWLEDGMENT

This research was supported by the MSIP, Korea, under the G-ITRC support program (IITP-2015-R6812-15-0001) supervised by the IITP.

8. REFERENCES

[1] Tomczak, J. M., and Zięba, M. 2015. Probabilistic combination of classification rules and its application to medical diagnosis. *Machine Learning*, 101, 1-3, 105-135.

[2] Sahin, Y., Bulkan, S., and Duman, E. 2013. A cost-sensitive decision tree approach for fraud detection. *Expert Systems with Applications*, 40, 15, 5916-5923.

[3] Wang, J., Fu, W., Lu, H., and Ma, S. 2014. Bilayer Sparse Topic Model for Scene Analysis in Imbalanced Surveillance Videos. *Image Processing, IEEE Transactions on*, 23, 12, 5198-5208.

[4] Murphey, Y. L., Chen, Z. H., and Feldkamp, L. A. 2008. An incremental neural learning framework and its application to vehicle diagnostics. *Applied Intelligence*, 28, 1, 29-49.

[5] He, H., and Garcia, E. 2009. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21, 9, 1263-1284.

[6] Garcı, S., Triguero, I., Carmona, C. J., and Herrera, F. 2012. Evolutionary-based selection of generalized instances for imbalanced classification. *Knowledge-Based Systems*, 25, 1, 3-12.

[7] Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. 2012. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42, 4, 463-484.

[8] Lee, J. S., and Zhu, D. 2011. When Costs Are Unequal and Unknown: A Subtree Grafting Approach for Unbalanced Data Classification*. *Decision Sciences*, 42, 4, 803-829.

[9] Bunkhumpornpat, C., Sinapiromsaran, K., and Lursinsap, C. 2012. DBSMOTE: density-based synthetic minority over-sampling technique. *Applied Intelligence*, 36, 3, 664-684.

[10] Batista, G. E., Prati, R. C., and Monard, M. C. 2004. A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter*, 6, 1, 20-29.

[11] Mani, I., and Zhang, I. 2003. kNN approach to unbalanced data distributions: a case study involving information extraction. *In Proceedings of the 20th International Conference on Machine Learning Workshop on Learning from imbalanced Data Sets* (Washington, USA., August 21-24, 2003)

[12] Prati, R. C., Batista, G. E., and Monard, M. C. 2009. Data mining with imbalanced class distributions: concepts and methods. *In Proceedings of the 4th Indian International Conference on Artificial Intelligence,* (Tumkur, Karnataka, India, December 16-18, 2009). 359-376.

[13] Cateni, S., Colla, V., and Vannucci, M. 2014. A method for resampling imbalanced datasets in binary classification tasks for real-world problems. *Neurocomputing*, 135, 32-41.

[14] Bradley, A. P. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30, 7, 1145-1159.

[15] Alcalá, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., and Herrera, F. 2010. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17. 2-3, 255-287.