A Density-based Under-sampling Algorithm for Imbalance Classification

Yun Hou¹, Bailin Li¹, Li Li¹ and Jiajia Liu²
¹Department of Mechanical Engineering, Southwest Jiaotong University, Chengdu, Sichuan 610031, China
²Department of Mechanical Engineering, Chengdu University of Information Technology, Chengdu, Sichuan 610031, China
Email: houyun87@163.com
Email: bli62@263.net
Email: lili@swjtu.edu.cn
Email: liujj@cuit.edu.cn

Abstract. Imbalance classification is an interesting issue in machine learning and data mining. In recent years, many related algorithms have been proposed to solve such an issue. Among them, under-sampling is an effective and timesaving data pre-processing method, which balances the dataset by removing some examples from the majority class. However, these proposed under-sampling methods often lose some useful information or ignore noise in the datasets, which will result in the performance degradation. This paper proposes a density-based under-sampling algorithm (DBU) to solve these two problems. In feature space, similar examples are close to each other and noisy example is far from other examples belonging to the same class. Thus the similar examples have a high density while the noisy example has a low density. DBU uses the local density peaks to represent the whole majority class, so that it can retain the useful information and eliminate the noisy examples automatically. To evaluate our algorithm, experiments are conducted on 15 two-class imbalanced datasets. Experimental results show that DBU achieves the better results than other under-sampling methods.

1. Introduction
Imbalance classification has attracted many researchers’ attentions in data mining and machine learning [1, 2]. Focusing on two-class problem, a dataset is imbalanced when the number of examples in one class (minority class) is less than the other (majority class) [3]. Many learning algorithms proposed for balanced data always have difficulties to represent the distribution of imbalanced data properly, thus they perform poorly for the imbalanced data. In other words, because the data distribution of imbalanced dataset is skewed, these standard classifiers always classify the minority class examples into majority class [4]. Actually, the minority class is often more interesting and important than the majority class [5].

Many algorithms have been proposed to alleviate the class imbalance problem. These algorithms can be mainly categorized into two groups: algorithmic-level algorithms and data-level algorithms. Algorithmic-level algorithms modify the traditional classifiers for balanced dataset, such that the modified algorithms (SVDD [6], Cost-sensitive methods [7], and Boosting SVM [8]) can be applied to imbalanced dataset. Data-level algorithms oversample the minority class [9] or under-sample the majority class [10] to build a balanced dataset before constructing the classifiers. Compared with algorithmic-level algorithms, data-level algorithms are widely used in imbalance classification because they can be independently performed before the classifier training tasks. In these two
sampling strategies, the oversampling methods, which address the imbalance problem by generating synthetic minority class examples, may lead to the over-fitting in the model construction process [11]. As a result, some researchers believe that the under-sampling strategy is superior to the oversampling strategy [12]. However, the under-sampling strategy usually loses some useful information about the majority class [19]. Real-world dataset always contains many similar examples. For the similar examples, one under-sampling strategy is to use one of these similar examples to represent them all while the remaining examples are treated as redundant information. Lin et al. [19] propose a clustering-based under-sampling algorithm (called CBU for short) to retain the useful information. CBU uses the k-means clustering algorithm to cluster the similar examples together and combines the clustering centers with the minority class examples to build a balanced dataset. However, the k-means clustering algorithm is sensitive to initial condition and is not robust to noise. Actually, noise is another important factor affecting the accuracy of classification. In this paper, we propose a density-based under-sampling algorithm to solve these problems (called DBU for short). Our DBU has its basis in the assumptions that similar examples are relatively close to each other and noise is far from other examples belonging to the same class in feature space. Thus, similar examples have a relative high density and noise has a relative low density. We use the local density peaks to represent the whole majority class, so that it can retain the useful information as much as possible and remove the noise at the same time. Firstly, we estimate the approximately probabilistic distribution of majority class examples by kernel density estimation. Then, we introduce a new evaluation metric to find the local density peaks and remove the noisy examples automatically. Finally, we combine these local density peaks with the original minority class examples to build a balanced dataset.

This paper is organized as follows. Section 2 recalls the problems of imbalance classification and some under-sampling algorithms. Section 3 describes our DBU in detail. Section 4 compares our DBU with other under-sampling methods, and Section 5 summarizes this paper.

2. Related Works

2.1. The Problems of Imbalance Classification

Imbalance ratio is a factor that affects the classification, but it is not the only one. Recent works have indicated that there are other relevant issues hindering the performance of classifiers [13]:

- Overlapping: In Figure 1a, the examples from different classes that have similar characteristics are usually cross-distribution in the regions around decision boundary.
- Noise: In Figure 1b, an example is noise if it is far from other examples belonging to the same class. According to [13], it can be treated as the example corrupted by class label noise. In this paper, the definition of noise is extended, which any example corrupted by class label noise or attribute noise can be treated as noise [31].

![Figure 1](image.png)

**Figure 1.** The problems of imbalanced classification. (a) overlapping; (b) noise

2.2. Some Well-known Under-sampling Algorithms

Many under-sampling algorithms have been proposed including RUS, CNN, TL, OSS, NCL and CBU. The details of these algorithms are as follows.
• RUS: Random under-sampling (RUS) is the simplest under-sampling algorithm that builds a balanced dataset by randomly selecting a certain number of examples from the majority class [10].
• CNN: Condensed Nearest Neighbor Rule (CNN) [14] considers the examples far from decision border are less relevant for learning. Thus, CNN builds a balanced dataset by eliminating them from the majority class.
• TL: Tomek link [15] consists by two examples that satisfy the conditions as follows: the two examples $E_i$ and $E_j$ are from different classes; let $d(E_i, E_j)$ represent the distance between $E_i$ and $E_j$, there is not an example $E_l$ that satisfies the condition $d(E_i, E_l) < d(E_i, E_j)$ or $d(E_j, E_l) < d(E_i, E_j)$. TL builds a balanced dataset by removing the majority class examples in Tomek links.
• OSS: One-sided selection (OSS) [16] is an integration method of TL and CNN. Firstly, OSS uses TL to remove the boundary and noisy examples from the majority class. Then OSS uses CNN to remove some majority class examples far from the decision boundary. Finally, OSS builds a balanced dataset by combining the original minority class examples with the remaining majority class examples.
• NCL: Neighborhood Cleaning Rule (NCL) [17] builds a balanced dataset by using Edited Nearest Neighbor Rule (ENN) [18]. For any example in the training set, NCL classifies it according to its three nearest neighbors. If a majority class example is classified into minority class, then NCL removes this majority class example. Analogously, if a minority class example is classified into majority class, then NCL removes its nearest neighbors belonging to the majority class.
• CBU: Lin et al. [19] cluster the majority class to generate $k$ clusters by using the k-means clustering algorithm [20], with each cluster containing the similar examples. Then they use these cluster centers to represent the whole majority class.

3. The Proposed Method

Not all examples in the dataset are useful for classification [13]. Some examples may be useless for classification, while some may degrade the classification performance. The former is called redundant example and the latter is called noise. Real-world dataset always contains many similar examples and noise. For the similar examples, we select one of them to represent the whole group, while the remaining examples are treated as redundant examples. For the noisy example, we will remove it. In feature space, similar examples are relatively close to each other and noise is far from other examples belonging to the same class. Thus, the similar examples have a relative high density and noise has a relative low density. Based on this assumption, we propose a density-based under-sampling algorithm in which the local density peaks are used to represent the majority class. Local density peak is the local maxima in the density of examples, and it is relatively far from examples with a higher local density [27]. Meanwhile, noisy example has a low density, and it is far from other examples. Rodriguez et al. [27] propose a clustering algorithm (called CFSFDP for short) in which a new approach is introduced to find the local density peaks. However, CFSFDP estimates the densities by setting a cut-off distance artificially, which may affect the clustering results. To overcome this drawback, we use kernel density estimation to estimate an approximately probabilistic distribution for the majority class. Besides, in order to reduce the risk of selecting noise as the element of balanced dataset in the under-sampling process, we modify the metric $\gamma$ in CFSFDP by assigning different weights to local density $\rho$ and distance $\delta$. The details of our DBU are as follows.

3.1. Local Density $\rho$

Kernel density estimation (KDE) is a non-parametric method for estimating the probability distribution density, which combines all the kernel functions generated from the given data to obtain an estimated kernel density function. The KDE function is shown in Equation (1) [23, 24].
\[ f = \frac{1}{n \cdot h} \sum_{i=1}^{n} K \left( \frac{X_i - X}{h} \right) \]  \hspace{1cm} (1)

Where \( X_i \) is a data in the given sample, \( n \) is the number of data points, \( K(\cdot) \) is a kernel function and \( h \) is the bandwidth of kernel function. In this paper, we use Gaussian kernel to obtain the KDE function of majority class. The original Gaussian kernel function is shown in Equation (2) [21, 22].

\[ K_h = \frac{1}{\sqrt{2\pi h}} \exp \left\{ -\frac{1}{2} \left( \frac{t}{h} \right)^2 \right\} \]  \hspace{1cm} (2)

Thus, the local density \( \rho_i \) of example \( i \) is computed by Equation (3).

\[ \rho_i = \frac{1}{\sqrt{2\pi N_{+} \cdot h}} \sum_{j=1}^{N_{+}} \exp \left\{ -\frac{1}{2} \left( \frac{d_{ij}}{h} \right)^2 \right\} \]  \hspace{1cm} (3)

Where \( N_{+} \) is the number of majority class examples, \( d_{ij} \) is the distance from example \( j \) to example \( i \) computed by Heterogeneous Value Difference Metric (HVDM) [26].

\[ d_{ij} = HVDM(X_i, X_j) = \sqrt{\sum_{a=1}^{m} d_a^2(x_{ia}, x_{ja})} \]  \hspace{1cm} (4)

Where \( m \) is the number of attributes. The function \( d_a(x, y) \), the distance between \( x \) and \( y \) for attribute \( a \), is calculated by Equation (5).

\[ d_a(x, y) = \begin{cases} diff_a(x, y), & \text{if } a \text{ is numerical} \\ vdm_a(x, y), & \text{if } a \text{ is nominal} \end{cases} \]  \hspace{1cm} (5)

The function \( diff_a \) is defined in Equation (6).

\[ diff_a(x, y) = \frac{|x - y|}{4\sigma_a} \]  \hspace{1cm} (6)

Where \( \sigma_a \) is the standard deviation of attribute \( a \). The function \( vdm_a \) is defined in Equation (7).

\[ vdm_a(x, y) = \sum_{c=1}^{C} \left( \frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right) \]  \hspace{1cm} (7)

Where \( N_{a,x} \) is the number of \( x \) in attribute \( a \), \( N_{a,x,c} \) is the number of \( x \) in attribute \( a \) with the output class be \( c \), \( C \) is the number of output classes in dataset \( D \).

In Equation (3), the bandwidth \( h \) reflects the overall flatness of KDE curve. An inappropriate value of \( h \) can cause the KDE curve to be over-smoothing or under-smoothing [24]. Silverman’s rule of thumb is an effective method to calculate the optimal bandwidth \( (h^*) \) for Gaussian kernel [23].

\[ h^* = \left( \frac{4}{3} \right)^{1/5} \hat{\sigma} N_{+}^{-1/5} \]  \hspace{1cm} (8)

Where \( \hat{\sigma} \) is the standard deviation. However, the standard deviation is sensitive to noise, thus our DBU uses a corrected standard deviation to replace it [25].

\[ \hat{\sigma} = \frac{\text{Median}(|X_i - \text{Median}(X_i)|)}{0.6745} \]  \hspace{1cm} (9)

3.2. Distance \( \delta \)

For the distance \( \delta_i \) of example \( i \), it is measured by calculating its minimum distance to any other example with a higher density [27].
\[ \delta_i = \min_{j \neq i} \{d_{ij}\} \]  

(10)

Specially, for the highest density example, we take \(\delta = \max(d_{ij})\). Note that \(\delta_i\) is much larger than the typical nearest neighbor distance only for the local density peaks.

3.3. Metric \(\gamma\)

As Figure 2b shows, local density peaks are the examples (point 1, 9, 10 and 20) with a high density \(\rho\) and a large distance \(\delta\) while noise (point 26, 27 and 28) are the examples with a low density \(\rho\) and an anomalously large distance \(\delta\). CFSFDP measures the probability of an example being a density peak according to the product value of density \(\rho\) and distance \(\delta\). However, as Figure 2c shows, noisy examples (point 26, 27 and 28) have a high value \(\gamma\), when \(\gamma = \rho \delta\). This may increase the risk of selecting noise as an element of balanced dataset. In this paper, we introduce a new metric \(\gamma'\) to decrease this risk by assigning different weights to the local density \(\rho\) and distance \(\delta\). The definition of \(\gamma'\) is shown in Equation (11).

\[ \gamma'_i = \frac{(1+\alpha)\rho_i \cdot \delta_i}{\alpha \cdot \rho_i + \delta_i} \]  

(11)

Where \(\alpha\) indicates the relative importance of local density \(\rho\) and distance \(\delta\). In this paper, we set \(\alpha = 0.4\) according to the experimental results.

![Figure 2](image-url)

**Figure 2.** Our DBU algorithm in two dimensions; (a) Point distribution, (b) Decision graph \(\rho \cdot \delta\), (c) Decision graph \(n \cdot \gamma, \gamma = \rho \delta\), (d) Decision graph \(n \cdot \gamma', \alpha = 0.4\)

As Figure 2d shows, when we set \(\alpha = 0.4\), the noisy examples (point 26, 27 and 28) have a lower \(\gamma'\) than other examples. We select the top \(N\) examples in descending order of \(\gamma'\) to represent the majority class, so that our DBU can remove the noisy examples automatically and retain the useful information. Note that \(N\) is the number of minority class examples.
**ALGORITHM 1: DBU Algorithm**

*Step 1.* The majority class: \( Ma = \{ X \in D \mid y_i \in c_1 \} \), the minority class: \( Mi = \{ X \in D \mid y_i \in c_2 \} \), \( N_+ (N) \) is the number of majority (minority) class examples

*Step 2.* for \( i = 1 \) to \( N_+ \) do
   a. \( d_{ij} = \text{HVDM}(X_i, X_j) \)
   b. \( \rho_i = \text{KDE}(d_{ij}) \)
   c. \( \delta_i = \min_{j: \rho_j > \rho_i} (d_{ij}) \)
   d. \( \gamma'_i = \frac{(1 + \alpha) \rho_i \times \delta_i}{\alpha \times \rho_i + \delta_i} \)

*Step 3.* \( S_{Ma} \leftarrow \text{selects top} N, \text{examples from} Ma \text{in descending order of} \gamma', \text{where} \gamma' = \{ \gamma'_i, i=1,...,N_+ \} \)

*Step 4.* \( D' = S_{Ma} \cup Mi \)

*Step 5.* return \( D' \)

### 4. Experiments

#### 4.1. Datasets

In this paper, the experiments are conducted on 15 two-class datasets from the KEEL dataset repository (http://www.keel.es/). To validate our DBU, we use the fivefold cross-validation approach to build the training and test set. The characteristics of these datasets are shown in Table 1. \( Num \) is the number of examples, \( Atts \) is the number of attributes and \( IR \) is the imbalance ratio.

| Datasets     | Num | Atts | IR   |
|-------------|-----|------|------|
| wisconsin   | 683 | 9    | 1.86 |
| pimaimb     | 768 | 8    | 1.87 |
| glass0      | 214 | 9    | 2.06 |
| yeast1      | 1484| 8    | 2.46 |
| haberman    | 306 | 3    | 2.78 |
| vehicle1    | 846 | 18   | 2.9  |
| segment0    | 2308| 19   | 6.02 |
| page-blocks0| 5472| 10   | 8.79 |
| yeast-2vs4  | 514 | 8    | 9.08 |
| vowel0      | 988 | 13   | 9.98 |
| glass2      | 214 | 9    | 11.59|
| ecoli4      | 336 | 7    | 15.8 |
| abalone9-18 | 731 | 8    | 16.4 |
| yeast6      | 1484| 8    | 41.4 |
| abalone19   | 4174| 8    | 128.87|

#### 4.2. Evaluation Criterion

Average accuracy is a commonly used evaluation criterion in the traditional classification. This evaluation criterion has been proved unsuitable for imbalance classification, since it may lead to the misclassification of minority class examples. Consequently, other evaluation criteria have been proposed successively, including F-Measure, ROC and AUC [28]. In this paper, we use AUC as the evaluation criterion in the follow experiments.

#### 4.3. Experimental Results

In this paper, the experiments are conducted with C4.5 as the baseline classifier. Note that C4.5 uses the default parameters in the Weka software package. The experiments can be divided into two parts.
In the first experiment, we compare our DBU with RUS, CNN, TL, OSS, NCL and CBU on the 15 two-class KEEL datasets. Table 2 shows the AUC results obtained by C4.5. The best result in each experiment is highlighted in bold and Avg. represents the average AUC result. From Table 2, we observe that our DBU achieves the highest average AUC result. Table 3 shows the Wilcoxon’s test results for the comparisons between DBU and other algorithms, where $R_s$ and $R$ are the sums of ranks and $P_{\text{Wilcoxon}}$ is the $p$-value of Wilcoxon’s test result. If $P_{\text{Wilcoxon}}<0.05$, it means that the comparison is significantly different. From Table 3, we conclude that our DBU achieves significantly different results than other algorithms.

### Table 2. AUC results on KEEL datasets (%)

| Datasets       | RUS | CNN | TL | OSS | NCL | CBU | DBU |
|----------------|-----|-----|----|-----|-----|-----|-----|
| wisconsin      | 95.1| 92.2| 95.1| 91.3| **95.4** | 94.5 | **95.4** |
| pima-mb       | 72.7| 72.2| 71.0| 66.5| 71.6 | 72.7 | **73.6** |
| glass0         | **82.1** | 79.5| 77.5| 76.6| 78.7 | 74.4 | 79.3 |
| yeast1         | 71.3| 70.6| 70.5| 66.7| 70.6 | 70.4 | **71.5** |
| haberman       | 60.9| 62.9| 63.3| 63.6| 63.1 | 59.5 | **64.6** |
| vehicle1       | 70.4| 66.9| 70.8| 75.2| 73.8 | 70.3 | 75.0 |
| segment0       | 97.9| 97.2| 98.4| 98.1| 98.3 | 98.0 | **98.5** |
| page-blocks0   | 94.6| 94.0| 93.6| 93.8| 93.7 | **95.8** | 94.3 |
| yeast-2vs4     | 88.6| 81.1| 84.7| 89.2| 86.5 | 77.8 | **92.8** |
| vowel10        | 94.4| 91.9| **97.1** | 91.9| 95.9 | 91.0 | 94.8 |
| glass2         | 66.6| 68.6| 66.4| 71.9| 55.6 | 75.6 | **82.5** |
| ecoli4         | 86.1| 83.4| 81.4| 84.1| 81.4 | 86.2 | **89.8** |
| abalone-18     | 68.9| 61.4| 61.1| 61.4| 70.2 | 64.8 | **74.9** |
| yeast6         | 81.1| 55.4| 79.6| 77.1| 77.9 | 81.8 | **85.9** |
| abalone19      | 62.5| 50.0| 50.0| 50.0| 50.0 | **70.4** | 65.2 |
| Avg.           | 79.5| 75.2| 77.4| 77.2| 77.5 | 78.9 | 82.5 |

### Table 3. Wilcoxon’s test results on KEEL datasets

| Methods       | $R_s$ | $R$ | $P_{\text{Wilcoxon}}$ |
|---------------|-------|-----|-----------------------|
| DBU vs. RUS   | 109   | 11  | 0.0033                |
| DBU vs. CNN   | 119   | 1   | 0.0001                |
| DBU vs. TL    | 113   | 7   | 0.0012                |
| DBU vs. OSS   | 119   | 1   | 0.0001                |
| DBU vs. NCL   | 100   | 5   | 0.0012                |
| DBU vs. CBU   | 103   | 17  | 0.0118                |

In the second experiment, we corrupt the class labels and attributes of some majority class examples by introducing class noise and attribute noise respectively. Given a noise level $x$, the introduction schemes are as follows [29].

- For the class noise, we randomly choose $x/100$ of minority class examples to replace the same number of majority class examples.
- For the attribute noise, we randomly choose $x/100$ of majority class examples and corrupt each attribute $A_i$ of them with a random value $m$. Note that, if $A_i$ is numerical, $m$ is a value between the minimum and maximum of the domain; if $A_i$ is nominal, $m$ is any value in the domain.

In this experiment, we set $x=40$. Table 4 and 6 show the AUC results on class and attribute noise-modified datasets respectively. For both the class and attribute noise-modified datasets, the average AUC results of all these algorithms decrease. Our DBU wins the highest average AUC results no matter which type the noise is (class or attribute noise). Table 5 and 7 show the Wilcoxon’s test results for the comparisons between DBU and other algorithms. From Table 5 and 7, we observe that all the
$P_{\text{Wilcoxon}} < 0.05$, which means that the comparisons between our DBU and other algorithms are significantly different.

### Table 4. AUC results on class noise-modified datasets, $x=40$ (%)

| Datasets      | RUS  | CNN  | TL   | OSS  | NCL  | CBU   | DBU   |
|---------------|------|------|------|------|------|-------|-------|
| wisconsin     | 93.2 | 91.9 | 93.9 | 89.5 | 94.8 | 93.3  | **95.0** |
| pimaimb      | 71.2 | 67.3 | 72.9 | 70.9 | 72.5 | 66.9  | **73.5** |
| glass0        | 77.7 | 76.6 | 76.8 | 76.5 | **78.3** | 74.63 | **78.3** |
| yeast1        | 67.8 | 66.5 | 68.4 | 63.6 | 67.5 | 67.3  | **70.4** |
| haberman      | 58.8 | 60.2 | 59.0 | 59.3 | 58.9 | 54.1  | **62.7** |
| vehicle1      | 70.8 | 61.4 | 72.9 | 68.6 | 73.4 | 67.8  | **73.8** |
| segment0      | 96.8 | 94.7 | 97.4 | 96.8 | 96.1 | 97.8  | **98.7** |
| page-blocks0  | 92.8 | 91.3 | 93.4 | **94.0** | 93.6 | 91.3  | 93.9  |
| yeast-2vs4    | 84.1 | 78.9 | 84.9 | 86.1 | 83.2 | 75.7  | **90.8** |
| vowel0        | **92.5** | 89.9 | 91.1 | 90.9 | 91.9 | 90.4  | 92.3  |
| glass2        | 63.3 | 58.3 | 66.4 | 58.7 | 69.9 | 48.6  | **76.1** |
| ecoli4        | 75.5 | 83.7 | 81.4 | 84.1 | 79.1 | 80.5  | **87.7** |
| abalone9-18   | 61.4 | 58.9 | 57.7 | 59.5 | 64.5 | 65.1  | **69.1** |
| yeast6        | 80.4 | 67.4 | 79.6 | 77.5 | 77.9 | 74.6  | **82.2** |
| abalone19     | 56.3 | 50.0 | 50.0 | 50.0 | 50.0 | 60.5  | **64.9** |
| Avg.          | 76.2 | 73.1 | 76.4 | 75.1 | 76.8 | 73.9  | 80.6  |

### Table 5. Wilcoxon’s test results on class noise-modified datasets, $x=40$ (%)

| Methods       | $R_s$ | $R_c$ | $P_{\text{Wilcoxon}}$ |
|---------------|-------|-------|-----------------------|
| DBU vs. RUS   | 119   | 1     | 0.0001                |
| DBU vs. CNN   | 120   | 0     | 0.00006               |
| DBU vs. TL    | 120   | 0     | 0.00006               |
| DBU vs. OSS   | 119   | 1     | 0.0001                |
| DBU vs. NCL   | 105   | 0     | 0.0001                |
| DBU vs. CBU   | 120   | 0     | 0.00006               |

### Table 6. AUC results on attribute noise-modified datasets, $x=40$ (%)

| Datasets      | RUS  | CNN  | TL   | OSS  | NCL  | CBU   | DBU   |
|---------------|------|------|------|------|------|-------|-------|
| wisconsin     | 92.2 | 87.3 | 89.5 | 89.3 | 92.5 | 93.5  | **94.2** |
| pimaimb      | 67.8 | 66.3 | 69.9 | 63.9 | 65.9 | 68.9  | **72.4** |
| glass0        | **77.8** | 74.1 | 76.7 | 72.7 | 73.7 | 64.1  | 77.5  |
| yeast1        | 67.0 | 68.3 | **68.6** | 64.1 | 68.4 | 67.8  | 68.5  |
| haberman      | 55.2 | 56.2 | 50.0 | 52.6 | 51.6 | 49.6  | **59.0** |
| vehicle1      | 70.2 | 67.0 | 69.4 | 67.1 | 68.2 | 65.3  | **72.9** |
| segment0      | **98.6** | 97.6 | 97.9 | 98.2 | 97.7 | 98.3  | 97.5  |
| page-blocks0  | 92.4 | 93.8 | 92.8 | **94.2** | 93.0 | 90.1  | 91.3  |
| yeast-2vs4    | 82.3 | 79.1 | 80.1 | 81.0 | 78.7 | 76.9  | **86.9** |
| vowel0        | 91.2 | 88.8 | 90.2 | **93.6** | 90.2 | 91.5  | 93.1  |
| glass2        | 60.3 | 64.3 | 62.4 | 65.5 | 64.7 | 52.9  | **70.1** |
| ecoli4        | 79.2 | 78.2 | 80.0 | 79.5 | 81.2 | 77.7  | **83.8** |
| abalone9-18   | 67.3 | 65.2 | 60.4 | 61.9 | 62.6 | 66.4  | **69.5** |
| yeast6        | 78.3 | 77.8 | 71.9 | 69.1 | 77.3 | 65.7  | **82.4** |
| abalone19     | 56.3 | 50.0 | 50.0 | 50.0 | 50.0 | 54.2  | **60.4** |
| Avg.          | 75.7 | 74.3 | 74.0 | 73.5 | 74.4 | 72.2  | 78.6  |
Table 7. Wilcoxon’s test results on attribute noise-modified datasets, x=40 (%)  

| Methods      | Rs  | R  | P_{Wilcoxon} |
|--------------|-----|----|--------------|
| DBU vs. RUS  | 114 | 6  | 0.0008       |
| DBU vs. CNN  | 116 | 4  | 0.0004       |
| DBU vs. TL   | 113 | 7  | 0.0012       |
| DBU vs. OSS  | 114 | 6  | 0.0008       |
| DBU vs. NCL  | 114.5 | 5.5 | 0.0007       |
| DBU vs. CBU  | 117 | 3  | 0.0003       |

5. Conclusions
In this paper, we propose an anti-noise under-sampling algorithm, which uses local density peaks to represent the entire majority class. Two experiments are conducted to verify the effectiveness of our DBU. The first experiment is conducted on 15 two-class KEEL datasets and the second experiment is conducted on their noise-modified datasets. Experimental results have shown that our method has the best results on both the imbalanced datasets and their noise-modified datasets. The important fact that makes our DBU suitable for imbalance classification is that it removes the noisy examples and retains the useful information at the same time.

6. Acknowledgments
This work is supported by National Natural Science Foundation Youth Fund under Grant 51705436 and the Sichuan Province Science and Technology Support Program Project under Grant 2016GZ0194 and Grant 2018GZ0361.

7. References
[1] Peng LZ, Zhang HL, Yang B and Chen YH 2014 A new approach for imbalanced data classification based on data gravitation Information Sciences 288(C) pp 347-373.
[2] Idris A and Khan A 2016 Churn prediction system for telecom using filter–wrapper and ensemble classification Computer Journal 60(3) pp 410-430.
[3] Beyan C and Fisher R 2015 Classifying imbalanced data sets using similarity based hierarchical decomposition Pattern Recognition 48(5) pp 1653-1672.
[4] He HB and Garcia E 2009 Learning from Imbalanced Data IEEE Transactions on Knowledge and Data Engineering 21(9) pp 1263-1284.
[5] Krawczyk B, Galar M, Jeleń L and Herrera F 2016 Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy Applied Soft Computing 38(C) pp 714-726.
[6] Tax DM and Duin RP 1999 Support vector domain description Pattern Recognition Letters 20(11–13) pp 1191-1199.
[7] Lomax S and Vadera S 2016 A cost-sensitive decision tree learning algorithm based on a multi-armed bandit framework Computer Journal 60 pp 941-956.
[8] Wang BX and Japkowicz N 2010 Boosting support vector machines for imbalanced data sets Knowledge & Information Systems 25(1) pp 1-20.
[9] Chawla NV, Bowyer KW, Hall LO and Kegelmeyer WP 2002 SMOTE: synthetic minority over-sampling technique Journal of Artificial Intelligence Research 16(1) pp 321-357.
[10] Seiffert C, Khoshgoftaar TM, Hulse JV and Napolitano A 2010 RUSBoost: a hybrid approach to alleviating class imbalance IEEE Transactions on Systems, Man, and Cybernetics, PartA: Systems and Humans 40(1) pp 185–197.
[11] Drummond C and Holte RC 2003 C4.5, class imbalance, and cost sensitivity: why undersampling beats oversampling Proc. of the ICML Workshop on Learning from Imbalanced Datasets II pp 1-8.
[12] Sun Z, et al. 2015 A novel ensemble method for classifying imbalanced data Pattern Recognition 48(5) pp 1623-1637.
[13] Sáez JA, Luengo J, Stefanowski J and Herrera F 2015 SMOTE–IPF: addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering Information Sciences 291(5) pp 184-203.
[14] Angiulli F 2005 Fast condensed nearest neighbor rule International Conference on Machine Learning (Bonn, Germany) (New York: ACM) p 25-32.
[15] Tomek I 1976 Two Modifications of CNN IEEE Transactions on Systems Man and Communications SMC-6 pp 769–772.
[16] Kubat M and Matwin S 1997 Addressing the course of imbalanced training sets: one-sided selection In ICML p 179–186.
[17] Laurikkala J 2001 Improving identification of difficult small classes by balancing class distribution Conference on Ai in Medicine in Europe: Artificial Intelligence Medicine vol. 2101 (Berlin: Springer-Verlag) p 63-66.
[18] Wilson DL 1972 Asymptotic properties of nearest neighbor rules using edited data Systems Man & Cybernetics IEEE Transactions on SMC-2 pp 408-421.
[19] Lin WC, Tsai CF, Hu YH and Jhang JS 2017 Clustering-based undersampling in class-imbalanced data Information Sciences 409-410 pp 17-26.
[20] Hartigan JA and Wong MA 1979 A k-means clustering algorithm Applied Statistics 28(1) pp 100–108.
[21] Chen S 2015 Optimal bandwidth selection for kernel density functional estimation Journal of Probability & Statistics 2015(3) pp 1-21.
[22] Guidoum AC 2015 Kernel estimator and bandwidth selection for density and its derivatives R Journal.
[23] Silverman BW Density estimation for statistics and data analysis Chapman & Hall (London).
[24] Wand MP and Jones MC 1994 Kernel smoothing (London: CRC press).
[25] Committee AM 1989 Robust statistics – how not to reject outliers part 1: basic concepts Analyst 114(12) pp 1693-1697.
[26] Wilson DR and Martinez TR 1997 Improved heterogeneous distance functions Journal of Artificial Intelligence Research 11(1) pp 1-34.
[27] Rodriguez A and Laio A 2014 Clustering by fast search and find of density peaks science 344(6191) pp 1492-1496.
[28] Bradley AP 1997 The use of the area under the ROC curve in the evaluation of machine learning algorithms Pattern Recognition 30(7) pp 1145–1159.
[29] Zhu X and Wu X 2004 Class noise vs. attribute noise: a quantitative study Artif. Intell. 22(3) pp 177–210.