Information Sciences 288 (2014) 347–373

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A new approach for imbalanced data classification based on data gravitation

Lizhi Peng [a,b], Hongli Zhang [a,*], Bo Yang [b], Yuehui Chen [b]

[a] School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150002, PR China
[b] Shandong Provincial Key Laboratory for Network Based Intelligent Computing, University of Jinan, Jinan 250022, PR China

ARTICLE INFO

Article history:
Received 6 March 2013
Received in revised form 15 April 2014
Accepted 27 April 2014
Available online 5 August 2014

Keywords:
Data gravitation
Classification
Imbalanced data
Machine learning

ABSTRACT

Imbalanced classification is an important machine learning research topic that troubles most general classification models because of the imbalanced class distribution. A newly developed physical-inspired classification method, i.e., the data gravitation-based classification (DGC) model, performs well in many general classification problems. However, like other general classifiers, the performance of DGC suffers in imbalanced tasks. In this study, we develop a specific DGC model namely Imbalanced DGC (IDGC) model for imbalanced problems. The amplified gravitation coefficient (AGC) is introduced for gravitation computing. AGC is a type of coefficient that contains class imbalance information, which can strengthen and weaken the gravitational field of the minority and majority classes. We also design a fitness evaluation function in the weight optimization procedure of the data distribution to ensure that the model parameters adapt to the imbalanced class distributions. A total of 44 binary class data sets and 15 multiclass imbalanced data sets are used to test the performance of the proposed method. Experimental results show that the adapted DGC model is effective for imbalanced problems.

1. Introduction

Imbalanced classification problems have attracted considerable research interests in recent years. Unlike standard classification problems, an imbalanced task involves a data set that has an "imbalanced" class distribution, i.e., the number of instances in one class (majority class) is outnumbered by the number of instances in another class (minority class). This phenomenon is mainly attributed to the limited instances of the minority class. For example, in Internet traffic classification problems, Web browsing traffic is a dominant type of traffic that occurs in the Internet at each moment [70,50]. However, capturing enough malicious traffic samples, such as attacks and virus traffic, for training is difficult. Many real-world classification tasks, such as medical diagnosis [44], fraud detection [14], finance risk management [7], network intrusion detection [9], stream classification [26], and bioinformatics [67], have similar diagnosis characteristics. In these imbalanced tasks, the minority class is usually more important than the majority class [18,68]. Thus, to maximize the recognition rate of the minority class on the premise of considering a good tradeoff for both of the minority and majority classes is the main goal of an imbalanced task.

* Corresponding author.
E-mail addresses: plz@ujn.edu.cn (L. Peng), zhanghongli@hit.edu.cn (H. Zhang), yangbo@ujn.edu.cn (B. Yang), yhchen@ujn.edu.cn (Y. Chen).

http://dx.doi.org/10.1016/j.ins.2014.04.046
0020-0255/

Most standard classification models are not suitable for imbalanced tasks because such models seek high classification accuracies across the entire data set. Owing to class imbalance, a standard classifier usually classifies a minority class instance incorrectly as the model is over-trained by the majority class instances. Therefore, although a standard classifier can achieve high accuracy in an imbalanced task, the actual performance is poor because its identifying rate of the minority class is low. However, a class imbalance is not the only factor that influences classification performance [34]. Other data distribution characteristics, such as small disjuncts [63], class overlaps [25], noise [53], and borderline samples [45], also hinder classification performance.

The data gravitation-based classification (DGC) [49] model is a new classification model that is based on Newton's law of universal gravitation. The DGC model refers to a data instance in the data space as a data "particle" and considers the type of "gravitation" between any two data particles in the computation. This gravitation is directly proportional to the product of the "masses" of two data particles and is inversely proportion to the square of the distance between the data particles. By comparing the gravitation from different data classes in the training set, DGC can effectively classify a testing data instance in a simple manner. Simić et al. [56] combined DGC with case-based reasoning to create a hybrid intelligent tool for financial forecasting. Similar classification models inspired by gravitation, such as GBC [48] and CGM [64], have also been presented in recent years.

However, DGC suffers from imbalanced data sets [49], which also affect other general classifiers. In a highly imbalanced training set, the gravitational fields of the minority and majority classes are usually weak and extremely strong, respectively. The strong gravitational field of the majority class attracts most of the minority class samples to the majority class, thus resulting in a low TPR. Cano et al. have done an excellent work to adapt DGC model for imbalanced tasks [8], and the improved model is called DGC+. They construct a class-independent attribute-class weights matrix instead of the weight vector in the basic DGC model. The attribute-class weights matrix is able to carry the accurate distribution information of different classes. By using the attribute-class weights matrix, DGC+ modify the computation method of gravitations. The modified computation method of gravitation is able to effectively stress the minority class, and weaken the majority class at the same time. And they use the covariance matrix adaption evolution strategy (CMA-ES) to search the optimum class-independent attribute-class weights matrix. Empirical studies have shown that DGC+ is effective for standard, noisy, and imbalanced data [8]. However, DGC+ is time-consuming because its model parameters in the attribute-class weights matrix are far more than the parameters of the basic DGC model. And to get higher classification accuracies, DGC+ defines each instance as a single data particle, which also increases the computational complexity.

The objective of this study is to design an imbalanced classification model based on the basic DGC model to improve its behavior for imbalanced problems, especially for high imbalanced tasks. Additionally, the study is to obtain a robust approach with respect to the state-of-the-art on this topic. We introduce a new factor namely amplified gravitation coefficient (AGC) to carry the imbalanced class distribution information. AGC is used to modify the gravitation computation method. The modified gravitation computation method strengthens and weakens the gravitational field of the minority and majority classes, respectively. Unlike DGC+, we do not increase the number of model parameters, i.e. we use the feature weight vector in the basic DGC model. We also apply AGC for the weight optimization procedure to ensure that the model parameters adapt to the imbalanced class distributions.

This paper is organized as follows. Section 2 introduces the imbalanced classification problem. Section 3 discusses the basic DGC model including its theoretical basis, classification principles, feature weighting method, and data particle-creating method. Section 4 presents our proposals. Section 5 depicts our experimental setup. Sections 6 and 7 present the results and analysis of the binary class data sets and multiclass data sets, respectively. And then we summarize some lessons learned in the study in Section 8. Finally, Section 9 concludes and outlines the recommendations for the future research.

2. Imbalanced classification problem

2.1. Imbalanced data sets

Many real-world classification tasks face a challenging problem: the instances of one class outnumbers the instances of another class. These classification data sets are called imbalanced data sets, and such classification tasks are called imbalanced classification problems [30,57]. The dominant class in an imbalanced data set is called the majority class, and the subordinate class is called the minority class. In a typical binary classification task, the majority and minority classes are considered negative and positive, respectively. The imbalance ratio (IR) is the basic measure in evaluating the degree of imbalance of an imbalanced data set. It is defined as the difference between the number of instances of the majority and minority classes [46,31].

$$IR = \frac{n_{maj}}{n_{min}} \tag{1}$$

Imbalanced data sets pose a significant challenge to traditional classification techniques. Minority class instances are easily "forgotten" by general classifiers because the classifiers are over-trained by numerous majority class instances. Thus, general classifiers usually misclassify minority class instances. The most important target for an imbalanced classification task is the accurate identification of the minority class instances. Therefore, traditional classification techniques tend to be ineffective when confronting imbalanced tasks.

2.2. Solutions to imbalanced classification

In [42], López et al. make a good review on the technical trends of imbalanced classification researches. A wide range of techniques has been proposed to manage imbalanced classification tasks. These techniques fall into three categories:

- Data-level Approaches
  Imbalanced data set resampling is an effective technique. This solution usually involves two strategies: over-sampling and under-sampling. Over-sampling methods diminish class imbalance by creating new minority class samples [11]. Many studies have devoted to explore effective strategies of over-sampling. The most famous over-sampling method is the synthetic minority over-sampling technique (SMOTE) [11]. SMOTE generates synthetic examples along line segments that are joined in the nearest neighbors of the k minority class. Neighbors from the k nearest neighbors are randomly chosen, and the number of synthetic samples depends on the required over-sampling amount. SMOTE can achieve a good balance between the majority and minority classes and is widely used with other algorithms in imbalanced tasks as a data-level approach [4,61].
  In contrast to over-sampling methods, under-sampling methods rebalance class distributions by reducing the number of majority class samples [10,40]. Chan and Stolfo [10] presented an under-sampling approach by splitting the majority class set into several non-overlapping subsets, with each subset having approximately the same number of examples as the minority class. Each of these subsets, along with the minority class set, is used to train a single classifier. Liu et al. improved this method by using the supervised technique [40]. They removed a number of correctly classified samples in the training process to improve the efficiency of the model building process. Yu et al. [67] presented a novel under-sampling method based on ant colony optimization and used this method to classify imbalanced DNA microarray data.
- Cost-Sensitive Learning
  Cost-sensitive learning is also effective in imbalanced classification tasks [43]. Cost-sensitive learning considers the varying costs of different misclassification types [18] and the cost matrix during model building. A cost-sensitive learning technique generates a model that has the lowest cost. The cost matrix encodes the penalty of classifying samples from one class as another class, e.g., misclassifying the minority class as the majority class. Unlike general classification procedures, cost-sensitive learning seeks the minimum total cost instead of the minimum error rate, thus guiding the classification procedure to highlight the minority class. Cost-sensitive learning methods are categorized into three types. The first type involves data space weighting by modifying the training data set distribution according to misclassification costs. This type is similar to resampling techniques to some extent. The modified distribution is biased toward minority classes [69]. Thus, classifiers can identify instances in minority classes easily. The second type involves building a cost-sensitive classifier by changing the learning process of a classifier [16]. This type uses cost functions to control the different learning steps of a specified classifier. Classic classification models, such as C4.5 decision trees [58], artificial neural networks [71], and support vector machines (SVMs) [59], have been modified by this type. The third type is based on Bayes risk theory and assigns each instance to the class with the lowest risk [68].
- Ensemble Learning
  This dependent technique is a type of hybrid model that embeds data-level approaches into general classifiers for better classification and generalization. Recent studies have focused on ensemble learning techniques. Chawla et al. [12] combined the SMOTE technique with Boost learning to develop the SMOTEBoost method. SMOTEBoost enhances classifiers by rebalancing the distribution of instances of different classes by using SMOTE. Liu et al. [40] presented BalanceCascade and EasyEnsemble, which combine the under-sampling method with AdaBoost classifiers. Seiffert et al. [54] presented RUSBoost, which is simpler and faster than SMOTEBoost. Bagging also has been adapted for ensemble learning. Wang and Yao [61] presented SMOTEBagging by introducing SMOTE into Bagging. Hido et al. [33] used the under-sampling method with Bagging. Khoshgoftaar et al. [37] used noisy and imbalanced data to compare Boosting and Bagging methods. Bagging methods generally outperform Boost methods. They also found that ensemble learning is more time-consuming than the data-level approach and cost-sensitive learning [37]. Galar et al. make a good review on ensemble learning which is helpful to learn more about this technique [22].

3. Basic DGC model

DGC [49] is a classification model that is based on Newton's law of universal gravitation. DGC classifies testing instances by comparing the gravitation of different classes. If a certain class exhibits the largest gravitation, the test instance should be classified into this class. In this section, we introduce the basic DGC model. Sections 3.1 and 3.2 present the concepts, lemmas, and laws that constitute the theoretical basis of the DGC model. Section 3.3 describes the details of the classification principle. Section 3.4 illustrates the data particle creation method. Section 3.5 introduces the feature weighting method of the DGC model. Finally, Section 3.6 compares the DGC model, k-nearest neighbor (KNN) algorithm, and K-means clustering algorithm.

3.1. Concepts

Definition 1 (*Data particle*). Data particle is a kind of data unit that has "Data Mass". Data particle is made up of a group of data elements in data space. These data elements have a certain relationship. Usually this relationship refers to the geometrical neighborhood of these data elements. That is to say the distances between any two data elements in a data particle must be shorter than a predefined value. Data particle has two basic properties:

Definition 2 (*Data mass*). The mass of a data particle is the number of data elements in the data particle.

Definition 3 (*Data centroid*). Suppose $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m (\mathbf{x}_i = \langle x_{i1}, x_{i2}, \ldots, x_{in} \rangle, i = 1, 2, \ldots, m)$ are a group of data elements in a n-dimensional data space S. P is a data particle built up by $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$. Therefore, the data centroid of $P, \mathbf{x}_i = \langle x_{01}, x_{02}, \ldots, x_{0n} \rangle$ is the geometrical center of $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$. It can be described as follows:

$$x_{0j} = \frac{\sum_{i=1}^{m} x_{ij}}{m}, i = 0, 1, \ldots, m, \quad j = 0, 1, \ldots, n \tag{2}$$

Since data particle has data mass and data centroid, so data particle can be described by a pair expression $\langle m, \mathbf{x} \rangle$. Where m is the data mass of the data particle and $\mathbf{x}$ is the data centroid. After the class information (feature y) has been taken into count, data particle could be described as a triple expression $\langle m, \mathbf{x}, y \rangle$.

Definition 4 (*Atomic data particle*). An atomic data particle is a data particle containing only one data element. The data mass of an atomic data particle is 1.

Definition 5 (*Data gravitation*). Data gravitation is defined as the similarity between data particles. Data gravitation is a kind of scalar. This is an important factor different from the physical force. For the same data particle, gravitation between data particles in different data classes can be compared.

On the other hand, data gravitation between data particles in the same class obey the superposition principle.

Lemma 1 (*Superposition principle*). *Suppose $p_1, p_2, \ldots, p_m$ are m data particles in a data space, and they belong to the same data class. The gravitations they act on another data particle are $F_1, F_2, \ldots, F_m$, and then the composition of gravitation is:*

$$F = \sum_{i=1}^{m} F_i \tag{3}$$

Definition 6 (*Data gravitation field*). Data particles act on each other by data gravitation, and form a field that congests the whole data space. Data gravitation field can also be compared. Data particles in different data classes produce different data gravitation fields, and these fields can be compared.

3.2. The law of data gravitation

The strength of gravitation between two data particles in data space is the direct ratio to the product of data mass of the two data particles, and reverse ratio to the square of distance between them. The law can be described as follows:

$$F = \frac{m_1 m_2}{r^2} \tag{4}$$

where$F$ gravitation between two data particles.
m_1 data mass of data particle 1.
m_2 data mass of data particle 2.
r the Euclidean distance between the two data particle in data space.

3.3. Classifier

Lemma 2. *Suppose c_1, c_2 are two data classes in a training data set. For a given test data element P (an atomic data particle), the gravitation that data particles in c_1 acts on P is F_1, and F_2 is the gravitation that data particles in c_2 acts on P. If $F_1 > F_2$, then the degree P belongs to c_1 is stronger than that to c_2.*

Fig. 1 describes the principle of classification.

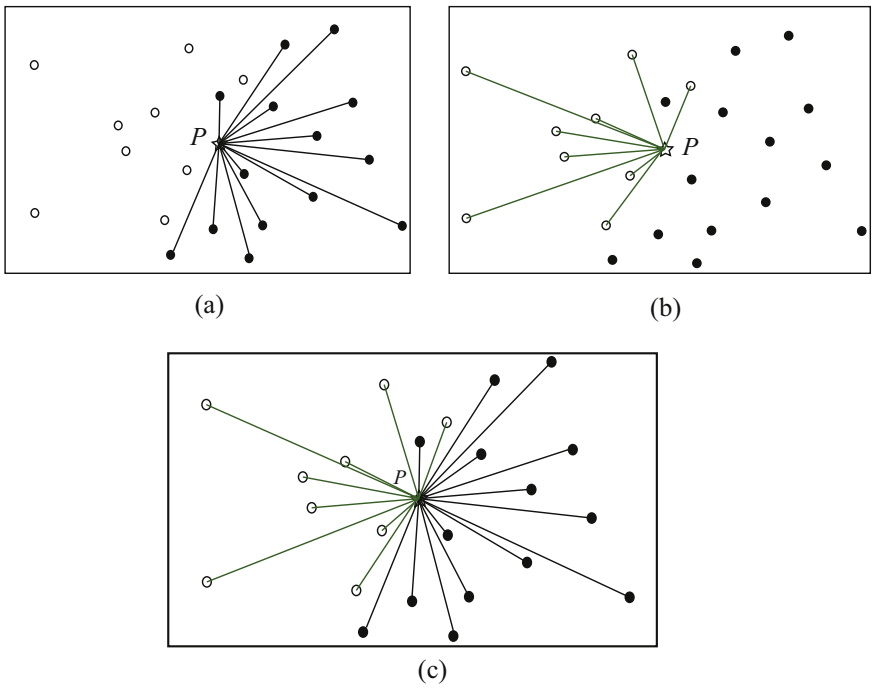

Fig. 1. Classification based on data gravitation. The strength of gravitation determine which class a test data element belongs to. The black dots denote data particles in class c_1. The circles denote data particles in class c_2.

Suppose $T = \{\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \ldots, \langle x_l, y_l\rangle\}$ is a training set in n-dimensional data space, $y \in \{c_1, c_2, \ldots, c_k\}, c_i$ represents data class i, k is the number of the data classes, l is the number of training samples. A new set of training data particles is created from the original training set. The new training data particle set is $T' = \{\langle m_1, x'_1, y_1\rangle, \langle m_2, x'_2, y_2\rangle, \ldots, \langle m_l, x'_l, y_l\rangle\}$, where l' is the number of the data particles, $l' \leqslant l, \mathbf{x}'_i$ is the centroid of data particle i, m_i is the data mass of data particle i.

After the training data particle set has been built, the strength of data gravitation field on any position in data space can be calculated. So when a test data element is given, which data class it belong to can be determined by the field strength of different data classes.

Suppose $c_1, c_2, \ldots, c_k$ are the data classes in the training set, they have $l_1, l_2, \ldots, l_k$ samples (data elements), the training data particle set created from the training set has $l'_1 + l'_2 + \ldots + l'_k$ data particles, where l'_i is the number of data particles which belong to data class i. A given test data can be treated as an atomic data particle P, the centroid is its position $\mathbf{x}$. The gravitation that data class i act on it is given by:

$$F_i = \sum_{j=1}^{l'_i} \frac{m_{ij}}{|\mathbf{x}_{ij} - \mathbf{x}|^2} \tag{5}$$

where m_{ij} is the data mass of data particle j in data class i, x_{ij} is its centroid.

If $F_{i'} = max\{F_1, F_2, \ldots, F_k\}$, then according to Lemma 2, the test data element belongs to data class i'.

3.4. Principles to create data particle

The simplest method to create data particle is to treat a single data element as one data particle. By this means, a training sample in training data set can create a data particle. According to how many data elements the original training data set possess, the required number of data particles will be created. This method is simple and easy to realize. Another advantage of this method is that the field strength of data gravitation calculated by this method can achieve the highest accuracy. But the shortage of the method is also obvious: The calculation might grow up tremendously with the expansion of the training data set and the efficiency of classification will be reduced smartly.

Another method to create data particle is the Maximum Distance Principle (MDP).

Algorithm 1.

(1) For a given distance threshold ε and a training sample $\mathbf{x}_0$, If there are some other training sample $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$ belong to the same data class with $\mathbf{x}_0$, they make that $|\mathbf{x}_i - \mathbf{x}_0| < \varepsilon, i = 1, 2, \ldots, p$. Then $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$ can be combined into a single data particle $\langle m_0, \mathbf{x}'_0\rangle$, where $\mathbf{x}'_0$ is centroid of the data particle, $m_0 = p + 1$ is its data mass.

(2) Repeat the algorithm of (1) with the new created data particle $(\mathbf{x}'_0, m_0)$ to find other training samples that match the condition, if found, combine them to the data particle, otherwise, this data particle is created finally.

Fig. 2 illustrates the main flow of MDP algorithm.

The advantage of MDP method is that it can combine data elements that have similar influence on data gravitation field, so the efficiency of classification can be enhanced remarkably. But the method reduced the accuracy of the calculation of the strength of data gravitation field, especially on the area nearby the data particle centroid. The data gravitation field around the data particle centroid formed by the original data was relatively complex, but after the data particle was created, some information of data gravitation field was discard, and this would reduce the accuracy of classification.

3.5. Weighting to features

DGC uses weight to measure the importance of a feature (input). Suppose there are n features in target problem feature set, and every feature has a weight value, so all feature weights form a n-expression: $\langle w_1, w_2, \ldots, w_n \rangle$, We mark it as the feature weight vector **w**. As the features have been weighted, the distance between two data particles in the data space is not their Euclidean distance in the data space, but the weighted distance:

$$r' = \sqrt{\sum_{i=1}^{n} w_i (x_{1i} - x_{2i})^2} \tag{6}$$

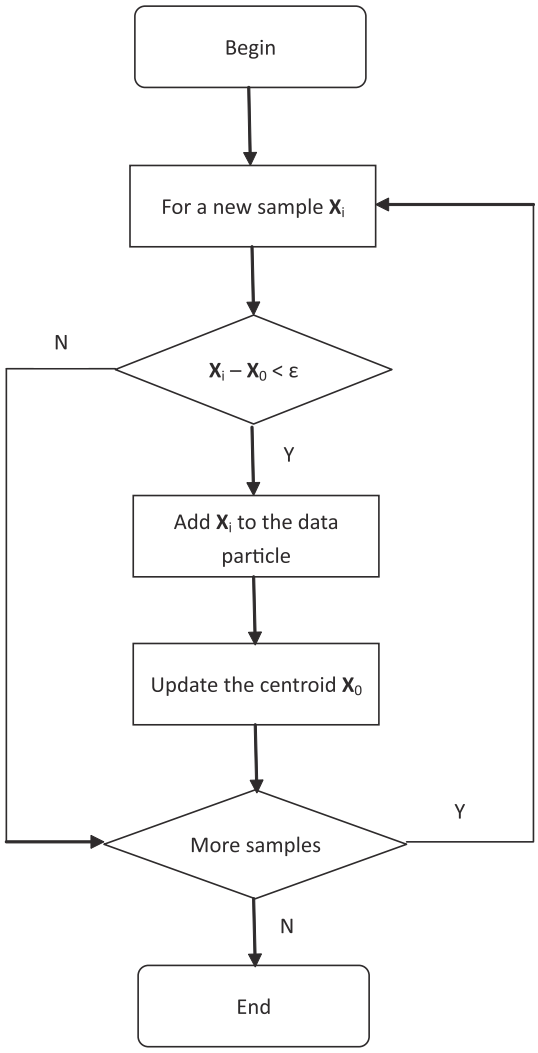

Fig. 2. Flow char of MDP.

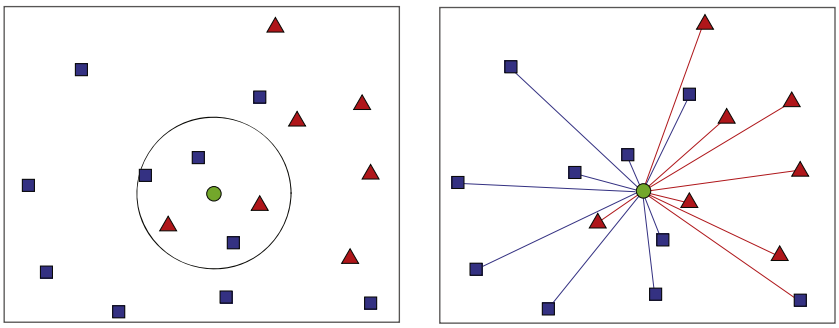

Fig. 3. The difference between KNN and DGC. The green circle point is a unknown target instance. Blue square points are instances of one class and red triangular points are instances of another class. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The calculation of the data gravitation is also changed as:

$$F = \frac{m_1 m_2}{r'^2} \tag{7}$$

3.6. Comparison of DGC, KNN and K-means

DGC shares some characteristics with KNN with regard to the geometrical relationship among data instances [5,13,15]. However, KNN focuses on local relationships, whereas DGC focuses on global relationships. The main principles of KNN and DGC are compared in Fig. 3. KNN first attempts to find the k-nearest neighbors of the target instance according to the Euclidean distance (for the example in Fig. 3, k is 5). The target instances are then classified according to the instance quantity comparison among different classes. KNN only considers a local area around the target instance. For Fig. 3, the local area is depicted by a circle with three blue square points and two red triangular points. All other instances outside the circle have no effect on the classification procedure of the target instance. Unlike the KNN algorithm, DGC includes the whole data space in its classification procedure. DGC calculates the gravitation between the target instance and every training instance and obtains the gravitation of each class. The target instance is classified by comparing the resultant gravitations. Any instance in the data space affects the classification procedure of the target instance.

K-means [27,65] is a cluster algorithm based on distance which also shares some features with DGC. K-means firstly finds k initial instances as k cluster centers using a specific strategy. Then it clusters neighbor instances around the cluster centers according to a minimum distance. Like KNN, K-means also focuses on the local area around a specific cluster center which makes the major difference in comparing with DGC. In fact, it can be seen from Section 3.4 that our algorithm of creating data particles just refers the model of K-means.

4. Imbalanced DGC model

The performance of the DGC model is high in general classification tasks but suffers in imbalanced data. Similar to other general classification methods, the basic DGC model does not include any mechanism to manage imbalanced class distributions. Therefore, the DGC model usually identifies most or even all minority class instances as majority class instances in imbalanced cases. This function makes the basic DGC model unsuitable for most imbalanced classification tasks because the most important goal of these tasks is to recognize rare minority class instances from a large number of majority class instances. In this study, we attempt to design an effective mechanism to enhance the ability of the DGC model in managing imbalanced data. And we call the improved model as imbalanced DGC model (IDGC).

4.1. Gravitation

The main aim of the IDGC model is to strengthen and weaken the gravitational fields of the minority and majority classes, respectively. In the basic DGC model, the gravitation of a certain class depends on the mass of the data particles of this class and their distance from the test data instance. The gravitation calculation is the same for different classes. For imbalanced classification, the gravitation calculation method will yield extremely strong and weak gravitational fields for the majority and minority classes, respectively. This result is obtained because of the limited instances of the minority class. When a testing minority class instance is classified, the strong gravitational field of the majority class will attracts the testing instance as a majority class instance with a large probability, and a false negative (FN) instance will be created. Therefore, the gravitational field strength of the majority and minority classes should be weakened and strengthened to some extent, respectively.

For an imbalanced data set, suppose that k classes $c_1, c_2, \ldots, c_k$ and $N_1, N_2, \ldots, N_k$ are the number of instances. We introduce a new coefficient for class c_i, i.e., the amplified gravitation coefficient (AGC) $\gamma_i, i = 1, 2, \ldots, k$, which is defined as follows:

$$\gamma_i = \frac{\sum_{j=1}^{k} N_j}{N_i} \tag{8}$$

γ_i is the reciprocal of the proportion of the number of instances in class c_i. Suppose that n features (inputs) exist in the target problem, and class c_i has l_i data particles that have the following centroids: $\mathbf{x}_1, \mathbf{x}_2, \ldots,$ and $\mathbf{x}_{l_i}$. For the testing instance (or position in the problem space) $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, we use γ to adjust the gravitation calculation as follows:

$$F_i = [G_1, G_2, \ldots, G_{l_i}] \cdot \begin{bmatrix} \gamma_i \\ \gamma_i \\ \vdots \\ \gamma_i \end{bmatrix} \tag{9}$$

where $G_j = \frac{m_j}{|\mathbf{r}_j|^2}$ is the original gravitation from data particle j, m_j is the data mass of the particle, and $\mathbf{r}_j$ is the Euclidean distance between the particle and testing instance/position. If $\mathbf{w} = [w_1, w_2, \ldots, w_n]^T$ is the feature weight vector, we use the weighted distance in Eq. (6) to obtain the following expression:

$$F_i = \left[\frac{m_1}{\sum_{j=1}^{n} w_j (x_{1j} - x_j)^2}, \frac{m_2}{\sum_{j=1}^{n} w_j (x_{2j} - x_j)^2}, \ldots, \frac{m_{l_i}}{\sum_{j=1}^{n} w_j (x_{l_i j} - x_j)^2} \right] \cdot \begin{bmatrix} \gamma_i \\ \gamma_i \\ \vdots \\ \gamma_i \end{bmatrix} \tag{10}$$

and

$$F_i = \sum_{k=1}^{l_i} \frac{\gamma_i m_k}{\sum_{j=1}^{n} w_j (x_{kj} - x_j)^2} \tag{11}$$

The above formula is also used to calculate the gravitation of class c_i. Finally, the maximum value principle is applied to obtain the classification result of $\mathbf{x}$:

$$cls(\mathbf{x}) = ID(MAX\{F_1, F_2, \ldots, F_k\}) \tag{12}$$

where $ID(\cdot)$ is the operator used to obtain the class label value. The key problem of the model is finding a group of effective feature weights.

4.2. Feature weight optimization using PSO

We use particle swarm optimization (PSO) [35,66] to optimize the feature weights. PSO has been proven as an effective random optimization algorithm in many studies. Comparing with evolutionary algorithms such as genetic algorithm (GA), PSO has no crossover and mutation operations, each particle moves according its velocity and the global sharing information. In GA, any pair of individuals (chromosomes) may exchange their information by the crossover operation. For PSO, all particles only receive the information from the global best particle and the best particle of the current population [17]. In many applications, PSO is proven to converge in fewer generations [47], or have better computational efficiency [29] than GA. Yet, GA also has advantages in comparing with PSO, and the further comparisons between the algorithms is beyond the scope of the paper.

PSO [35,66] conducts searches by using a population of particles that correspond to individuals in an evolutionary algorithm. A population of particles is randomly generated initially. Each particle represents a potential solution and has a position that is represented by position vector $\bar{\mathbf{x}}_i$. A swarm of particles moves through the problem space, and the velocity of each particle is represented by velocity vector $\bar{\mathbf{v}}_i$. At each time step, a function f_i representing a quality measure is calculated by using $\bar{\mathbf{x}}_i$ as the input. Each particle monitors its own best position, which is associated with the best fitness achieved by a particle in vector $\bar{\mathbf{p}}_i$. Furthermore, the best position among all particles obtained in the population is tracked as $\bar{\mathbf{p}}_g$. The best position signifies the best global solution obtained so far. In addition to the global version, another PSO version monitors the best position among all the topological neighbors of a particle (for more information on neighborhood topology, refer to [36,38]).

At each time step t, by using the individual best position, $\bar{\mathbf{p}}_i(t)$, and the global best position, $\bar{\mathbf{p}}_g(t)$, a new velocity for particle i is updated by

$$\bar{\mathbf{v}}_i(t+1) = \bar{\mathbf{v}}_i(t) + c_1\phi_1(\bar{\mathbf{p}}_i(t) - \bar{\mathbf{x}}_i(t)) + c_2\phi_2(\bar{\mathbf{p}}_g(t) - \bar{\mathbf{x}}_i(t)) \tag{13}$$

where c_1 and c_2 are positive constant and ϕ_1 and ϕ_2 are uniformly distributed random numbers in [0,1]. The term $\bar{\mathbf{v}}_i$ is limited to the range of $\pm\mathbf{v}_{max}$. If the velocity violates this limit, it is set to its proper limit. Changing velocity this way enables the particle i to search around its individual best position, $\bar{\mathbf{p}}_i$, and global best position, $\bar{\mathbf{p}}_g$. Based on the updated velocities, each particle changes its position according to the following equation:

$$\bar{\mathbf{x}}_i(t+1) = \bar{\mathbf{x}}_i(t) + \bar{\mathbf{v}}_i(t+1) \tag{14}$$

Based on Eqs. (13) and (14), the population of particles tend to cluster together with each particle moving in a random direction. This may result in premature convergence in many problems. An effective method to avoid premature convergence is to update the velocity by the following formula [36]:

$$\bar{\mathbf{v}}_i(t+1) = \chi(\omega\bar{\mathbf{v}}_i(t) + c_1\phi_1(\bar{\mathbf{p}}_i(t) - \bar{\mathbf{x}}_i(t)) + c_2\phi_2(\bar{\mathbf{p}}_g(t) - \bar{\mathbf{x}}_i(t))) \tag{15}$$

where two new parameters, χ and ω, are also real numbers. The parameter χ controls the magnitude of $\bar{\mathbf{v}}$, whereas the inertia weight ω weights the magnitude of the old velocity $\bar{\mathbf{v}}_i(t)$ in the calculation of the new velocity $\bar{\mathbf{v}}_i(t+1)$.

In DGC model, we use n real values in [0, 1] to create a real value sequence, and the sequence is a PSO particle. If the population size is s, then we generate s such sequences which make up the population for each PSO iteration. Here, n is the feature number of the target problem. Therefore, a single particle is an alternative solution of the feature weights $\langle w_1, w_2, \ldots, w_n\rangle$ defined in Section 3.5. And the fitness evaluating method will be illustrated in the next subsection in detail. By this mean, PSO algorithm is used to optimize the feature weights of the DGC model.

4.3. Fitness evaluation

The basic DGC model uses the tentative random feature selection algorithm to optimize the feature weights. And the feature weights are evaluated by using the simple error rate:

$$fitness(\mathbf{w}) = \frac{\sum_{i=1}^{k} err_i}{\sum_{i=1}^{k} N_i} \tag{16}$$

where err_i is the number of misclassified instances in class c_i. This fitness evaluation method is not effective for imbalanced tasks because this method may obtain high false negative rates (FNRs) for the minority class. A low fitness value calculated by Eq. (16) may correspond to a high misclassification rate for the minority class.

In I model, we apply the AGC γ to evaluate the feature weights:

$$fitness(\mathbf{w}) = \sum_{i=1}^{k} \gamma_i \frac{err_i}{N_i} \tag{17}$$

By integrating Eq. (8) in the aforementioned equation, we obtain the following:

$$fitness(\mathbf{w}) = \sum_{i=1}^{k} N_i \sum_{i=1}^{k} \frac{err_i}{N_i^2} \tag{18}$$

This equation shows that the misclassification of a minority class instance is severely penalized by a significant increase in fitness value. For a misclassified majority class instance, the penalty is relatively light. This fitness evaluation significantly decreases the FN rate of the minority class and slightly increases the FN rate of the majority class. In imbalanced classification tasks, the most important objective is the attainment of a high hit rate (TPR) for the minority class. Total classification accuracy is considered only a secondary objective in these tasks. Therefore, we use Eq. (18) to evaluate the feature weights in the imbalanced DGC model.

5. Experimental settings

In this section, we present the settings used for the performance evaluation of the experiments. A total of 44 imbalanced binary class data sets and 15 imbalanced multiclass data sets were selected from the KEEL data set repository [1] (http://www.keel.es/imbalanced.php). The imbalance ratios of these data sets range from 1.50 to 853.00 and their characteristics are depicted in Section 5.1. And the statistical test methods used in this study are introduced in Section 5.2. Three classic classification algorithms are employed for comparisons. For binary imbalanced tasks, we combine the algorithms with several data-level preprocessing, cost-sensitive, and ensemble techniques. Section 5.3 briefly describes these algorithms, and Section 5.4 discusses the performance measures used in this study.

5.1. Data sets

Table 1 shows the characteristics of the binary class data sets, where #inst denotes the number of instances of the data set, #attrs denotes the number of attributes (features) of the data set, and IR is the imbalance ratio. Table 1 is sorted by IR in ascending order. According to the KEEL data set repository, 22 of these data sets are low imbalanced with IR < 9.0, and the

Table 1
Characteristics of selected binary class data sets.

Low imbalanced				High imbalanced			
Data set	#inst	#attr	IR	Data set	#inst	#attr	IR
glass1	214	9	1.82	yeast-2_vs_4	514	8	9.08
ecoli-0_vs_1	220	7	1.86	yeast-0-5-6-7-9_vs_4	528	8	9.35
wisconsinImb	683	9	1.86	vowel0	988	13	9.98
pimaImb	768	8	1.87	glass-0-1-6_vs_2	192	9	10.29
iris0	150	4	2.00	glass2	214	9	11.59
glass0	214	9	2.06	shuttle-c0-vs-c4	1829	9	13.87
yeast1	1484	8	2.46	yeast-1_vs_7	459	7	14.30
habermanImb	306	3	2.78	glass4	214	9	15.46
vehicle2	846	18	2.88	ecoli4	336	7	15.80
vehicle1	846	18	2.90	page-blocks-1-3_vs_4	472	10	15.86
vehicle3	846	18	2.99	abalone9-18	731	8	16.00
glass-0-1-2-3_vs_4-5-6	214	9	3.20	glass-0-1-6_vs_5	184	9	19.44
vehicle0	846	18	3.25	shuttle-c2-vs-c4	129	9	20.50
ecoli1	336	7	3.36	yeast-1-4-5-8_vs_7	693	8	22.10
new-thyroid1	215	5	5.14	glass5	214	9	22.78
new-thyroid2	215	5	5.14	yeast-2_vs_8	482	8	23.10
ecoli2	336	7	5.46	yeast4	1484	8	28.10
segment0	2308	19	6.02	yeast-1-2-8-9_vs_7	947	8	30.57
glass6	214	9	6.38	yeast5	1484	8	32.73
yeast3	1484	8	8.10	ecoli-0-1-3-7_vs_2-6	281	7	39.14
ecoli3	336	7	8.60	yeast6	1484	8	41.40
page-blocks0	5472	10	8.79	abalone19	4174	8	129.44

other 22 data sets are high imbalanced with IR $\geqslant$ 9.0. Table 2 lists the characteristics of the imbalanced multiclass data sets. More than two classes are included in each imbalanced multiclass data sets. Each data set also contains a minority class that has a significantly smaller amount of instances than the other classes.

K-folder cross-validation is a popular and effective experimental validation procedure for classifying tasks. In this study, the 5-folder cross-validation is used. Each selected data set is uniformly divided into 5 subsets, i.e., both positive and negative classes are uniformly distributed in all subsets. The training and testing processes of each data set are conducted five times. For the *i*th time, the *i*th subset is used as the testing set, and the other 4 subsets are used as the training set. The main reason that we choose the 5-folder cross-validation instead of the standard 10-folder cross-validation is the lack of minority class instances in imbalanced data sets. Particularly for some small imbalanced data sets, the minority class instances are extremely few. For example, the glass-5 data set only has 9 positive instances. According to [41], the partition of an imbalanced data set should be deliberate with regard to the distributions of the data. A 10-folder cross-validation will generate a subsets without any positive instance for the glass-5 data set.

5.2. Statistical tests

For an experimental study that compares different algorithms, a statistical analysis is usually conducted to find if significant differences exist among the results of the algorithms [23,24]. In this study, we apply Friedman's non-parametric test and Wilcoxon signed-rank test for the statistical analysis.

Table 2
Characteristics of selected multiclass data sets.

Data set	#inst	#attr	IR
Wine	178	13	1.50
Hayes-roth	132	4	1.70
Contraceptive	1473	9	1.89
Penbased	1100	16	1.95
New-thyroid	215	5	4.84
Dermatology	366	34	5.55
Balance	625	4	5.88
Glass	214	9	8.44
Autos	159	25	16.00
Yeast	1484	8	23.15
Thyroid	720	21	36.94
Lymphography	148	18	40.50
Ecoli	336	7	71.50
Pageblocks	548	10	164.00
Shuttle	2175	9	853.00

- **Friedman's test:** We firstly use Friedman's non-parametric test in our study because the basic conditions of parameter tests cannot be satisfied. Considering that 17 methods are used in our study on binary classifications, we do not use the $n \times n$ pairwise comparison because the comparison results will be too many to include in the paper. Furthermore, the goal of our test is to find out whether IDGC performs better than the other compared methods. Thus, an $1 \times n$ comparison is relatively reasonable for this study.
  For a multiple hypothesis testing, a post hoc procedure is usually applied to determine whether a hypothesis of mean comparison could be rejected at a specified significance level α. In most cases, we not only concern about the p-value of each hypothesis in the family but also the lowest significance that will result in a rejection. The lowest significance is called adjusted p-value (APV). And a post hoc procedure is one to search the APV for each hypothesis. In our study, we use Finner's test as the post hoc procedure which is able to offer good results and is easy to comprehend [24].
  The average rank is also applied in this study. For a specific data set, the method that achieves the highest performance measure value is ranked at the first position and its rank value is set as one. The method that achieves the second highest value is given a rank value of two, and so forth. Finally, the average rank of each method is computed for comparison.
- **Wilcoxon signed-rank test:** We also employ Wilcoxon signed-rank test, which is also a non-parametric test for performing pairwise comparison between two methods [55]. We just apply Wilcoxon test for those methods which do not show significant performance differences in Friedman's test. Wilcoxon test firstly calculates the differences between the measure scores of the compared methods on the ith out of N data sets. And then, the differences are ranked according to their absolute values in ascending order. A positive difference value means that the first method outperformed the second one on the corresponding data set, and vice versa. The sum of the positive difference values is called R^+, which is the quantized advantage of the first method over the second one. Inversely, R^- is the sum of the negative difference values and means the advantage of the second method over the first one. If the difference between R^+ and R^- is large enough, the null hypothesis of equality of means will be rejected. Like Friedman's test, this statistical test is also used to determine whether a hypothesis of mean comparison could be rejected at a specified significance level α. p-value is also important for Wilcoxon test, which represents the lowest level of significance that results in a rejection of the hypothesis.

5.3. Algorithms selected for the study

We first use 3 data-level approaches, namely, SMOTE [11], SMOTE-ENN [4], and SMOTE-TL [4], for data resampling. We then integrate these approaches into 3 widely used classifiers, namely, C4.5 [51], KNN [13], and SVM [60]. Thus, 9 data-level methods, i.e., SMOTE+(C4.5, KNN, and SVM), SMOTE-ENN+(C4.5, KNN, and SVM), and SMOTE-TL+(C4.5, KNN, and SVM) are used. Cost-sensitive versions of C4.5 [58] and SVM [59] are used as cost-sensitive learning methods. Finally, 4 ensemble learning methods, including SMOTEBagging [61], SMOTEBoost [12], BalanceCascade [40], and EasyEnsemble [40], are applied. And all of the ensembles use C4.5 as the baseline classifier. All these methods are executed by KEEL software [2],[1] which provides a powerful platform for our studies. In addition, we also employ the basic DGC model integrated with SMOTE in the experiments. DGC+ [8] is also an important algorithm used in our experiments, since it is another enhanced DGC model for imbalanced classification tasks. DGC+ runs on WEKA [62], which is a widely used open source Java software for data mining, as a plug-in WEKA module. And we will compare the results of DGC+ and our method individually.

As far as we know, only a few open approaches are available for imbalanced multi-classification tasks. Thus, we use the three standard algorithms in our experiments: C4.5 [51] for decision trees, KNN [13] for lazy learning, and SVM [60]. And we apply two binarization techniques which have been proven to be effective for imbalanced multi-classification tasks [20]:

- One-versus-one method: The OVO method tries to train a binary classifier for each possible pair of classes in a multiclass data set [28]. And the classifier of a class pair ignores the instances from other classes. By this mean, the OVO method trains $n(n-1)$ binary classifier for a n classes data set. For a testing instance, all of the binary models are demand to test it. And the testing results are combined as the overall classifying result.
- One-versus-all method: Different from the OVO method, the OVA method trains a binary classifier for each single class [52], and the classifier regards all instances from other classes as negative instances. Thus, the OVA method also builds a set of binary classifiers. However, the size of the set is n. The testing procedure is similar to that of OVO: Each binary classifier predicts the testing instance, and all results are combined into the overall result.

Table 3 lists all algorithms applied in this study. We cite the original literature of each algorithm in the table. Readers can find technical details of each algorithm in its corresponding literature. The detailed parameter settings of each algorithm can be found in Table 4.

5.4. Performance measures

The confusion matrix is the basis in measuring a classifier, wherein rows denote the actual class of the instances and the columns denote the predicted class. Fig. 4 shows a typical confusion matrix of a binary classification. True positive (TP) is the

[1] The KEEL software is available on web site http://www.keel.es/.

Table 3
Algorithms selected for the study.

Type	Algorithm
Standard	C4.5 [51] SVM [60] KNN [13]
Data-level processing	SMOTE [11] + DGC SMOTE + C4.5 SMOTE + SVM SMOTE + KNN SMOTE-ENN [4] + C4.5 SMOTE-ENN + SVM SMOTE-ENN + KNN SMOTE-TL [4] + C4.5 SMOTE-TL + SVM SMOTE-TL + KNN
Cost-sensitive learning	C4.5CS [58] SVMCS [59]
Ensembles	SMOTEBagging [61] SMOTEBoost [12] BalanceCascade [40] EasyEnsemble [40]
Integrated model	DGC+ [8]

Table 4
Parameter settings of the compared algorithms.

Algorithm	Parameters
IDGC	Radius of data particle = 0.01, PSO pop size = 30, Max iteration time = 500
DGC	Radius of data particle = 0.01, PSO pop size = 30, Max iteration time = 500
DGC+	Generations = 500, Pop size = 0, ProblemType = Imbalanced data sets, seed = 123456789
SMOTE	#Neighbors = 5, Type of SMOTE = both, Balancing = Y, Quality of generated examples = 1, Type of Interpolation = standard, $\mu = 0.5$, $\alpha = 0.5$

SMOTE-ENN	#Neighbors ENN = 3, #Neighbors SMOTE = 5, Type of SMOTE = both,
	Balancing = Y, Quality of generated examples = 1

SMOTE-TL	#Neighbors = 5, Type of SMOTE = both, Balancing = Y, Quality of generated examples = 1
C4.5	Pruned = true, Confidence = 0.25, Instance per leaf = 2
C4.5CS	Pruned = true, Confidence = 0.25, Instance per leaf = 2, minimumExceptedCost = true
SMOTEBagging	Pruned = true, Confidence = 0.25, Instance per leaf = 2, #Classifier = 10
SMOTEBoost	Pruned = true, Confidence = 0.25, Instance per leaf = 2, #Classifier = 10, Quality of balancing SMOTE = 100, Training method = Noresampling
BalanceCascade	Pruned = true, Confidence = 0.25, Instance per leaf = 2, #Classifier = 10 #Bag = 100, Training method = Noresampling
EasyEnsemble	Pruned = true, Confidence = 0.25, Instance per leaf = 2, #Classifier = 10 #Bag = 100, Training method = Noresampling
SVM	Kernel = POLY, $C = 100.0, \epsilon = 0.001$, Degree = 1, $\gamma = 0.01$, coef0 = 0.0, nu = 0.1, $p = 1.0$, shrink = 1
SVMCS	Kernel = POLY, $C = 100.0, \epsilon = 0.001$, Degree = 1, $\gamma = 0.01$, coef0 = 0.0, nu = 0.1, $p = 1.0$, shrink = 1
KNN	K value = 1, Distance Function = Euclidean

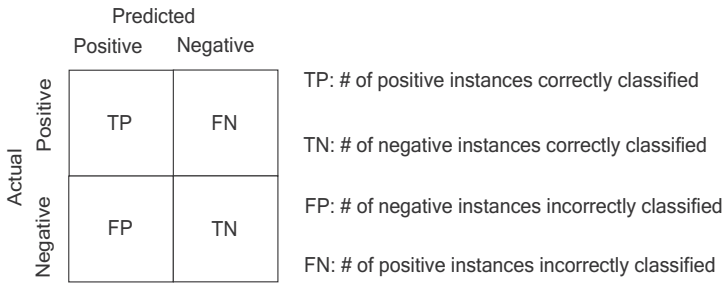

Fig. 4. Confusion matrix.

number of positive instances that are correctly classified, false positive (FP) is the number of negative instances that are incorrectly classified as positive samples, true negative (TN) is the number of negative instances that are correctly classified, and false negative (FN) is the number of positive instances that are incorrectly classified as negative samples. We conduct many types of measures based on the confusion matrix to evaluate classifier performance. For an imbalanced classification task, different classes have different levels of importance. As stated in the introduction, the minority class is more important than the majority class. Furthermore, the difference in the levels of importance has a certain relationship with the IR. Thus, the measures using for general classification tasks cannot effectively measure the performances of classifiers under imbalanced cases. We use the following measures to evaluate the performances of classifiers in the study.

- **Area Under Curve:** The receiver operating characteristic (ROC) curve [6] is a 2D graphical illustration of the trade-off between the TPR and FPR. The TPR is also called sensitivity (*Sens*), and the FPR is related to another general measure namely specificity (*Spec*), and they are defined as follows:

$$Sens = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{19}$$

$$Spec = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} = 1 - \mathrm{FPR} \tag{20}$$

The ROC curve illustrates the behavior of a classifier without considering the class distribution or misclassification cost. An example of an ROC curve is depicted in Fig. 5. The diagonal line represents the trade-off between the sensitivity and 1-specificity for a random model. For a well-performing classifier, the ROC curve needs to be as far to the top left-hand corner as possible, i.e., the area under the ROC curve (AUC) [32] requires to be as large as possible. The AUC is also a commonly used evaluation measure for imbalanced classifications [19]. AUC is computed by the confusion matrix values in relation to the TP rate (TPR) and FP rate (FPR):

$$\mathrm{AUC} = \frac{1 + \mathrm{TPR} - \mathrm{FRP}}{2} \tag{21}$$

Since $Spec = 1 - \mathrm{FPR}$ and $Sens = \mathrm{TPR}$, we get:

$$\mathrm{AUC} = \frac{Sens + Spec}{2} \tag{22}$$

The aforementioned equation shows that AUC, which is the mean value of *Sens* and *Spec*, balances the proportion of the correctly predicted positive and negative instances.

- **Geometric Mean:** Geometric mean (GM) is another method for comparing positive and negative classes regardless of their size [3,39]:

$$\mathrm{GM} = \sqrt{Sens \cdot Spec} \tag{23}$$

Similar to AUC, GM attempts to maximize the accuracy of each class in a balanced manner. GM is a performance metric that links both objectives and has increasingly used in imbalanced classifications.

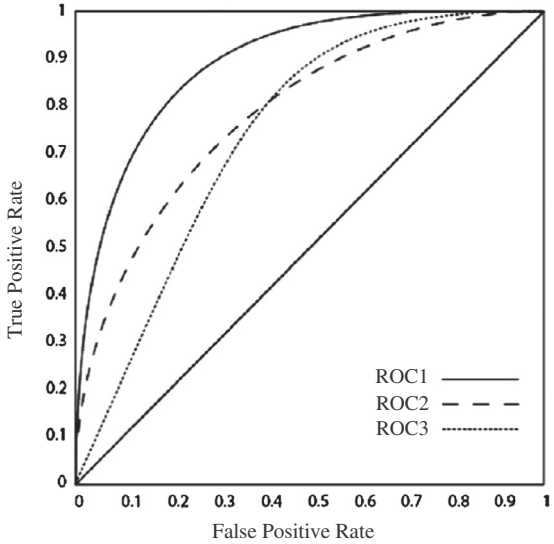

Fig. 5. Example ROC curve.

6. Results and analysis for binary data sets

In this section, we set out to show the experimental results and the statistical analysis on the binary data sets. We firstly give the experimental results in 6.1, and then the statistical analysis will be executed in 6.2. We will compare the results of IDGC and DGC+ in an individual subsection of 6.3, because DGC+ is also an effective improved DGC model for imbalanced classification tasks.

6.1. Results

We summarize the experimental results using the average values in Table 5. And the results of the low imbalanced (IR < 9.0) and the high imbalanced (IR $\geqslant$ 9.0) data sets are displayed separately in the table. The table also shows the average results of all data sets. Abbreviation of SMT for SMOTE is used in the table to save spaces. We also use box plots (Figs. 6 and 7) to show the testing results. We generate the upper and lower edges of each box according to the standard deviation of its measure, and the transverse line in each box is the median line. We also use a small circle in each box to mark the mean value of the measure, the short upper and lower dashes represent the maximum and minimum values respectively. Given the limited length of the paper, the detailed results of AUC and GM are shown in Tables 13–16 of Appendix A. Both the training and testing results are shown in these tables. Abbreviations are also used for the algorithms in the tables to save the page space: SMT is SMOTE, SMTBG is SMOTEBagging, SMTBST is SMOTEBoost, BC is BalanceCascade, and EE is EasyEnsemble.

When observing the box plots, IDGC firstly shows good robustness with regard to both of AUC and GM. The heights of the AUC and GM boxes of IDGC are relatively small in comparing with the other algorithms. That is to say, the standard devia-

Table 5
Average results for binary class data sets.

Algorithm	Low imbalanced				High imbalanced				All data sets			
	AUC		GM		AUC		GM		AUC		GM	
	trn	tst	trn	tst	trn	tst	trn	tst	trn	tst	trn	tst
IDGC	.8959	.8684	.8944	.8666	.8938	**.8634**	.8948	**.8601**	.8967	**.8659**	.8945	**.8633**
SMT + DGC	.8741	.8608	.8721	.8584	.9012	.8550	.8990	.8517	.8881	.8579	.8856	.8550
SMT + C4.5	.9501	.8629	.9497	.8623	.9728	.8168	.9733	.7929	.9618	.8399	.9615	.8276
SMT + SVM	.8849	.8739	.8803	.8692	.8693	.8431	.8659	.8396	.8781	.8585	.8731	.8544
SMT + KNN	.8983	.8525	.8955	.8504	.9498	.8184	.9491	.7896	.9247	.8355	.9223	.8200
SMT-ENN + C4.5	.9266	.8624	.9262	.8616	.9634	.8165	.9642	.7940	.9455	.8394	.9452	.8278
SMT-ENN + SVM	.8825	.8748	.8788	.8713	.8691	.8379	.8670	.8351	.8766	.8563	.8729	.8532
SMT-ENN + KNN	.9106	.8659	.9078	.8649	.9477	.8330	.9471	.8186	.9299	.8495	.9274	.8418
SMT-TL + C4.5	.9410	.8666	.9382	.8651	.9693	.8171	.9701	.7975	.9557	.8419	.9542	.8313
SMT-TL + SVM	.8835	.8711	.8761	.8639	.8668	.8431	.8600	.8365	.8761	.8571	.8680	.8502
SMT-TL + KNN	.9118	.8642	.9064	.8632	.9468	.8339	.9460	.8207	.9300	.8491	.9262	.8419
C4.5CS	.9401	.8610	.9341	.8579	.9805	.8038	.9805	.7723	.9605	.8324	.9573	.8151
SVMCS	.8364	.8249	.7851	.7734	.7738	.7523	.6201	.5894	.8034	.7886	.7026	.6814
SMTBagging	.9429	**.8828**	.9424	**.8817**	.9748	.8277	.9751	.8101	.9591	.8552	.9587	.8459
SMTBoost	**.9806**	.8771	**.9805**	.8756	**.9994**	.8155	**1.0000**	.7803	**.9903**	.8463	**.9902**	.8279
BalanceCascade	.9160	.8632	.9127	.861	.8932	.8232	.8878	.8209	.9045	.8432	.9002	.8409
EasyEnsemble	.9272	.8699	.9259	.8694	.8928	.8333	.8869	.8318	.9100	.8516	.9064	.8506

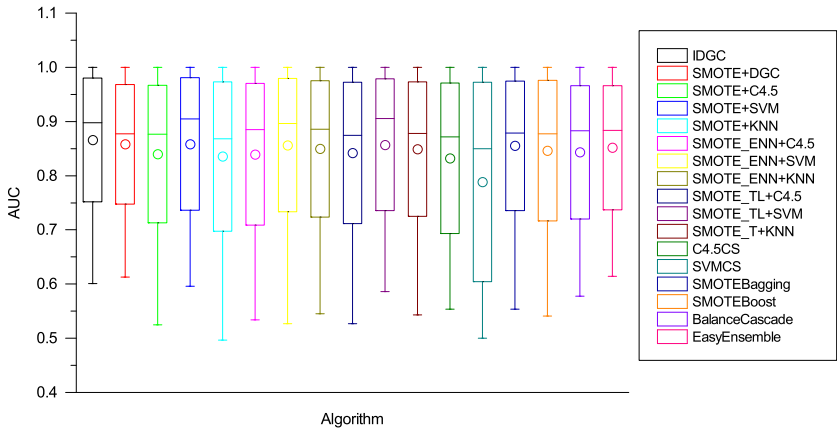

Fig. 6. Testing AUC results for binary class data sets.

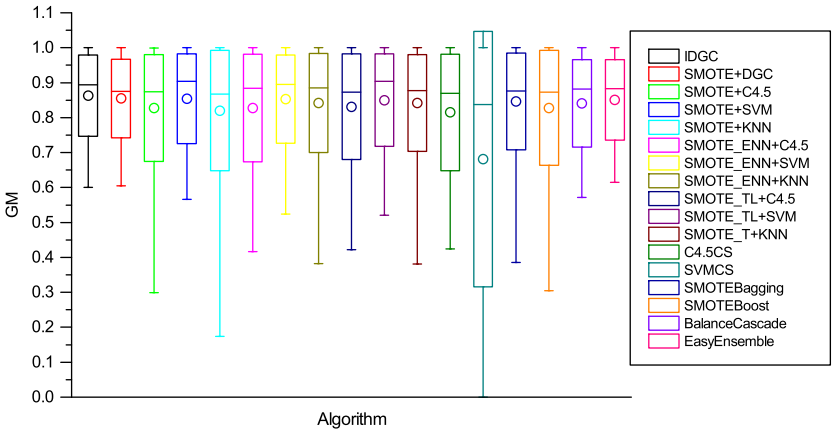

Fig. 7. Testing GM results for binary class data sets.

tions of AUC and GM of IDGC are relatively small. And the differences between the maximum and minimum values are also relatively small. SMOTE + DGC and EasyEnsemble also show good robustness. SMOTE + SVM, SMOTE + ENN + SVM and SMOTE + TL + SVM get high median values for both AUC and GM, it means that the three SVM-based techniques behave fairly well in this study. However, as another SVM-based technique, SVMCS shows does not perform so well according to the box plots. The reasons behind the performance differences between the cost-sensitive method and data-level methods of SVM may lie in the parameter settings, but further discussions with regard to the SVM-based imbalanced algorithms are beyond the scope of this study.

The average results in Table 5 also show some interesting patterns. For IDGC, SMOTE + DGC, SMOTE + SVM, SMOTE + ENN + SVM, SMOTE + TL + SVM and SVMCS, the gaps between their training and testing results are evidentally smaller than that of the other algorithms. Thus, we say that the DGC-based and SVM-based algorithms show good generalization abilities for imbalanced tasks. For the low imbalanced data sets, SMOTEBoost and SMOTEBagging are the best behaved algorithms for the average training and testing results, respectively. While for the high imbalanced data sets, IDGC shows high performances with the highest average testing values for both AUC and GM.

When observing the detailed result tables, SMOTEBoost achieves 100% training accuracies for 39 data sets, thus resulting in the highest value of 1.0 for both AUC and GM. The average values of all measures of SMOTEBoost are also significantly higher than that of the other algorithms, thus indicating that SMOTEBoost outperforms all other algorithms during the training phase. However, SMOTEBoost does not perform so well when observing its testing results. Therefore, SMOTEBoost is overfitted in this case. IDGC obtains seven of the highest testing values for each of AUC and GM. The results show that IDGC can effectively balance the sensitivity and specificity in imbalanced tasks. DGC also shows good performances for AUC and GM when integrated with SMOTE. SMOTE + DGC achieves seven of the highest AUC values and six for GM. The averages of these AUC and GM values are ranked at the third and second position, respectively.

6.2. Statistical analysis

To determine whether significant differences exist among the results of the compared algorithms, we use Friedman's non-parametric test. Finner's test is also employed as the post hoc procedure to compute the APVs. The statistical analysis is performed on the testing results of the 22 high imbalanced data sets. For the low imbalanced cases, it can be observed from the results that IDGC is not the best performed method, and the statistical analysis is not necessary for these cases. IDGC is the control method in Friedman's test. For each measure, we obtain the average ranking, p-value and APV of all compared algorithms. Table 6 shows the statistical testing results. The results are sorted by average ranking in ascending order. We give the p-value of the Friedman's test at the last row. And we also use the abbreviation of SMT for SMOTE to save page spaces.

When observing the results in Table 6, IDGC outperforms all the other compared algorithms except SMOTE + DGC in the multiple hypothesis testing. The Friedman's test p-values of AUC and GM are 0.001715 and 0.001482 respectively, which result in the rejections of the null hypotheses of equality among the 17 methods. As the control method, IDGC gets the lowest average ranking values for both AUC and GM measures. And all the p-values of the multiple hypothesis testing are lower than their corresponding APVs except the case of SMOTE + DGC which result in the rejections of null hypotheses of equality.

6.3. Results comparing between IDGC and DGC+

We compare the experimental results of IDGC and DGC+ in this separate subsection because DGC+ is another kind of enhanced DGC model for imbalanced classification tasks [8]. We got the DGC+ module for WEKA with the authors' permission, and used the module to perform 5-folder cross-validation tasks on all of the 44 binary class data sets. As DGC+ module

Table 6
Results of Friedman's test and Finner's post hoc procedure for the high imbalanced binary class data sets.

Algorithms	AUC			Algorithms	GM		
	Rank	*p*-Value	APV		Rank	*p*-Value	APV
IDGC	4.3864	–	–	IDGC	4.6136	–	–
SMT + DGC	7.1591	0.068592	0.050000	SMT + DGC	7.2500	0.083356	0.050000
SMT-TL + SVM	7.9318	0.019879	0.046950	SMT-TL + SVM	7.8409	0.034036	0.046950
SMT + SVM	8.1591	0.013216	0.043889	SMT-ENN + SVM	8.0000	0.026140	0.043889
SMT-ENN + SVM	8.2045	0.012151	0.040819	SMT + SVM	8.0682	0.023274	0.040819
SMT-ENN + KNN	8.5455	0.006302	0.037739	EasyEnsemble	8.3409	0.014364	0.037739
EasyEnsemble	8.7045	0.004566	0.034650	SMT-ENN + KNN	8.6364	0.008240	0.034650
SMTBagging	8.7500	0.004157	0.031550	SMT-TL + KNN	8.6818	0.007541	0.031550
SMT-TL + KNN	8.8409	0.003437	0.028440	SMTBagging	8.9773	0.004157	0.028440
SMTBoost	9.5455	0.000703	0.025321	BalanceCascade	9.2045	0.002568	0.025321
SMT-TL + C4.5	9.8636	0.000321	0.022191	SMT-TL + C4.5	9.7727	0.000703	0.022191
BalanceCascade	9.9545	0.000255	0.019051	SMTBoost	10.0909	0.000321	0.019051
SMT + KNN	9.9773	0.000241	0.015901	SMT + KNN	10.1591	0.000270	0.015901
SMT-ENN + C4.5	10.2273	0.000125	0.012741	SMT-ENN + C4.5	10.3182	0.000179	0.012741
SVMCS	10.2955	0.000104	0.009571	SVMCS	10.3409	0.000169	0.009571
SMT + C4.5	11.0682	0.000011	0.006391	SMT + C4.5	11.0682	0.000022	0.006391
C4.5CS	11.3864	0.000004	0.003201	C4.5CS	11.6364	0.000004	0.003201
Friedmans' test	–	0.001715	–	Friedmans' test	–	0.001482	–

outputs the time taken to build model for a classification task, thus we also compare the time measure of IDGC and DGC+ in addition to AUC and GM. It should be noticed that although both IDGC and DGC+ programs ran on the same machine with four 3.2 GHz CPU cores, the WEKA module of DGC+ is programmed using JAVA language and run on JAVA virtual machine, and IDGC is programmed using C language. There exists some efficiency differences between the two software system. Yet, we do not think the differences are significant enough to impact the fairness of the time comparisons.

We summarize the AUC, GM and time results of IDGC and DGC+ using average values in Table 7. And the detailed results are given in Table 17 in Appendix A.

IDGC gets an obvious advantage over DGC+ for the average results in Table 7. All average training and testing values of AUC and GM of IDGC are greater than that of DGC+, and the average time of IDGC is also far lower than the average time of DGC+. DGC+ is time-consuming due to its large scale of model parameters. And the model building efficiency of IDGC is also not very high: it spent nearly 6 min for each data set averagely. Both IDGC and DGC+ show a good generalization ability as the differences between their training and testing values are quite small.

When observing the detailed results in Table 17, it can be found that IDGC outperforms DGC+ for 29 data sets with regard to the testing results of AUC. For the results of GM, the number is 30. IDGC shows distinct advantages over DGC+ for the two measures. For the time measure, IDGC is also better than DGC+. IDGC spent less time than DGC+ for most of the data sets. Especially for the large data sets, e. g. abalone19 and page-blocks0, the time differences are particularly significant.

Wilcoxon sign-ranked test is applied to compare the testing results of IDGC and DGC+. The results are given in Table 8. It can be seen from the results that all R^+ values are far higher than the corresponding R^- values. And all p-values are also far lower than 0.05. Thus, for AUC, GM and time, the null hypothesis of equality can be rejected under the standard level of significant.

7. Results and analysis for multiclass data sets

In this section, we carry out two set of experiments on the selected multiclass imbalanced data sets. The first one is the OVO method versus the OVA method applying for IDGC, and the results will be discussed in Section 7.1. The second one is the comparison among IDGC, C4.5, KNN and SVM using OVO method, and the results will be analyzed in Section 7.2. For an experimental execution on a single multiclass data set using a specified method, the AUC and GM of each class are computed. And then the average values are used as the final results. We firstly summarize the results by average values in Table 9, and then a box plot (Fig. 8) is given to provide a simple statistical view of the testing results. We show the detailed results in Tables 18 (training) and 19 (testing) in Appendix A in consideration of the constraint of page space.

Table 7
Average results of IDGC and DGC+ for binary class data sets.

Algorithms	AUC		GM		Time (s)
	trn	tst	trn	tst	
IDGC	0.8967	0.8659	0.8951	0.8633	341.91
DGC+	0.8347	0.8183	0.8350	0.7879	5689.63

Table 8
Wilcoxon sign-rank test results of IDGC vs. DGC+ for binary class data sets.

Measures	R+	R−	*p*-Value
AUC	670.0	276.0	0.017088
GM	687.0	259.0	0.009379
time	779.0	211.0	0.000900

Table 9
Average results for multiclass data sets.

Algorithm	AUC		GM	
	trn	tst	trn	tst
IDGC(OVA)	0.8033	0.7699	0.8000	0.7631
IDGC(OVO)	0.8372	0.8006	0.8335	0.7937
C4.5	0.9037	0.7741	0.8948	0.7381
KNN	0.7084	0.7227	0.6642	0.6813
SVM	0.7656	0.7242	0.7656	0.7064

7.1. Result comparing between OVO and OVA methods

It can be seen from Table 9 that all average values of the two measures of IDGC(OVO) are greater than that of IDGC(OVA). Thus, IDGC(OVO) outperforms IDGC(OVA). The box plots also show similar circumstances. In fact, many related works have proven that the OVO method performs better than the OVA method in most cases [20,21]. Both IDGC(OVO) and IDGC(OVO) show small differences between the training results and the testing results in Table 9. This means that the generalization abilities of the two methods are quite well.

To further analyze the testing results of the two method, we apply Wilcoxon sign-rank test, and the results are shown in Table 10. For both of AUC and GM, the *R*+ values are far greater than the *R*− values. According to Wilcoxon test, it means that IDGC(OVO) outperforms IDGC(OVA) for most of the data sets. Furthermore, the *p*-values of AUC and GM are both lower than 0.05. Thus, the null hypothesis of equality between the testing results of IDGC(OVO) and IDGC(OVA) should be rejected under the standard significant level.

7.2. Result comparing of IDGC and the other algorithms

Observing the average results in Table 9, IDGC (IDGC(OVO)) shows an obvious advantage over the other algorithms in the testing phase. It achieves the highest average AUC and GM values. It should be noticed that we execute experiments using the OVO method for all of the compared algorithms, and do not mark "OVO" for C4.5, KNN and SVM in the table any more. All compared algorithms show good generalization abilities except C4.5. C4.5 gains the highest average training values for AUC and GM, but it does not behave so well in the testing phase. The box plots in Fig. 8 show that IDGC and SVM are more robust than the other two algorithms. The boxes generated by the former two are obviously shorter than the boxes generated by the latter two.

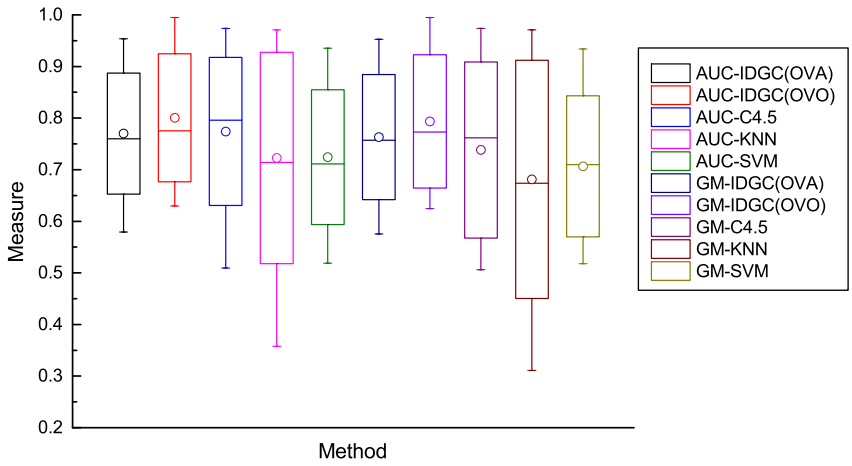

Fig. 8. Testing results for multiclass data sets.

Table 10
Wilcoxon sign-rank test results of IDGC using OVO vs. OVA for multiclass data sets.

Measure	R+	R−	*p*-Value
AUC	108.0	12.0	0.005876
GM	108.0	12.0	0.005876

Table 11
Average Friedman rankings, *p*-values and APVs using Finner's procedure for multiclass data sets using OVO method.

Measure	Algorithm	Ranking	*p*-Value	APV
AUC	IDGC	1.6667	–	–
	C4.5	3.0667	0.002979	0.016952
	KNN	2.5333	0.065992	0.050000
	SVM	2.7333	0.023652	0.033617
	Friedman' test	–	0.021888	–
GM	IDGC	1.7333	–	–
	C4.5	3.0667	0.004678	0.016952
	KNN	2.7333	0.033895	0.033617
	SVM	2.4667	0.119795	0.050000
	Friedman' test	–	0.033862	–

Table 12
Wilcoxon sign-rank test results for multiclass data sets using OVO method.

Measure	IDGC vs.	R+	R−	*p*-Value
AUC	KNN	93.0	27.0	0.057083
GM	KNN	97.0	23.0	0.033183
	SVM	98.0	22.0	0.028768

Like the binary class cases, we firstly apply Friedman's $1 \times n$ test and Finner's post hoc procedure for comparisons. Table 11 gives the test results. We can observe that all *p*-values of Friedman's tests are greater than 0.05. Thus for each measure, there exists significant differences among the results of the compared algorithms. The average rankings show that IDGC is the best behaved algorithm for all measures. However, Finner's test results show that IDGC does not significantly different from the other algorithms in more than half of the cases, and the control method obtains significant advantages over C4.5 for AUC and GM, SVM for AUC.

Wilcoxon sign-rank test is applied again to compare the pairs which do not show significant differences in the Friedman's $1 \times n$ tests, and the results are shown in Table 12. For the AUC comparison between IDGC and KNN, the $R+$ is far higher than $R-$, this means that most testing results of IDGC are better than that of KNN. However, the *p*-value is greater than 0.05, which results in the acceptance of the null hypothesis of equality under the standard significant level. The result is in accord with that of the Friedman's test. For the GM comparing, the circumstance is quite different. IDGC get obvious advantages over both of KNN and SVM in the ranking procedure. And both *p*-values are lower than 0.05, which result in rejections of the null hypotheses of equality. The results are not identical to that of the Friedman's test.

8. Lessons learned

In this study, we have compared our method with 17 imbalanced classification algorithms base on 44 imbalanced binary class data sets, and 3 algorithms based on 15 imbalanced multiclass data sets. Detailed statistical analysis has been carried out using Friedman's test and Wilcoxon sign-ranked test to find out whether there exists significant differences between the experimental results of IDGC and the other algorithms. Several lessons learned can be stressed from the study:

- IDGC is more effective for highly imbalanced data sets than low imbalanced cases. It did not gain significant advantages in comparing with most of the other algorithms for the results of the low imbalanced binary class data sets. However, its results on high imbalanced data sets are much better than the results of the other compared algorithms. The reason lies on the rebalancing mechanisms of IDGC. We designed the amplified gravitation coefficient (AGC) to compute gravitations. AGC will strengthen gravitations from the minority class and weaken that of the majority class. This is the first rebalancing mechanism. AGC is also used to punish incorrectly classified minor class instances in fitness evaluation

procedures. And this is the second rebalancing mechanism. The double mechanisms strongly emphasis the minority class, and weaken the majority class simultaneously. Thus IDGC can get high performances for highly imbalanced data sets.

- A well-performed classifier should consider a good tradeoff between the minority class and majority class for an imbalanced problem. AUC/GM evaluates the performances of a classifier by the arithmetic/geometric mean of the sensitivity and specificity, which are the prediction accuracies of the minority class and majority class, respectively. Both of them can effectively measure the tradeoff ability of a classifier in an imbalanced task. It can be seen from the experimental results that IDGC performed quite well for AUC and GM. Therefore, we say that IDGC is able to make a good tradeoff between the minority class and the majority class in an imbalanced task.
- Although IDGC shows obvious advantage in comparing with DGC+ with respect of the efficency, it is still a time-consuming algorithm, especially for large data sets. The model building procedure is a procedure to find a group of optimum weights. And this is a typical nonlinear optimization problem. In this study, we use PSO algorithm for the optimization. If the populations size is s, the number of instances in the data set is n, and we construct m data particle, then for each PSO iteration, we should do $s \times n \times m$ times of gravitation computations. This is a time consuming procedure. It can be observed from Table 19 that the model building time for large data sets such as page-blocks0, segment0, vehicle0, vehicle1 and vehicle3, exceed 1000 s. So researching for fast and effective weight optimization algorithms of IDGC model is an important future work.
- Multiclass imbalanced classification tasks are relatively difficult for IDGC. Although IDGC shows some advantages in comparing with other algorithms for multiclass data sets, the overall performances of IDGC for binary class data sets are superior to that for the multiclass cases. The rebalancing mechanisms of IDGC strongly stress the function of the minor class. However for a multiclass data sets, over-stress the minor class may result in higher error rates of other classes.
- The over-fitting problem should not be neglected for imbalanced classifications. SMOTEBoost achieved full values for most binary class data sets in training procedures, and outperformed all other algorithms. But it did not show obvious advantages over other algorithms in the testing phases. On the contrary, DGC based and most of SVM based techniques show good generalization abilities in the experiments.

9. Conclusions

We have adapted the DGC model to manage imbalanced classification problems. AGC is used to modify the gravitation computation method and rebalance the gravitational field strength disparity between the majority and minority classes. AGC is also used for the parameter evaluating procedure of the DGC model, and this provide another rebalancing mechanism. We compared the improved DGC model (IDGC) with 17 imbalanced binary classification methods on 44 imbalanced binary-class data sets and 3 multi classifiers on 15 imbalanced multi-class data sets. Experimental results have shown that the two rebalancing mechanisms can effectively overcome the weakness of the basic DGC model for imbalanced tasks. Particularly for highly imbalanced problems, our method showed better performances in comparing with the other algorithms. IDGC also showed good generalization ability in the study. Another characteristic of the IDGC model shown in the study is that it can consider a good tradeoff between the minority and majority classes for an imbalanced task. And this is also important for imbalanced classifiers. We conclude that the IDGC model provides an effective means of solving imbalanced problems. However, the computational complexity of IDGC is relatively high with respect to the weight optimization procedure. It consumes considerable time for large data sets. Thus, researching for fast and effective weight optimization algorithms of IDGC model is one of the important future works of us.

Acknowledgements

This research was partially supported by the National Basic Research Program of China (973 Program) under Grant No. 2011CB302605, the National High Technology Research and Development Program of China (863 Program) under Grant No. 2012AA012502, the National Key Technology R&D Program of China under Grant No. 2012BAH37B00, the Program for New Century Excellent Talents in University under contract number NCET-10-0863, National Natural Science Foundation of China under Grant Nos. 61173078, 61203105, 61173079 and 61373054, the Provincial Natural Science Foundation of Shandong under Grant Nos. ZR2012FM010, ZR2011FZ001 and ZR2012FQ016.

Appendix A. Detailed results of the experimental study

See Tables 13–19.

Table 13
Results of AUC for binary class data sets (training).

Data set	IDGC	SMT DGC	SMT C4.5	SMT SVM	SMT KNN	SMT ENN C4.5	SMT ENN SVM	SMT ENN KNN	SMT TL C4.5	SMT TL SVM	SMT TL KNN	C4.5 CS	SVM CS	SMT BG	SMT BST	BC	EE
abalone19	.7975	.8279	.9321	.8190	.9725	.9315	.8225	.9632	.9476	.8210	.9622	.9839	.8170	.9583	**1.0000**	.7832	.7884
abalone9-18	.8564	.8821	.9559	.8208	.8943	.9546	.8160	.8958	.9573	.8271	.8986	.9864	.8352	.9692	**1.0000**	.8073	.8274
ecoli-0_vs_1	.9369	.9764	.9870	.9844	.9753	.9870	.9811	.9821	.9875	.9835	.9801	.9870	.9675	.9853	**1.0000**	.9826	.9833
ecoli-0-1-3-7_vs_2-6	.9227	.9507	.9891	.9503	.9731	.9881	.9512	.9690	.9836	.9535	.9685	.9804	.8750	.9900	**1.0000**	.8584	.8558
ecoli1	.9108	.9049	.9575	.9062	.9311	.9426	.9037	.9305	.9507	.8985	.9315	.9457	.9083	.9309	**1.0000**	.9246	.9401
ecoli2	.9453	.9269	.9671	.9048	.9569	.9693	.9072	.9542	.9727	.9080	.9507	.9593	.5000	.9724	**1.0000**	.9329	.9303
ecoli3	.9212	.8945	.9767	.8940	.9390	.9667	.8913	.9348	.9639	.8960	.9327	.9585	.8222	.9601	**1.0000**	.9097	.9020
ecoli4	.9700	.9486	.9803	.9691	.9735	.9768	.9664	.9727	.9741	.9640	.9719	.9680	.9834	.9799	**1.0000**	.9412	.9206
glass0	.7719	.8073	.9381	.7727	.8428	.8827	.7594	.8536	.9070	.7725	.8643	.9205	.5215	.9302	**.9991**	.8721	.9016
glass-0-1-2-3_vs_4-5-6	.9183	.9479	.9931	.9324	.9716	.9631	.9427	.9693	.9724	.9393	.9693	.9805	.8571	.9830	**1.0000**	.9574	.9595
glass-0-1-6_vs_2	.8462	.7793	.9571	.6975	.8850	.9632	.6861	.8821	.9571	.6712	.8857	.9829	.5000	.9591	**1.0000**	.8143	.8037
glass-0-1-6_vs_5	.9729	.9193	.9929	.9450	.9600	.9886	.9393	.9693	.9886	.9436	.9714	.9914	.5556	.9761	**1.0000**	.9429	.9429
glass1	.8224	.8560	.8956	.6615	.8248	.8565	.6867	.8638	.9071	.6543	.8491	.9068	.6626	.9180	**1.0000**	.8092	.9044
glass2	.7793	.7938	.9645	.6898	.8991	.9528	.7027	.8921	.9632	.6925	.8928	.9734	.7069	.9390	**1.0000**	.8122	.8261
glass4	.9615	.8980	.9813	.9496	.9583	.9813	.9471	.9608	.9770	.9453	.9596	.9104	.9614	.9795	**1.0000**	.9235	.9583
glass5	.9581	.9152	.9933	.9445	.9640	.9835	.9427	.9677	.9884	.9433	.9671	.9976	.9713	.9939	**1.0000**	.9488	.9488
glass6	.9583	.9527	.9926	.9520	.9676	.9617	.9473	.9777	.9842	.9452	.9784	.9865	.8879	.9810	**1.0000**	.9429	.9449
habermanImb	.7836	.6483	.7725	.6421	.6474	.7163	.6340	.7067	**.7880**	.6590	.7570	.6380	.5220	.7524	.7772	.7110	.7196
iris0	**1.0000**	.9600	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	.9900	**1.0000**	**1.0000**	**1.0000**
new-thyroid1	.9972	.9528	.9938	.9847	.9549	.9860	.9763	.9694	.9944	.9826	.9688	.9903	.9943	.9944	**1.0000**	.9610	.9728
new-thyroid2	.9837	.9646	.9931	.9833	.9604	.9830	.9812	.9625	.9903	.9806	.9646	.9903	.9972	.9844	**1.0000**	.9652	.9854
page-blocks0	.9069	.8921	.9830	.9248	.9460	.9718	.9199	.9486	.9709	.9158	.9471	**.9903**	.9248	.9823	.9869	.9721	.9691
page-blocks-1-3_vs_4	.9865	.9158	.9975	.9673	.9724	.9949	.9555	.9834	.9938	.9581	.9792	.9989	.8510	.9941	**1.0000**	.9696	.9682
pimaImb	.8129	.7745	.8601	.7539	.7389	.8056	.7507	.7812	.8439	.7534	.7783	.8571	.7379	.8237	**.9416**	.8070	.8665
segment0	.9289	.9326	.9974	.9958	.9929	.9976	.9945	.9951	.9969	.9961	.9950	.9988	.9990	.9984	**1.0000**	.9893	.9922
shuttle-c0-vs-c4	.9967	.9960	.9999	.9999	.9996	.9999	**1.0000**	**1.0000**	.9999	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
shuttle-c2-vs-c4	.9934	.9827	.9990	**1.0000**	.9959	**1.0000**	**1.0000**	.9970	**1.0000**	**1.0000**	.9970	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
vehicle0	.9146	.8346	.9843	.9712	.9242	.9723	.9642	.9330	.9734	.9611	.9404	.9861	.9782	.9864	**1.0000**	.9681	.9655
vehicle1	.8382	.7589	.9391	.8252	.7873	.8692	.8204	.8109	.8859	.8207	.8103	.9362	.7929	.9248	**1.0000**	.8695	.8758
vehicle2	.9529	.8692	.9926	.9681	.9162	.9835	.9587	.9303	.9852	.9626	.9260	.9866	.9734	.9928	**1.0000**	.9767	.9792
vehicle3	.8046	.7325	.9299	.8095	.7679	.8709	.7969	.7955	.8862	.8082	.8064	.9221	.8072	.9340	**1.0000**	.8864	.8825
vowel0	.9950	.9776	.9978	.9736	.9912	.9971	.9733	.9975	.9989	.9745	.9974	.9925	.8655	.9976	**1.0000**	.9692	.9730
wisconsinImb	.9705	.9499	.9837	.9781	.9769	.9785	.9800	.9818	.9781	.9802	.9805	.9780	.9724	.9797	**1.0000**	.9737	.9810
yeast-0-5-6-7-9_vs_4	.8757	.8671	.9549	.7962	.9243	.9355	.7920	.9198	.9339	.8022	.9127	.9741	.5000	.9600	**1.0000**	.8780	.8853
yeast1	.7030	.8077	.8049	.7114	.8121	.7801	.7088	.8208	.7974	.7018	.8029	.7855	.6676	.7731	**.8680**	.7915	.8018
yeast-1_vs_7	.7562	.8409	.9502	.7731	.8966	.8932	.7740	.8893	.9273	.7692	.8834	.9741	.5000	.9512	**1.0000**	.8356	.8181
yeast-1-2-8-9_vs_7	.8491	.8616	.9376	.7422	.9327	.9089	.7444	.9278	.9367	.7301	.9238	.9752	.5000	.9675	**1.0000**	.8076	.8037
yeast-1-4-5-8_vs_7	.6252	.8086	.9293	.6882	.9138	.9028	.7030	.8980	.9305	.6834	.8963	.9640	.5000	.9385	**1.0000**	.7715	.7591
yeast-2_vs_4	.9480	.9422	.9852	.9107	.9511	.9720	.9112	.9479	.9814	.9072	.9460	.9798	.5000	.9730	**1.0000**	.9573	.9482
yeast-2_vs_8	.9156	.9662	.9710	.8204	.9573	.9783	.8210	.9518	.9753	.8180	.9543	.9927	.8223	.9797	**1.0000**	.8625	.8692
yeast3	.9275	.8852	.9611	.9125	.9280	.9408	.9093	.9308	.9653	.9186	.9266	.9784	.9057	.9674	**1.0000**	.9489	.9415
yeast4	.8950	.9107	.9716	.8607	.9556	.9468	.8586	.9513	.9636	.8570	.9479	.9722	.8604	.9762	**1.0000**	.8923	.8745
yeast5	.9579	.9467	.9903	.9635	.9817	.9843	.9631	.9807	.9868	.9629	.9798	.9929	.9648	.9888	**1.0000**	.9619	.9597
yeast6	.8847	.9180	.9863	.8861	.9704	.9837	.8867	.9664	.9830	.8884	.9662	.9883	.8807	.9853	**1.0000**	.9095	.9088
Avg.	.8967	.8881	.9618	.8781	.9247	.9455	.8766	.9299	.9557	.8761	.9300	.9605	.8034	.9591	**.9903**	.9045	.9100

Table 14
Results of AUC for binary class data sets (Testing).

Data set	IDGC	SMT DGC	SMT C4.5	SMT SVM	SMT KNN	SMT ENN C4.5	SMT ENN SVM	SMT ENN KNN	SMT TL C4.5	SMT TL SVM	SMT TL KNN	C4.5 CS	SVM CS	SMT BG	SMT BST	BC	EE
abalone19	.6970	.7306	.5245	**.7681**	.4964	.5708	.7671	.5453	.5696	.7660	.5429	.5733	.7627	.5536	.5413	.6957	.6905
abalone9-18	.8249	.8292	.7616	.8583	.7391	.7725	.8612	.7580	.7049	.8539	.7616	.6715	**.8765**	.7951	.7740	.7224	.7703
ecoli-0_vs_1	.9835	**.9870**	.9835	.9800	.9630	.9835	.9835	.9800	.9765	.9765	.9695	.9835	.9675	.9835	.9765	.9800	.9695
ecoli-0-1-3-7_vs_2-6	.8939	.8225	.7602	.8684	.8425	.8261	.8133	**.9067**	.8243	.8775	.9048	.8352	.8571	.8389	.8425	.7367	.7935
ecoli1	.8975	.9014	.9044	.9059	.8673	.8836	.8975	.8849	.8805	.9040	.8798	**.9108**	.9059	.9065	.8640	.8845	.8824
ecoli2	.9078	.9219	.8975	.9131	.9177	.8861	.9095	**.9369**	.8965	.9095	.9263	.8887	.5000	.8861	.8941	.9001	.8799
ecoli3	.8708	.8708	.8535	.8884	.8249	.8375	**.8934**	.8435	.8502	.8850	.8352	.8326	.7924	.8721	.8558	.8691	.8850
ecoli4	.9652	.9636	.9215	**.9668**	.9608	.9168	.9434	.9294	.8449	.9386	.9294	.8636	.9528	.8794	.8889	.8804	.8804
glass0	.7556	.7768	.8173	.7451	.8097	.7930	.7631	**.8424**	.8040	.7598	.8250	.8211	.5073	.8138	.8304	.7580	.8246
glass-0-1-2-3_vs_4-5-6	.9301	.8645	.9068	.9172	.9393	.9178	.8952	**.9497**	.9111	.9178	.9466	.8786	.8437	.9301	.9166	.9111	.8988
glass-0-1-6_vs_2	.7203	.6738	**.7672**	.5958	.6467	.5613	.5267	.6116	.6876	.5861	.6410	.6042	.5000	.6582	.6279	.6975	.7118
glass-0-1-6_vs_5	.9486	.9429	.9686	.9429	.8603	.9130	.9400	.9187	.9187	.9429	.9159	**.9886**	.5000	.8546	.8746	.9429	.9429
glass1	.7843	.7900	.7384	.6404	.7966	.7239	.6487	.7880	.7702	.5900	.7811	.7146	.6260	.7631	**.8354**	.7066	.7525
glass2	.7411	.7011	.7398	.6006	.7281	.7525	.5966	.7575	.7281	.6417	.7231	.6475	.5814	**.8087**	.7383	.6986	.6738
glass4	.9441	**.9627**	.8833	.9118	.8907	.9292	.9143	.8858	.9267	.9068	.8858	.8546	.9242	.8783	.9267	.8819	.9168
glass5	.9585	.9366	.8523	.9366	.9152	.9079	.9390	.9152	.8523	.9415	.8572	.9371	.9732	.9201	**.9878**	.9488	.9488
glass6	**.9267**	.9057	.8695	.9240	.8749	.9104	.9067	.8949	.9003	.9240	.8884	.8868	.8685	.9013	.8803	.9023	.8942
habermanImb	.6420	.6126	.6353	.6185	.5548	.6400	.6449	.6047	.6198	.6114	.5958	.5746	.5405	**.6563**	.6519	.6106	.6398
iris0	**1.0000**	**1.0000**	.9900	**1.0000**	**1.0000**	.9900	**1.0000**	**1.0000**	.9900	**1.0000**	**1.0000**	.9900	**1.0000**	.9800	.9900	.9900	.9900
new-thyroid1	**.9889**	**.9889**	.9690	.9778	.9722	.9353	.9750	.9552	.9718	.9833	.9778	.9746	.9687	**.9889**	.9861	.9413	.9524
new-thyroid2	**.9917**	.9690	.9349	.9750	.9611	.9690	.9694	.9694	.9440	.9611	.9611	.9802	.9829	.9690	.9833	.9385	.9524
page-blocks0	.8688	.8710	.9385	.9222	.9124	.9385	.9197	.9263	.9388	.9170	.9140	.9458	.9254	.9567	.9363	.9540	**.9609**
page-blocks-1-3_vs_4	.9887	.9707	.9809	.9662	.9831	.9641	.9527	.9854	.9775	.9315	.9809	.9810	.8538	.9831	**.9955**	.9640	.9707
pimaImb	.7316	.7215	.7124	.7387	.6847	.7099	.7395	.7056	.6951	.7448	.7059	.7125	.7288	**.7660**	.7377	.7116	.7124
segment0	.9896	.9841	.9911	**.9972**	.9934	.9906	.9929	.9934	.9914	.9939	.9937	.9919	.9965	.9937	.9947	.9810	.9881
shuttle-c0-vs-c4	.9997	.9959	.9997	.9991	.9953	**1.0000**	.9959	.9959	.9997	.9994	.9959	.9997	**1.0000**	.9997	**1.0000**	**1.0000**	**1.0000**
shuttle-c2-vs-c4	.9959	.9959	.9878	.9837	.9959	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	.9959	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
vehicle0	.8976	.8727	.9324	.9608	.9239	.9308	.9616	.9175	.9281	.9587	.9277	.9291	.9494	.9612	**.9681**	.9476	.9388
vehicle1	.7227	.7046	.7062	.8050	.6689	.7194	**.8100**	.7194	.7422	.8065	.6979	.7017	.7554	.7513	.7299	.7241	.7256
vehicle2	.9324	.9146	.9452	.9528	.9314	.9396	.9510	.9328	.9506	.9450	.9320	.9433	.9572	.9669	**.9859**	.9514	.9540
vehicle3	.7015	.6945	.6992	.7897	.6773	.7040	.7912	.6875	.7463	**.7950**	.7188	.7284	.7905	.7512	.7158	.7455	.7558
vowel0	.9889	.9805	.9672	.9660	.9955	.9916	.9722	**.9994**	.9850	.9683	**.9994**	.9422	.8461	.9817	.9911	.9422	.9438
wisconsinImb	.9667	.9491	.9418	.9686	.9614	.9469	.9744	.9704	.9627	.9712	**.9789**	.9636	.9718	.9633	.9652	.9582	.9596
yeast-0-5-6-7-9_vs_4	**.8433**	.8388	.7781	.7870	.7918	.7540	.7912	.8195	.7564	.7926	.8111	.7242	.5000	.8076	.8044	.7912	.7723
yeast1	.7019	**.7266**	.7175	.7015	.6516	.7136	.7126	.6862	.6809	.7000	.6805	.6780	.6747	.7201	.7145	.6941	.7022
yeast-1_vs_7	.7575	.7051	.6731	.7497	.7312	.7242	.7613	.6921	.6994	**.7690**	.7389	.6139	.5000	.7017	.6592	.6915	.7156
yeast-1-2-8-9_vs_7	.6989	**.7033**	.6031	.6957	.5843	.6042	.6984	.6485	.5755	.7025	.6346	.6769	.5000	.6403	.6541	.6445	.6627
yeast-1-4-5-8_vs_7	.6009	**.7273**	.5760	.6357	.5805	.5336	.6448	.6686	.5269	.6321	.6845	.5540	.5000	.6162	.5698	.5773	.6146
yeast-2_vs_4	.9272	.8837	.9001	.8892	.8924	.8717	.8913	.8674	.9316	.8838	.8718	.8881	.5000	.9056	.8880	**.9405**	.9384
yeast-2_vs_8	.8056	.7663	.8055	.7642	.8251	.8023	.7642	.8154	**.8686**	.7609	.8382	.8653	.7663	.7827	.7653	.7365	.7668
yeast3	.9123	.9100	.9004	.9041	.8687	.9084	.9052	.8607	.9146	.9091	.8761	.9118	.8952	**.9398**	.8833	.9315	.9193
yeast4	**.8445**	.8434	.7326	.8133	.7480	.7872	.8252	.7851	.7810	.8203	.8016	.7207	.8161	.7813	.6972	.7949	.8267
yeast5	.9576	.9545	.9489	.9622	.9527	.9513	.9625	.9499	**.9709**	.9622	.9610	.9331	.9656	.9634	.9104	.9518	.9473
yeast6	**.8919**	.8821	.8368	.8866	.8497	.8292	.8723	.8714	.8462	.8713	.8710	.8082	.8758	.8591	.8043	.8718	.8442
Avg.	**.8659**	.8579	.8399	.8585	.8355	.8394	.8563	.8495	.8419	.8571	.8491	.8324	.7886	.8552	.8463	.8432	.8516

Table 15
Results of GM for binary class data sets (training).

Data set	IDGC	SMT DGC	SMT C4.5	SMT SVM	SMT KNN	SMT ENN C4.5	SMT ENN SVM	SMT ENN KNN	SMT TL C4.5	SMT TL SVM	SMT TL KNN	C4.5 CS	SVM CS	SMT BG	SMT BST	BC	EE
abalone19	.7957	.8210	.9308	.8159	.9721	.9302	.8190	.9625	.9473	.8174	.9614	.9838	.8159	.9582	**1.0000**	.7645	.7675
abalone9-18	.8562	.8804	.9559	.8207	.8880	.9544	.8158	.8904	.9570	.8271	.8929	.9863	.8332	.9690	**1.0000**	.8070	.8205
ecoli-0_vs_1	.9369	.9763	.9869	.9843	.9752	.9869	.9810	.9821	.9875	.9835	.9801	.9869	.9670	.9852	**1.0000**	.9826	.9833
ecoli-0-1-3-7_vs_2-6	.9222	.9504	.9890	.9490	.9727	.9881	.9499	.9685	.9834	.9523	.9680	.9802	.8660	.9899	**1.0000**	.8556	.8489
ecoli1	.9086	.9038	.9569	.9039	.9290	.9421	.9014	.9279	.9499	.8956	.9289	.9452	.9060	.9296	**1.0000**	.9229	.9392
ecoli2	.9451	.9257	.9671	.9046	.9559	.9693	.9069	.9531	.9726	.9075	.9494	.9590	.0000	.9724	**1.0000**	.9328	.9303
ecoli3	.9194	.8925	.9765	.8923	.9370	.9665	.8898	.9325	.9632	.8928	.9303	.9576	.8206	.9593	**1.0000**	.9076	.8998
ecoli4	.9699	.9482	.9803	.9687	.9731	.9768	.9658	.9723	.9741	.9633	.9715	.9674	.9832	.9799	**1.0000**	.9406	.9190
glass0	.7629	.7927	.9372	.7529	.8384	.8815	.7403	.8469	.9032	.7394	.8563	.9195	.2151	.9291	**.9991**	.8656	.8986
glass-0-1-2-3_vs_4-5-6	.9183	.9479	.9931	.9323	.9712	.9630	.9426	.9688	.9720	.9393	.9688	.9805	.8453	.9829	**1.0000**	.9570	.9593
glass-0-1-6_vs_2	.8347	.7616	.9562	.6589	.8775	.9631	.6721	.8742	.9562	.5932	.8783	.9827	.0000	.9587	**1.0000**	.7928	.7891
glass-0-1-6_vs_5	.9725	.9186	.9928	.9434	.9592	.9885	.9373	.9688	.9885	.9419	.9710	.9914	.3333	.9761	**1.0000**	.9411	.9411
glass1	.8192	.8531	.8950	.6335	.8229	.8541	.6800	.8621	.9047	.5882	.8422	.9068	.6309	.9174	**1.0000**	.7880	.9018
glass2	.7591	.7787	.9638	.6479	.8934	.9528	.6709	.8856	.9625	.6276	.8863	.9730	.6719	.9384	**1.0000**	.7902	.8076
glass4	.9615	.8970	.9812	.9483	.9574	.9812	.9457	.9600	.9767	.9437	.9587	.9060	.9607	.9793	**1.0000**	.9203	.9574
glass5	.9580	.9135	.9933	.9429	.9634	.9834	.9409	.9671	.9883	.9416	.9665	.9976	.9709	.9939	**1.0000**	.9474	.9474
glass6	.9582	.9527	.9925	.9518	.9670	.9615	.9471	.9774	.9842	.9451	.9781	.9864	.8808	.9810	**1.0000**	.9426	.9447
habermanImb	**.7831**	.6483	.7681	.6024	.6451	.7130	.5945	.7066	.7638	.6589	.7470	.5376	.2681	.7521	.7772	.7109	.7195
iris0	**1.0000**	.9600	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	.9899	**1.0000**	**1.0000**	**1.0000**
new-thyroid1	.9972	.9525	.9937	.9846	.9538	.9860	.9761	.9690	.9944	.9825	.9682	.9902	.9943	.9944	**1.0000**	.9605	.9726
new-thyroid2	.9837	.9645	.9930	.9832	.9596	.9830	.9811	.9618	.9902	.9804	.9639	.9902	.9972	.9844	**1.0000**	.9648	.9853
page-blocks0	.9061	.8910	.9829	.9242	.9451	.9717	.9194	.9485	.9706	.9143	.9462	**.9902**	.9246	.9822	.9868	.9719	.9688
page-blocks-1-3_vs_4	.9864	.9156	.9975	.9668	.9720	.9949	.9545	.9832	.9938	.9571	.9789	.9989	.8384	.9941	**1.0000**	.9691	.9677
pimaImb	.8128	.7745	.8588	.7531	.7357	.8056	.7507	.7768	.8368	.7441	.7604	.8537	.7334	.8237	**.9415**	.7866	.8624
segment0	.9288	.9326	.9974	.9958	.9928	.9976	.9945	.9951	.9969	.9961	.9950	.9988	.9990	.9984	**1.0000**	.9893	.9922
shuttle-c0-vs-c4	.9967	.9960	.9999	.9999	.9996	.9999	**1.0000**	**1.0000**	.9999	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
shuttle-c2-vs-c4	.9931	.9827	.9990	**1.0000**	.9959	**1.0000**	**1.0000**	.9969	**1.0000**	**1.0000**	.9969	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
vehicle0	.9111	.8273	.9843	.9709	.9223	.9721	.9639	.9315	.9731	.9606	.9387	.9860	.9781	.9863	**1.0000**	.9677	.9650
vehicle1	.8336	.7557	.9387	.8211	.7758	.8687	.8149	.7994	.8800	.8095	.7909	.9346	.7929	.9245	**1.0000**	.8618	.8682
vehicle2	.9525	.8683	.9926	.9679	.9125	.9835	.9583	.9280	.9851	.9623	.9230	.9866	.9734	.9928	**1.0000**	.9764	.9790
vehicle3	.7993	.7295	.9288	.8061	.7585	.8709	.7937	.7870	.8795	.7941	.7867	.9193	.8054	.9333	**1.0000**	.8794	.8756
vowel0	.9950	.9775	.9978	.9732	.9912	.9971	.9730	.9975	.9989	.9742	.9974	.9925	.8566	.9976	**1.0000**	.9688	.9728
wisconsinImb	.9704	.9494	.9836	.9781	.9769	.9784	.9799	.9817	.9780	.9800	.9804	.9779	.9724	.9797	**1.0000**	.9734	.9809
yeast-0-5-6-7-9_vs_4	.8733	.8654	.9544	.7960	.9212	.9355	.7917	.9163	.9332	.8022	.9086	.9737	.0000	.9595	**1.0000**	.8721	.8807
yeast1	.7028	.8040	.8048	.7106	.8004	.7798	.7088	.8068	.7894	.6830	.7815	.7650	.6619	.7661	**.8658**	.7889	.8014
yeast-1_vs_7	.7483	.8350	.9499	.7701	.8906	.8930	.7728	.8824	.9268	.7640	.8757	.9737	.0000	.9506	**1.0000**	.8253	.8155
yeast-1-2-8-9_vs_7	.8470	.8578	.9374	.7406	.9302	.9075	.7429	.9249	.9367	.7275	.9207	.9749	.0000	.9674	**1.0000**	.8060	.7918
yeast-1-4-5-8_vs_7	.6168	.7979	.9293	.6816	.9098	.9021	.6973	.8922	.9301	.6686	.8903	.9633	.0000	.9380	**1.0000**	.7607	.7388
yeast-2_vs_4	.9479	.9422	.9850	.9102	.9499	.9720	.9108	.9465	.9813	.9068	.9445	.9795	.0000	.9726	**1.0000**	.9564	.9467
yeast-2_vs_8	.9155	.9662	.9709	.8051	.9563	.9783	.8055	.9506	.9753	.8031	.9532	.9927	.8066	.9796	**1.0000**	.8567	.8642
yeast3	.9262	.8843	.9611	.9122	.9252	.9407	.9092	.9283	.9651	.9177	.9237	.9782	.9057	.9673	**1.0000**	.9486	.9414
yeast4	.8940	.9096	.9716	.8607	.9546	.9466	.8586	.9501	.9635	.8570	.9465	.9718	.8603	.9759	**1.0000**	.8863	.8674
yeast5	.9570	.9459	.9902	.9628	.9815	.9843	.9624	.9805	.9868	.9622	.9796	.9929	.9642	.9887	**1.0000**	.9611	.9589
yeast6	.8846	.9170	.9863	.8861	.9700	.9837	.8867	.9658	.9829	.8884	.9656	.9882	.8806	.9852	**1.0000**	.9086	.9079
Avg.	.8945	.8856	.9615	.8731	.9223	.9452	.8729	.9274	.9542	.8680	.9262	.9573	.7026	.9587	**.9902**	.9002	.9064

Table 16
Results of GM for binary class data sets (testing).

Data set	IDGC	SMT DGC	SMT C4.5	SMT SVM	SMT KNN	SMT ENN C4.5	SMT ENN SVM	SMT ENN KNN	SMT TL C4.5	SMT TL SVM	SMT TL KNN	C4.5 CS	SVM CS	SMT BG	SMT BST	BC	EE
abalone19	.6958	.7304	.2993	**.7680**	.1734	.4230	.7670	.3821	.4224	.7659	.3811	.4241	.7626	.3855	.3045	.6904	.6845
abalone9-18	.8248	.8286	.7536	.8579	.7307	.7663	.8609	.7534	.6895	.8534	.7589	.6121	**.8763**	.7891	.7552	.7208	.7674
ecoli-0_vs_1	.9835	**.9869**	.9835	.9800	.9630	.9835	.9835	.9800	.9765	.9765	.9695	.9835	.9670	.9835	.9765	.9800	.9695
ecoli-0-1-3-7_vs_2-6	.8931	.8153	.7364	.8683	.8327	.8185	.8073	**.9053**	.8169	.8772	.9036	.8264	.8452	.8296	.8327	.7364	.7910
ecoli1	.8953	.8994	.9043	.9034	.8666	.8836	.8953	.8847	.8803	.9013	.8798	**.9108**	.9034	.9055	.8620	.8831	.8823
ecoli2	.9071	.9216	.8969	.9126	.9171	.8852	.9089	**.9368**	.8965	.9089	.9263	.8884	.0000	.8852	.8916	.8999	.8798
ecoli3	.8697	.8697	.8518	.8867	.8208	.8349	**.8920**	.8424	.8487	.8832	.8345	.8303	.7885	.8720	.8516	.8679	.8832
ecoli4	.9646	.9629	.9213	**.9662**	.9607	.9166	.9433	.9290	.8396	.9385	.9290	.8613	.9528	.8758	.8845	.8802	.8802
glass0	.7416	.7618	.8169	.7289	.8097	.7914	.7416	**.8404**	.7998	.7338	.8216	.8196	.1679	.8133	.8292	.7547	.8233
glass-0-1-2-3_vs_4-5-6	.9301	.8608	.9058	.9171	.9391	.9178	.8951	**.9497**	.9111	.9178	.9465	.8779	.8324	.9301	.9160	.9111	.8988
glass-0-1-6_vs_2	.7129	.6676	**.7648**	.5855	.6223	.4937	.5237	.5780	.6691	.5358	.6179	.5186	.0000	.6309	.5645	.6860	.7029
glass-0-1-6_vs_5	.9472	.9411	.9681	.9411	.8563	.9127	.9381	.9182	.9182	.9411	.9155	**.9885**	.0000	.8511	.8692	.9411	.9411
glass1	.7765	.7872	.7340	.6196	.7933	.7209	.6458	.7871	.7689	.5202	.7803	.7144	.6005	.7575	**.8352**	.6908	.7523
glass2	.6944	.6903	.7339	.5778	.7146	.7450	.5865	.7494	.7146	.5948	.7104	.6031	.5679	**.8075**	.7229	.6873	.6733
glass4	.9439	**.9620**	.8825	.9117	.8896	.9292	.9142	.8849	.9267	.9067	.8849	.8518	.9242	.8777	.9267	.8810	.9167
glass5	.9576	.9344	.8490	.9344	.9148	.9077	.9370	.9148	.8490	.9396	.8535	.9359	.9728	.9195	**.9877**	.9474	.9474
glass6	**.9262**	.9023	.8662	.9235	.8711	.9103	.9056	.8923	.8973	.9235	.8833	.8848	.8615	.9005	.8760	.9023	.8942
habermanImb	.6415	.6053	.6351	.5666	.5349	.6344	.6088	.5944	.6102	.6113	.5957	.5274	.3115	**.6532**	.6492	.6091	.6394
iris0	**1.0000**	**1.0000**	.9899	**1.0000**	**1.0000**	.9899	**1.0000**	**1.0000**	.9899	**1.0000**	**1.0000**	.9899	**1.0000**	.9798	.9899	.9899	.9899
new-thyroid1	**.9888**	**.9888**	.9690	.9775	.9718	.9353	.9747	.9550	.9718	.9832	.9775	.9746	.9683	**.9888**	.9860	.9408	.9522
new-thyroid2	**.9916**	.9690	.9347	.9747	.9603	.9690	.9690	.9690	.9437	.9603	.9603	.9801	.9829	.9690	.9832	.9379	.9522
page-blocks0	.8654	.8685	.9385	.9217	.9124	.9385	.9192	.9263	.9387	.9155	.9139	.9455	.9252	.9567	.9360	.9539	**.9608**
page-blocks-1-3_vs_4	.9887	.9703	.9807	.9656	.9830	.9641	.9515	.9853	.9772	.9309	.9807	.9809	.8423	.9830	**.9955**	.9633	.9703
pimaImb	.7311	.7182	.7119	.7378	.6816	.7098	.7395	.7051	.6903	.7350	.7020	.7122	.7240	**.7658**	.7369	.6930	.7119
segment0	.9896	.9840	.9911	**.9972**	.9934	.9906	.9929	.9934	.9914	.9939	.9937	.9919	.9965	.9937	.9947	.9810	.9881
shuttle-c0-vs-c4	.9997	.9959	.9997	.9991	.9953	**1.0000**	.9959	.9959	.9997	.9994	.9959	.9997	**1.0000**	.9997	**1.0000**	**1.0000**	**1.0000**
shuttle-c2-vs-c4	.9959	.9959	.9877	.9836	.9959	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	.9959	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
vehicle0	.8928	.8654	.9322	.9605	.9235	.9300	.9613	.9175	.9276	.9581	.9268	.9288	.9492	.9606	**.9681**	.9475	.9381
vehicle1	.7215	.7046	.7049	.8019	.6666	.7193	**.8034**	.7193	.7408	.7968	.6973	.6998	.7543	.7507	.7277	.7224	.7237
vehicle2	.9320	.9127	.9452	.9526	.9307	.9395	.9505	.9317	.9505	.9449	.9309	.9432	.9572	.9668	**.9858**	.9513	.9540
vehicle3	.7001	.6945	.6990	.7871	.6750	.7038	.7888	.6867	.7431	.7792	.7159	.7278	**.7889**	.7508	.7133	.7412	.7530
vowel0	.9888	.9803	.9671	.9658	.9955	.9916	.9718	**.9994**	.9850	.9681	**.9994**	.9417	.8334	.9816	.9911	.9422	.9438
wisconsinImb	.9667	.9486	.9417	.9686	.9613	.9469	.9743	.9704	.9626	.9711	**.9788**	.9635	.9718	.9633	.9652	.9580	.9596
yeast-0-5-6-7-9_vs_4	**.8431**	.8387	.7727	.7867	.7847	.7433	.7907	.8176	.7485	.7925	.8097	.7027	.0000	.8051	.7956	.7906	.7723
yeast1	.7017	**.7257**	.7170	.7006	.6479	.7127	.7126	.6857	.6769	.6820	.6790	.6670	.6691	.7103	.7080	.6938	.7019
yeast-1_vs_7	.7574	.7041	.6504	.7480	.7194	.7135	.7603	.6736	.6867	**.7628**	.7354	.5276	.0000	.6886	.6061	.6914	.7139
yeast-1-2-8-9_vs_7	.6958	.6998	.5214	.6957	.4904	.5220	.6984	.6118	.4856	**.7019**	.5896	.6315	.0000	.5789	.5701	.6349	.6627
yeast-1-4-5-8_vs_7	.6009	**.7237**	.4859	.6350	.4884	.4165	.6444	.6470	.4376	.6284	.6676	.4261	.0000	.5634	.4027	.5721	.6143
yeast-2_vs_4	.9269	.8816	.8993	.8880	.8910	.8691	.8900	.8651	.9316	.8828	.8705	.8870	.0000	.9056	.8857	**.9397**	.9364
yeast-2_vs_8	.7987	.7352	.7904	.7335	.8156	.7877	.7335	.8072	**.8659**	.7311	.8335	.8575	.7352	.7611	.7344	.7363	.7666
yeast3	.9114	.9088	.8999	.9040	.8677	.9078	.9052	.8601	.9146	.9087	.8759	.9114	.8952	**.9398**	.8803	.9314	.9193
yeast4	**.8445**	.8431	.7039	.8118	.7210	.7747	.8242	.7729	.7694	.8195	.7933	.6892	.8144	.7697	.6445	.7944	.8249
yeast5	.9567	.9534	.9487	.9614	.9525	.9511	.9618	.9498	**.9709**	.9614	.9609	.9319	.9650	.9633	.9077	.9515	.9468
yeast6	**.8916**	.8821	.8278	.8866	.8429	.8212	.8722	.8684	.8399	.8712	.8681	.7940	.8756	.8547	.7849	.8717	.8441
Avg.	**.8633**	.8550	.8276	.8544	.8200	.8278	.8532	.8418	.8313	.8502	.8419	.8151	.6814	.8459	.8279	.8409	.8506

Table 17
Results of IDGC and DGC+.

Data set	AUC				GM				time (s)	
	IDGC		DGC+		IDGC		DGC+		IDGC	DGC+
	trn	tst	trn	tst	trn	tst	trn	tst		
abalone19	**0.7975**	**0.6970**	0.7643	0.7545	**0.7957**	**0.6958**	0.7646	0.7494	**895.23**	44472.00
abalone9-18	**0.8564**	**0.8249**	0.5980	0.5900	**0.8562**	**0.8248**	0.5996	0.4337	**113.27**	536.53
ecoli-0_vs_1	**0.9227**	**0.8939**	0.5269	0.5000	**0.9222**	**0.8931**	0.5264	0.0000	36.41	**4.23**
ecoli-0-1-3-7_vs_2-6	**0.9369**	**0.9835**	0.7638	0.7565	**0.9369**	**0.9835**	0.7632	0.7553	44.91	**35.73**
ecoli1	**0.9108**	**0.8975**	0.8042	0.7875	**0.9086**	**0.8953**	0.8044	0.7844	**19.36**	87.10
ecoli2	**0.9453**	**0.9078**	0.8803	0.8685	**0.9451**	**0.9071**	0.8804	0.8635	78.21	**44.55**
ecoli3	**0.9212**	**0.8708**	0.7921	0.7835	**0.9194**	**0.8697**	0.7928	0.7822	73.35	**73.31**
ecoli4	**0.9700**	**0.9652**	0.8526	0.8375	**0.9699**	**0.9646**	0.8525	0.8352	**69.78**	74.89
glass0	**0.9183**	**0.9301**	0.8318	0.8235	**0.9183**	**0.9301**	0.8325	0.8232	**55.60**	85.84
glass-0-1-2-3_vs_4-5-6	0.8462	0.7203	**0.9134**	**0.9040**	0.8347	0.7129	**0.9129**	**0.9037**	**51.72**	61.93
glass-0-1-6_vs_2	**0.9729**	**0.9486**	0.7570	0.6995	**0.9725**	**0.9472**	0.7569	0.6906	**42.51**	71.36
glass-0-1-6_vs_5	0.7719	0.7556	**0.9081**	**0.8805**	0.7629	0.7416	**0.9080**	**0.8745**	**40.24**	42.65
glass1	**0.8224**	**0.7843**	0.7960	0.7530	**0.8192**	**0.7765**	0.7955	0.7521	**53.13**	60.89
glass2	**0.7793**	**0.7411**	0.6556	0.6545	**0.7591**	**0.6944**	0.6572	0.6282	**61.41**	146.12
glass4	**0.9615**	**0.9441**	0.8615	0.8545	**0.9615**	**0.9439**	0.8626	0.8502	**48.87**	50.07
glass5	**0.9581**	**0.9585**	0.9400	0.9370	**0.9580**	**0.9576**	0.9408	0.9358	45.86	**41.27**
glass6	**0.9583**	**0.9267**	0.9045	0.9005	**0.9582**	**0.9262**	0.9056	0.8976	45.37	**30.87**
haberman	**0.7836**	**0.6420**	0.6752	0.6145	**0.7831**	**0.6415**	0.6767	0.6141	47.14	**23.20**
iris0	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	2.79	**0.08**
new-thyroid1	0.9972	0.9889	**1.0000**	**0.9970**	0.9972	0.9888	**1.0000**	**0.9970**	22.94	**0.73**
new-thyroid2	**0.9837**	**0.9917**	0.9702	0.9685	**0.9837**	**0.991**6	0.9692	0.9682	20.41	**0.71**
page-blocks0	**0.9865**	**0.9887**	0.9500	0.9455	**0.9864**	**0.9887**	0.9505	0.9455	**1168.63**	145217.76
page-blocks-1-3_vs_4	0.9069	0.8688	**0.9987**	**0.9920**	0.9061	0.8654	**0.9987**	**0.9920**	122.27	**4.32**
pima	**0.8129**	**0.7316**	0.7619	0.7245	**0.8128**	**0.7311**	0.7621	0.7243	**748.54**	1068.19
segment0	**0.9289**	**0.9896**	0.5263	0.5000	**0.9288**	**0.9896**	0.5260	0.0000	**1294.40**	3519.32
shuttle-c0-vs-c4	**0.9967**	**0.9997**	0.9957	0.9955	**0.9967**	**0.9997**	0.9945	0.9955	117.02	**9.95**
shuttle-c2-vs-c4	**0.9934**	**0.9959**	0.9151	0.9125	**0.9931**	**0.9959**	0.9156	0.9090	0.81	**0.08**
vehicle0	0.9146	0.8976	**0.9768**	**0.9500**	0.9111	0.8928	**0.9768**	**0.9499**	**1012.07**	4266.52
vehicle1	**0.8382**	0.7227	0.7704	**0.7650**	**0.8336**	0.7215	0.7700	**0.7623**	**1175.07**	6196.01
vehicle2	0.9529	0.9324	**0.9809**	**0.9615**	0.9525	0.9320	**0.9808**	**0.9615**	**965.82**	4847.35
vehicle3	**0.8046**	0.7015	0.7958	**0.7480**	**0.7993**	0.7001	0.7961	**0.7405**	**1185.15**	5878.44
vowel0	0.9950	0.9889	**0.9991**	**0.9940**	0.9950	0.9888	**0.9980**	**0.9940**	1081.51	**22.56**
wisconsin	**0.9705**	**0.9667**	0.9672	0.9665	**0.9704**	**0.9667**	0.9672	0.9665	842.12	**271.24**
yeast-0-5-6-7-9_vs_4	**0.8757**	**0.8433**	0.7963	0.7465	**0.8733**	**0.8431**	0.7962	0.7369	**100.61**	400.60
yeast1	0.7030	0.7019	**0.7369**	**0.7270**	0.7028	0.7017	**0.7367**	**0.7268**	**603.61**	4685.42
yeast-1_vs_7	**0.7562**	**0.7575**	0.7101	0.7045	**0.7483**	**0.7574**	0.7108	0.6833	**148.10**	339.02
yeast-1-2-8-9_vs_7	**0.8491**	0.6989	0.7267	**0.7045**	**0.8470**	**0.6958**	0.7282	0.6833	**237.29**	385.13
yeast-1-4-5-8_vs_7	0.6252	0.6009	**0.7074**	**0.6750**	0.6168	0.6009	**0.7088**	**0.6708**	**169.68**	588.26
yeast-2_vs_4	**0.9480**	**0.9272**	0.8355	0.8050	**0.9479**	**0.9269**	0.8356	0.8041	**86.91**	1344.39
yeast-2_vs_8	**0.9156**	**0.8056**	0.7845	0.7665	**0.9155**	**0.7987**	0.7850	0.7353	**72.94**	776.26
yeast3	0.9275	0.9123	**0.9367**	**0.9210**	0.9262	0.9114	**0.9357**	**0.9210**	**527.03**	3145.30
yeast4	**0.8950**	**0.8445**	0.7913	0.7905	**0.8940**	**0.8445**	0.7924	0.7878	**520.33**	10799.93
yeast5	0.9579	0.9576	**0.9685**	**0.9635**	0.9570	0.9567	**0.9698**	**0.9635**	**516.49**	3205.04
yeast6	0.8847	**0.8919**	**0.9013**	0.8795	0.8846	**0.8916**	**0.9007**	0.8759	**479.28**	7428.51
Avg.	**0.8967**	**0.8659**	0.8347	0.8183	**0.8951**	**0.8633**	0.8350	0.7879	**341.91**	5689.63

Table 18
Results for multiclass data sets (training).

Data set	AUC					GM				
	IDGC OVA	IDGC OVO	C4.5	KNN	SVM	IDGC OVA	IDGC OVO	C4.5	KNN	SVM
autos	.8001	.8194	**.9290**	.6825	.8067	.7974	.8152	**.9288**	.6768	.8099
balance	.8037	.8395	.8095	.6667	**.8794**	.8037	.8394	.7793	.5192	**.8784**
contraceptive	.6691	.6636	**.7254**	.4257	.5412	.6667	.6602	**.7235**	.4244	.5418
dermatology	.9136	**.9727**	.9693	.9551	.9587	.9133	**.9725**	.9692	.9545	.9629
ecoli	.7370	.7868	**.8118**	.6742	.7079	.7156	**.7674**	.7458	.5710	.7035
glass	.6942	.7323	**.9125**	.6652	.7380	.6898	.7244	**.9118**	.6536	.7381
hayes-roth	.8189	.8325	**.8921**	.3647	.8013	.8166	.8295	**.8902**	.3205	.7996
lymphography	.8341	.8935	**.9176**	.6275	.8415	.8306	.8922	**.9173**	.5303	.8448
new-thyroid	.9340	**.9823**	.9768	.9510	.9645	.9340	**.9823**	.9767	.9508	.9651
pageblocks	.8568	.8865	**.9633**	.7023	.5842	.8560	.8858	**.9626**	.6441	.5828

Table 18 (*continued*)

Data set	AUC					GM				
	IDGC OVA	IDGC OVO	C4.5	KNN	SVM	IDGC OVA	IDGC OVO	C4.5	KNN	SVM
penbased	.8876	.9441	**.9782**	.9738	.9459	.8871	.9435	**.9782**	.9737	.9485
shuttle	.8206	.8576	.9057	**.9075**	.8113	.8196	.8563	.8861	**.8908**	.8093
thyroid	.6824	.6850	**.9925**	.5529	.5136	.6763	.6767	**.9925**	.3934	.5100
wine	.9389	.9887	**.9900**	.9610	.6325	.9389	.9886	**.9900**	.9602	.6311
yeast	.6587	.6742	**.7813**	.5164	.7572	.6543	.6682	**.7699**	.4991	.7586
Avg.	.8033	.8372	**.9037**	.7084	.7656	.8000	.8335	**.8948**	.6642	.7656

Table 19
Results for multiclass data sets (testing).

Data set	AUC					GM				
	IDGC OVA	IDGC OVO	C4.5	KNN	SVM	IDGC OVA	IDGC OVO	C4.5	KNN	SVM
autos	.7600	.7289	.7679	.7412	**.7691**	.7569	.7255	.7616	.7388	**.7688**
balance	.7547	**.7752**	.6581	.6677	.6637	.7532	**.7731**	.5167	.5211	.6636
contraceptive	.5791	**.6566**	.5093	.4227	.5213	.5753	**.6560**	.5063	.4203	.5276
dermatology	.8965	**.9607**	.9137	.9604	.9327	.8955	**.9603**	.9128	.9599	.9314
ecoli	.7230	**.7281**	.6582	.6882	.7111	.6684	.6720	.5541	.5839	**.7111**
glass	.6302	.6752	.6738	**.7141**	.7009	.6227	.6668	.6652	.7039	**.7100**
hayes-roth	.8248	**.8651**	.8469	.3577	.7624	.8228	**.8634**	.8436	.3109	.7583
lymphography	**.7810**	.7668	.7164	.6479	.7414	**.7727**	.7576	.7126	.5520	.7401
new-thyroid	.9306	.9558	.8986	**.9649**	.9354	.9303	.9555	.8983	**.9648**	.9339
pageblocks	.7569	**.8518**	.8065	.7218	.6011	.7527	**.8489**	.7772	.6738	.6119
penbased	.8511	.9065	.8934	**.9710**	.9039	.8504	.9052	.8930	**.9709**	.9088
shuttle	.8537	.8760	.7958	**.9471**	.7767	.8519	.8729	.5963	**.9385**	.5789
thyroid	.6215	.6299	**.9739**	.5723	.5188	.6152	.6245	**.9738**	.4340	.5179
wine	.9535	**.9950**	.9493	.9598	.6350	.9529	**.9950**	.9492	.9588	.6218
yeast	.6325	.6380	.5492	.5041	**.6901**	.6258	**.6283**	.5114	.4875	.6118
Avg.	.7699	**.8006**	.7741	.7227	.7242	.7631	**.7937**	.7381	.6813	.7064

Appendix B. Source code

The source code of IDGC, written in C/C++, is available at https://sourceforge.net/projects/dgc-cn/files/. We provide the whole MS-VS2010 project. The code also can been compiled using g++ on Linux by changing the OS_TYPE macro in DEF.h.

References

[1] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Multi-Valued Logic Soft Comput. 17 (2011) 255–287.
[2] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Comput. 13 (2009) 307–318.
[3] R. Barandela, J.S. Sánchez, V. García, E. Rangel, Strategies for learning in class imbalance problems, Pattern Recogn. 36 (2003) 849–851.
[4] G.E.A.P.A. Batista, R.C. Prati, M.C. Monard, A study of the behaviour of several methods for balancing machine learning training data, SIGKDD Explor. 6 (2004) 20–29.
[5] T. Basu, C.A. Murthy, Towards enriching the quality of k-nearest neighbor rule for document classification, Int. J. Mach. Learn. Cybern. (2013) 1–9.
[6] A.P. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recogn. 30 (1997) 1145–1159.
[7] I. Brown, C. Mues, An experimental comparison of classification algorithms for imbalanced credit scoring data sets, Expert Syst. Appl. 39 (2012) 3446–3453.
[8] A. Cano, A. Zafra, S. Ventura, Weighted data gravitation classification for standard and imbalanced data, IEEE Trans. Cybern. 43 (6) (2013) 1672–1687.
[9] I. Chairi, S. Alaoui, A. Lyhyaoui, Intrusion detection based sample selection for imbalanced data distribution, in: Proceeding of 2012 Second International Conference on Innovative Computing Technology (INTECH), 2012, pp. 259–264.
[10] P.K. Chan, S.J. Stolfo, Toward scalable learning with non-uniform class and cost distributions: a case study in credit card fraud detection, in: Proceedings of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1998, pp. 164–168.
[11] N.V. Chawla, K. Bowyer, L. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.
[12] N.V. Chawla, A. Lazarevic, L.O. Hall, et al., SMOTEBoost: improving prediction of the minority class in boosting, in: Proceedings of the 7th European Conference on Principles of Data Mining and Knowledge Discovery, 2003, pp. 107–119.
[13] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Trans. Inform. Theory 13 (1) (1967) 21–27.
[14] V. Dheepa, R. Dhanapal, G. Manjunath, Fraud detection in imbalanced datasets using cost based learning, Eur. J. Sci. Res. 91 (2012) 486–490.
[15] A. Dhurandhar, A. Dobra, Probabilistic characterization of nearest neighbor classifier, Int. J. Mach. Learn. Cybern. 4 (4) (2013) 259–272.
[16] P. Domingos, MetaCost: a general method for making classifiers cost-sensitive, in: Proceedings of the 5th ACM SIGKDD International Conference of Knowledge Discovery and Data Mining, SanDiego, CA, USA 1999, pp. 155–164.
[17] R.C. Eberhart, Y. Shi, Comparison between genetic algorithms and particle swarm optimization, Evolutionary Programming VII, Springer, Berlin Heidelberg, 1998. pp. 611–616.

[18] C. Elkan, The foundations of cost-sensitive learning, in: Proceedings of the 17th IEEE International Joint Conference on Artificial Intelligence (IJCAI01), 2001, pp. 973–978.
[19] A. Fernández, M.J. del Jesus, F. Herrera, On the 2-tuples based genetic tuning performance for fuzzy rule based classification systems in imbalanced data-sets, Inform. Sci. 180 (2010) 1268–1291.
[20] A. Fernández, V. López, M. Galar, et al, Analysing the classification of imbalanced data-sets with multiple classes: binarization techniques and ad-hoc approaches, Knowl.-Based Syst. 42 (2013) 97–110.
[21] M. Galar, A. Fernández A, E. Barrenechea, et al, An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes, Pattern Recogn. 44 (8) (2011) 1761–1776.
[22] M. Galar, A. Fernández A, E. Barrenechea, et al, A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches, IEEE Tran. Syst. Man Cybern. Part C: Appl. Rev. 42 (4) (2012) 463–484.
[23] S. García, A. Fernández, J. Luengo, F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, Soft Comput. 13 (2009) 959–977.
[24] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inform. Sci. 180 (10) (2010) 2044–2064.
[25] V. García, R. Mollineda, J.S. Sánchez, On the k-NN performance in a challenging scenario of imbalance and overlapping, Pattern Anal. Appl. 11 (2008) 269–280.
[26] A. Ghazikhani, R. Monsefi, H.S. Yazdi, Online neural network model for non-stationary and imbalanced data stream classification, Int. J. Mach. Learn. Cybern. 5 (1) (2014) 51–62.
[27] J.A. Hartigan, M.A. Wong, A K-means clustering algorithm, Appl. Statist. 28 (1979) 100–108.
[28] T. Hastie, R. Tibshirani, Classification by pairwise coupling, Ann. Statist 26 (2) (1998) 451–471.
[29] R. Hassan, B. Cohanim, O. Weck, et al, A comparison of particle swarm optimization and the genetic algorithm, in: Proceedings of the 1st AIAA Multidisciplinary Design Optimization Specialist Conference, 2005.
[30] H. He, E.A. García, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (2009) 1263–1284.
[31] H. He, Y. Ma, Imbalanced Learning: Foundations, Algorithms, and Applications, Wiley.com, 2013.
[32] J. Huang, C.X. Ling, Using AUC and accuracy in evaluating learning algorithms, IEEE Trans. Knowl. Data Eng. 17 (2005) 299–310.
[33] S. Hido, H. Kashima, Y. Takahashi, Roughly balanced bagging for imbalanced data, Stat. Anal. Data Min 2 (2009) 412–426.
[34] N. Japkowicz, S. Stephen, The class imbalance problem: a systematic study, Intell. Data Anal. J. 6 (2002) 429–450.
[35] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[36] J. Kennedy, SmallWorlds and mega-minds: effects of neighborhood topology on particle swarm performance, in: Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, 1999, pp. 1931–1938.
[37] T.M. Khoshgoftaar, J.V. Hulse, A. Napolitano, Comparing boosting and bagging techniques with noisy and imbalanced data, IEEETrans. Syst. Man Cybern. Part B Cybern. 41 (3) (2011) 552–568.
[38] T. Krink, J. Vesterstrom, J. Riget. Particle swarm optimization with spatial particle extension, in: Proceedings of the Congress on Evolutionary Computation, 2002.
[39] M. Kubat, S. Matwin, Addressing the curse of imbalanced training sets: onesided selection, in: 14th International Conference on Machine Learning(ICML97), 1997, pp. 179–186.
[40] X.Y. Liu, J. Wu, Z.H. Zhou, Exploratory undersampling for classimbalance learning, IEEE Trans. Syst. Man Cybern. Part B Cybern. 39 (2009) 539–550.
[41] V. López, A. Fernández, F. Herrera, On the importance of the validation technique for classification with imbalanced datasets: addressing covariate shift when data is skewed, Inform. Sci. 257 (2014) 1–13.
[42] V. López, A. Fernández, S. García, et al, An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics, Inform. Sci. 250 (20) (2013) 113–141.
[43] V. López, A. Fernández, J.G. Moreno-Torres, F. Herrera, Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics, Expert Syst. Appl. 39 (7) (2012) 6585–6608.
[44] M.A. Mazurowski, P.A. Habas, J.M. Zurada, et al, Training neural network classifiers for medical decision making: the effects of imbalanced datasets on classification performance, Neural Networks: Off. J. Int. Neural Network Soc. 21 (2008) 427.
[45] K. Napierala, J. Stefanowski, S. Wilk, Learning from imbalanced data in presence of noisy and borderline examples, in: Proceedings of 7th International Conference on Rough Sets and Current Trends in Computing (RSCTC2010), 2010, pp. 158–167.
[46] A. Orriols-Puig, E. Bernad-Mansilla, Evolutionary rule-based systems for imbalanced datasets, Soft Comput. 13 (2009) 213C225.
[47] S. Panda, N.P. Padhy, Comparison of particle swarm optimization and genetic algorithm for FACTS-based controller design, Appl. Soft Comput. 8 (4) (2008) 1418–1427.
[48] S. Parsazad, H.S. Yazdi, S. Effati, Gravitation based classification, Inform. Sci. 220 (2013) 319–330.
[49] L. Peng, B. Yang, Y. Chen, A. Abraham, Data gravitation based classification, Inform. Sci. 179 (2009) 809–819.
[50] L. Peng, H. Zhang, B. Yang, Y. Chen, M.T. Qassrawi, G. Lu, Traffic identification using flexible neural trees, in: Proceeding of the 18th International Workshop of QoS (IWQoS 2012), 2012, pp. 1–5.
[51] J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kauffman, 1993.
[52] R. Rifkin, A. Klautau, In defense of one-vs-all classification, J. Mach. Learn. Res. 5 (2004) 101–141.
[53] C. Seiffert, T.M. Khoshgoftaar, J. Van Hulse, et al. An empirical study of the classification performance of learners on imbalanced and noisy software quality data, in: Procedings of IEEE International Conference on Information Reuse and Integration, 2007, pp. 651–658.
[54] C. Seiffert, T.M. Khoshgoftaar, J. Van Hulse, A. Napolitano, Rusboost: a hybrid approach to alleviating class imbalance, IEEE Trans. Syst. Man Cybern. Part A 40 (2010) 185–197.
[55] D. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, 2nd ed., Chapman & Hall, CRC, 2006.
[56] D. Simić, I. Tanackov, V. Gajić, et al, Financial Forecasting of Invoicing and Cash Inflow Processes for Fair Exhibitions, Hybrid Artificial Intelligence Systems, Springer, Berlin Heidelberg, 2009. pp. 686–693.
[57] Y. Sun, A.K.C. Wong, M.S. Kamel, Classification of imbalanced data: a review, Int. J. Pattern Recogn. Artif. Intell. 23 (2009) 687–719.
[58] K.M. Ting, An instance-weighting method to induce cost-sensitive trees, IEEE Trans. Knowl. Data Eng. 14 (2002) 659–665.
[59] K. Veropoulos, C. Campbell, N. Cristianini, Controlling the sensitivity of support vector machines, in: Proceedings of the International Joint Conference on AI, 1999, pp. 55–60.
[60] V. Vapnik, Statistical Learning Theory, WileyPublishers, NewYork, USA, 1998.
[61] S. Wang, X. Yao, Diversity analysis on imbalanced data sets by using ensemble models, in: Proceedings of IEEE Symposium Series on Computational Intelligence and Data Mining (IEEE CIDM 2009), 2009, pp. 324–331.
[62] Weka 3: Data Mining Software in Java. <http://www.cs.waikato.ac.nz/ml/weka/>.
[63] G.M. Weiss, Mining with rarity: a unifying framework, SIGKDD Explor. 6 (2004) 7–19.
[64] G. Wen, J. Wei, J. Wang, et al, Cognitive gravitation model for classification on small noisy data, Neurocomputing 118 (2013) 245–252.
[65] D. Yeung, X. Wang, Improving performance of similarity-based clustering by feature weight learning, IEEE Trans. Pattern Anal. Mach. Intell. 24 (4) (2002) 556–561.
[66] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, Y. Nakanishi, A particle swarm optimization for reactive power and voltage control considering voltage security assessment, IIEEE Trans. Power Syst. 15 (2000) 1232–1239.
[67] H. Yu, J. Ni, J. Zhao, ACOSampling: an ant colony optimization-based undersampling method for classifying imbalanced DNA microarray data, Neurocomputing 101 (2013) 309–318.

[68] B. Zadrozny, C. Elkan, Learning and making decisions when costs and probabilities are both unknown, in: Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining (KDD01), 2001, pp. 204–213.
[69] B. Zadrozny, J. Langford, N. Abe, Cost-sensitive learning by cost-proportionate example weighting, in: Proceedings of the 3rd International Conference of Data Mining, Melbourne, Florida, USA, 2003, pp. 435–442.
[70] H. Zhang, G. Lu, M.T. Qassrawi, Y. Zhang, X. Yu, Feature selection for optimizing traffic classification, Comput. Commun. 35 (2012) 1457–1471.
[71] Z.H. Zhou, X.Y. Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, IEEE Trans. Knowl. Data Eng. 18 (2006) 63–77.