Fuzzy–synthetic minority oversampling technique: Oversampling based on fuzzy set theory for Android malware detection in imbalanced datasets

Yanping Xu, Chunhua Wu, Kangfeng Zheng, Xinxin Niu and Yixian Yang

Abstract
In previous work, imbalanced datasets composed of more benign samples (the majority class) than the malicious one (the minority class) have been widely adopted in Android malware detection. These imbalanced datasets bias learning toward the majority class, so that the minority class examples are more likely to be misclassified. To solve the problem, we propose a new oversampling method called fuzzy–synthetic minority oversampling technique, which is based on fuzzy set theory and the synthetic minority oversampling technique method. As the sample size of the majority class increases relative to that of the minority class, fuzzy–synthetic minority oversampling technique generates more synthetic examples for each minority class example in the fuzzy region, where the minority examples have a low degree of membership to the minority class and are more likely to be misclassified. Using the new synthetic examples, the classifiers build larger decision regions that contain more minority examples, and they are no longer biased to the majority class. Compared with synthetic minority oversampling technique and Borderline–synthetic minority oversampling technique methods, fuzzy–synthetic minority oversampling technique achieves higher accuracy on both the minority class and the entire datasets.

Keywords
Fuzzy–synthetic minority oversampling technique, imbalanced datasets, oversample, fuzzy region, Android malware detection

Date received: 9 September 2016; accepted: 13 March 2017

Academic Editor: Ibrahim Kamel

Introduction
Malware (malicious software) continues to increase with the rapid development of mobile networks, especially on the Android platform and services. In November 2015, McAfee Labs Threats Report announced that in first three quarters of the year, mobile malware incidents increased by approximately 4 million, more than twice the same period of 2014, and the total number of mobile malware incidents reached approximately 10 million. Android malware can infect and harm Android platforms and services through various methods such as malicious websites, spam, malicious SMS messages, and malware-bearing advertisements. Android malware causes security threats such as phishing, banking-Trojans, spyware, bots, root exploits, SMS fraud, premium dialers, and fake installers. Moreover, the most-frequent malware behaviors are escalating privileges, taking remote control, incurring financial charges, and stealing personal information.
information. Therefore, Android malware detection is a necessary and pressing task. There are two effective types of approaches: static analysis by decompiling the source code and dynamic analysis by monitoring application execution at runtime. The datasets for most Android malware detection experiments are composed of both benign and malicious applications. The benign applications are collected from Google Play Store, and the malware applications are collected from malware share websites or by researchers themselves. Because it is difficult to create a comprehensive malware collection, the datasets used in the experiments are typically imbalanced: number of benign applications is larger than the number of malware applications. However, this imbalance problem is usually not considered seriously.

Dataset imbalance is not a new problem; it occurs in many real-world situations including intrusion detection, risk management, text categorization, and information filtering. In these problems, just as in Android malware detection, it is important to correctly identify the minority class. There is a strong learning bias toward the majority class when using imbalanced datasets. As a result, the minority class examples are more likely to be misclassified; therefore, the minority class should be given more attention. Machine learning from imbalanced datasets has been widely researched. Overall, the approaches for solving the imbalance dataset problem can be divided into two types: resampling methods and imbalanced learning algorithms. Resampling tries to balance the class distribution inside the training data through both oversampling method (adding examples to the minority class to approach the majority class size) and undersampling method (removing examples from the majority class to approach the minority class size). The most famous oversampling method is the synthetic minority oversampling technique (SMOTE). Later, studies have proposed modifications to existing methods that have achieved good performances on balanced datasets or proposed new algorithms to resolve the imbalance problem.

In the SMOTE method, the minority class is oversampled to generate synthetic minority examples. For each minority example, the k-nearest neighbors (KNNs) in the minority class are calculated. Then, some neighbors are randomly selected depending on the oversampling rate, which is the integer ratio of the sample size of the majority class to that of the minority class. Subsequently, new minority synthetic examples are generated along the lines between the minority example and its selected nearest neighbors. These synthetic minority synthetic examples increase the number of the examples in the minority class, balance the distribution of the datasets, and build larger decision regions that contain more minority class examples. The results of experiments show that the SMOTE approach improves the accuracy of classifiers for the minority class.

The imbalance ratio (IR) characterizes the imbalance degree of the dataset. It is equal to the sample size of the majority class divided by the size of the minority class. The IR is considered to be an important factor that affects the classification accuracy of the minority class. However, recent studies have indicated that the IR is not the only factor. For some datasets with a high IR, standard classifiers achieve minority class accuracy that is still high. Other factors also reduce minority class classification accuracy in the imbalanced datasets, such as overlap between classes and the presence of many minority examples within the majority class. However, when multiple factors occur together in imbalanced datasets, the accuracy of minority class classifications can be seriously affected. Therefore, data distribution can have a large impact on the quality of imbalanced datasets and on classifier performance. The membership degree of the examples to the classes is computed to measure the distribution of the datasets. Therefore, data distribution and the membership degree of the examples to the classes lead to the main goals of our study.

Our main goal is to solve imbalanced dataset distributions by adding new minority examples and to improve the accuracy of the minority class. The most important task is to construct a principle for oversampling the minority class. This principle should act as a guide to help determine which minority examples should be used to generate new synthetic examples and how many new synthetic examples should be generated for each minority example.

The concept of membership degree was first proposed in fuzzy set theory: it reflects the degree of uncertainty about whether an example belongs to a set, and it permits the gradual assessment of the membership of the examples in a set. Membership degree quantifies the relationship of each example to a given dataset in a range [0.0, 1.0]. When the value of the membership degree of an example equals 1.0, that example is sure to belong to the dataset. When the value of membership degree is between 0.0 and 1.0, the example is fuzzy and only partially belongs to the dataset. Fuzzy set theory provides a methodology for data analysis; here, we extend fuzzy set theory to the task of Android malware detection in imbalanced datasets.

According to fuzzy set theory, minority examples in the imbalanced datasets that have a low membership degree to the minority class can easily be misclassified. However, the decision region of the minority class can be broadened to ensure correct classification by creating new minority examples that are similar to the minority class. The result is that classifiers are no longer biased away the minority class. Based on the above concepts, a fuzzy region is defined to contain minority
examples with low membership degree, and there is a need to generate new synthetic examples. To generate such examples, the following two questions should be answered:

(1) What is the range of the fuzzy region?
(2) How many new synthetic examples should be generated for each minority example in the fuzzy region?

The reason for question (1) is that when the range of the fuzzy region is small and contains few minority examples, the number of new synthetic examples needed is small, and the data distribution may still be imbalanced. In contrast, when the range of the fuzzy region is large and contains more minority examples, redundant examples may be generated that increase the cost of system resources and waste time. Therefore, answering question (1) involves finding proper range of the fuzzy region through a series of experiments. The reason for question (2) is that minority examples with low membership degree are more important; therefore, more synthetic examples should be generated for them. Answering question (2) involves combining the IR of the imbalanced dataset with fuzzy-SMOTE to calculate the appropriate number of synthetic examples to generate for each minority example.

In our work, Android malware applications are viewed as the minority class in the imbalanced dataset. To improve the classification performance of the minority class, we propose a new algorithm called fuzzy-SMOTE, which is based on the membership degree in fuzzy set theory and the SMOTE method. First, fuzzy-SMOTE computes the membership degree of each minority example to the minority class. Second, fuzzy-SMOTE explores the fuzzy region containing the minority examples that have low membership degrees and are likely to be misclassified. Third, fuzzy-SMOTE generates new synthetic examples for the minority examples in the fuzzy region, using the same process as SMOTE. Compared with other SMOTE-based methods, fuzzy-SMOTE pays more attention to find which minority examples should be oversampled and determine how many new synthetic examples should be generated for each minority example. Our research makes several significant contributions as follows:

- We propose a new oversampling method called fuzzy-SMOTE, based on fuzzy set theory and SMOTE, to solve the imbalance problem in Android malware detection. The results indicate that fuzzy-SMOTE improves performance both on the accuracy of the minority class and on entire datasets.
- Fuzzy-SMOTE calculates the membership degree of each minority example to the minority class. In addition, it defines a fuzzy region that contains the minority examples with low membership degrees that are more likely to be misclassified.
- Combined with the IR, fuzzy-SMOTE generates more synthetic examples for each minority example with low membership degree. The new synthetic examples broaden the decision region of the minority class and reduce classifier bias to the majority class.

The remainder of this article is organized as follows: section “Related work” reviews related works on Android malware detection methods and imbalanced learning. Section “Methodology” presents the details of the research methodology. Section “Experiment and discussion” introduces and discusses the experiments, and section “Conclusion” provides conclusions.

Related work

Methods for Android malware detection

Android malware detection methods are divided into three categories: static analysis, dynamic analysis, and hybrid analysis. Static detection methods analyze the decompiled source code at the binary level or the API level without executing the Android applications. Dynamic detection methods analyze application behaviors at runtime by monitoring behaviors indicative of Android malware activity. Hybrid analysis methods combine both static and dynamic techniques.

Static analysis methods contain rule policy, and machine learning methods. Machine learning detection process consists of feature extraction, feature selection, and machine learning. In feature extraction part, permissions, API, combination of permissions and APIs, system calls, signatures, and component information are extracted as features from the decompiled source files. In feature selection part, the common methods are Information Gain, CHI, and Fisher Score. Machine learning classification methods include KNN, naive Bayes, decision tree, logistic regression, support vector machines (SVMs), AdaBoost, and k-means. However, dynamic analysis methods focus on application behavior collection and relational analysis. The behavior objects include semantic-based approach, data flow, system call sequence, inter-application communication (IPC), and privilege escalation.

Imbalanced datasets have been widely used in previous studies. In Cen et al., five training datasets were used in which the ratios of malicious applications to benign applications in the datasets were 0.0%, 0.23%, 4.79%, 0.25%, and 8.07%. In Aafer et al., the dataset
included 16,000 benign and 3987 malware applications. In Sanz et al., the dataset included 1811 benign and 249 malware applications, and Jang et al., included 109,193 benign and 9990 malware applications. Although the malware detection accuracy was high, reaching 99% in Cen et al., the researchers did not pay attention to the imbalance problem or take special measures to solve it. However, in our work, we aim to explore the nature of the imbalance problem and its solution, and we propose a new oversampling method to ensure the performance of imbalanced datasets in malware detection.

**Methods for imbalanced learning**

The methods to solve the problem of the imbalanced dataset are divided into two types: data resampling methods and imbalanced learning algorithms. The data resampling methods aim to balance the given dataset by adding or removing samples. While imbalanced learning algorithms focus on modifying the existing machine learning mechanism or creating new algorithms to improve the detection rate of minority class.

Resampling methods consist mainly of two types: oversampling and undersampling. Oversampling duplicates existing examples or generates new examples for the minority class, while undersampling removes examples randomly from the majority class. Both oversampling and undersampling methods have been combined to solve the imbalance problem. Specifically, random oversampling is a non-heuristic method that balances the class distribution by randomly replicating minority class examples. The most famous of these methods is SMOTE, which generates new (synthetic) minority examples between each minority example and its nearest neighbors. This method makes decision regions larger and less specific; consequently, classifiers achieve better performance. Han et al. presented Borderline-SMOTE, which oversampled only the minority examples near the borderline. Borderline-SMOTE achieved a better true positive (TP) rate and F-measure than SMOTE and random oversampling methods. Chawla et al. integrated SMOTE and the standard boosting method to create SMOTEBoost, which synthesized minority class examples and indirectly changed the updating weights to compensate for skewed distributions. Bunkhumpornpat et al. proposed Safe-Level-SMOTE, which oversampled minority instances along the same line at a safe level. The safe level was defined based on the nearest-neighbor minority instances. Guo and Viktor proposed DataBoost-IM, which generated synthetic examples for the majority and minority classes. Qiong et al. proposed genetic algorithm-based synthetic minority oversampling technique (GASMOTE) to improve SMOTE-based on the genetic algorithm (GA) by setting different sampling rates for different minority samples. Moreover, SMOTE has been integrated with numerous machine learning and deep learning algorithms. Undersampling methods also have many different forms including random undersampling, inverse random undersampling, and the EasyEnsemble and BalanceCascade undersampling strategies.

Imbalanced learning algorithms are divided into four types: cost-sensitive methods, kernel-based learning methods, active learning methods, and ensemble methods. (1) Cost-sensitive methods focus on the costs of the misclassified examples using different cost matrices and have outperformed various other empirical sampling methods. Cost-sensitive methods can be categorized into three types: those that use misclassification costs as a form of dataspaces weighting, those that use cost-minimization techniques to combine ensemble methods to improve performance, and those that indirectly incorporate cost-sensitive functions into classification paradigms to train classification models. Several different cost-sensitive boosting methods based on the AdaBoost algorithm have been proposed. Sun et al. changed AdaBoost’s weight-updating strategy by introducing cost items. They proposed three cost-sensitive boosting methods: AdaC1, AdaC2, and AdaC3. Song et al. proposed BABoost, which assigned higher weights to the misclassified examples in the minority class. Cost-sensitive DTs and cost-sensitive neural networks have also been widely studied for imbalanced learning. (2) Kernel-based learning methods such as SVM mainly center on statistical learning and can achieve relatively robust classification results when addressing imbalanced datasets. SVM adopts different error cost terms to shift the decision boundary away from positive examples to guarantee that negative examples are classified correctly. (3) Active learning methods are often integrated into kernel-based learning methods. As an active learning method, SVM is used to select the most informative examples from unknown training data while retaining the kernel-based methods. (4) Ensemble methods are effective in averaging prediction errors and reducing bias and error variance. Most current ensemble methods have similar procedures for imbalanced datasets: resampling and voting. Most of these ensembles are based on known strategies from bagging, boosting, or random forests. Moreover, the diversity of ensemble approaches determines the final prediction accuracy of an example. Consequently, creating algorithm strategies that ensure and enlarge diversity is a critical task.

**Methodology**

**SMOTE**

Chawla et al. proposed the SMOTE approach, which can improve classifier minority class accuracy on
imbalanced datasets. SMOTE performs oversampling by creating synthetic samples for the minority class rather than duplicating existing minority class samples. The synthetic samples are generated based on a random $k$ grouping of the minority class’ nearest neighbors. Then, the generated synthetic samples are added to the dataset when training the machine learning classifier model.

The principles of the SMOTE algorithm are as follows: (1) find $k$NNs $x$ for each sample $x$ in the minority class based on Euclidean distance. Commonly, $k$ is set to 5. (2) Calculate the difference between the minority example and a nearest neighbor selected randomly from the $k$-minority class nearest neighbors. (3) Multiply the difference by a random number between 0 and 1, and add it to the original sample $x$. A synthetic sample is generated as follows

$$x_{\text{New}} = x + \text{rand}(0, 1) \ast (x - x)$$ (1)

More synthetic samples of the minority class can be obtained using the above steps. The new synthetic samples can maintain the distribution of the original minority class and balance the distribution of the entire training set. Therefore, SMOTE causes classifiers to create a new decision boundary that is no longer biased to the majority class.

SMOTE can improve classifier performances on imbalanced datasets. Meanwhile, improving SMOTE has attracted much research attention; consequently, several improved algorithms have been proposed that include Borderline-SMOTE and SMOTEBoost.\textsuperscript{35} Borderline-SMOTE takes only the minority examples near the borderline to oversample based on SMOTE. SMOTEBoost is a combination of SMOTE and the boosting procedure that creates synthetic examples for minority class and thus indirectly changes the updating weights and the compensation for skewed distributions. These methods improve classifier performance on the minority class and overall performance on imbalanced datasets; however, they do not consider the distribution of the imbalanced dataset nor do they specifically consider the easily misclassified minority examples. Therefore, we propose an improved algorithm, called fuzzy-SMOTE, which is based on the membership degree concept from fuzzy set theory and the SMOTE method.

**Fuzzy-SMOTE**

For balanced datasets, most machine learning methods build an impartial decision boundary to achieve good overall performance. However, in the original imbalanced dataset space, this decision boundary tends to be biased toward the majority class because fewer minority examples exist to train the classification model.\textsuperscript{8} To better identify the minority class, we concentrate on oversampling the minority examples. Therefore, we explore the distribution of the minority and majority classes and generate additional synthetic samples for those minority examples that are most easily misclassified. This is the difference between our method and the existing oversampling methods.

In the fuzzy-SMOTE method, we first calculate the membership degree of each minority example to the minority class based on fuzzy set theory; then, we define a fuzzy region in which the minority class examples have low membership degree to the minority class and are easily misclassified; and finally, by combining the imbalance factor, we create synthetic samples for the minority examples in the fuzzy region and add them to the original training set.

Suppose that the training set is $X = \{x_1, x_2, \ldots, x_N\}$, its size is $N$; the majority class is $X_p = \{x_{p1}, x_{p2}, \ldots, x_{pN_p}\}$ and has a size of $N_p$; and the minority class is $X_q = \{x_{q1}, x_{q2}, \ldots, x_{qN_q}\}$, whose size is $N_q$. The IR is defined based on the size of the majority class divided by the size of the minority class and used to characterize the degree of imbalance of the dataset. The IR function is defined as follows

$$\text{IR} = \frac{N_q}{N_p}$$ (2)

The IR is related to the oversampling rate of the training minority class. The oversampling rate $K$ denotes the number of the synthetic samples to be generated and is defined as follows

$$K = \lfloor \text{IR} \rfloor$$ (3)

where $\lfloor \text{IR} \rfloor$ is the rounded-down value of IR.

The fuzzy-SMOTE method is executed in the following steps:

**Step 1: membership degree.** Let the classes $C_p$ and $C_q$ be the distributions of the majority and minority classes, respectively. Let $V_p$ and $V_q$ be the centroids of classes $C_p$ and $C_q$, respectively, and let $\mu_{C_i}(x_i)$ be the membership degree of an example $x_i$ to a given class $C$.\textsuperscript{15} These can be defined as follows

$$V_p = \frac{1}{N_p} \sum_{n=1}^{N_p} x_{pn}$$ (4)

$$V_q = \frac{1}{N_q} \sum_{n=1}^{N_q} x_{qn}$$ (5)

$$\mu_{C_i}(x_i) = \left[ \sum_{k=p}^{q} \left( \frac{||x_i - V_p||^2}{||x_i - V_k||^2} \right)^{1/(m-1)} \right]^{-1}$$ (6)
where \( ||x_j - V_i|| \) and \( ||x_j - V_k|| \) are the Euclidean distance of example \( x_j \) to the centroids \( V_i \) and centroids \( V_k \) (\( k \) is set to \( p \) or \( q \)). Here, \( m \) denotes the fuzziness of membership to each class, and it is set to 2 in our work. \( \mu_{C_i}(x_j) \) denotes the membership degree with which an example belongs to a class, and its range is \([0.0, 1.0]\). In this work, \( x_j \) belongs either to class \( C_p \) or \( C_q \), such that its total membership to both classes is

\[
\mu_{C_p}(x_j) + \mu_{C_q}(x_j) = 1
\] (7)

A large value for \( \mu_{C_i}(x_j) \) indicates that example \( x_j \) has high membership degree to the class \( C_i \) and, therefore, can easily be classified to class \( C_i \). As shown in Figure 1, the membership degree of \( x_j \) to the class \( C_q \) is larger than that of \( x_j \); consequently, \( x_j \) is easier to classify to \( C_q \).

**Step 2: fuzzy region.** Suppose example \( x_{qj} \) is a minority class example. When \( \mu_{C_p}(x_{qj}) \leq 0.5 \) and \( \mu_{C_q}(x_{qj}) > 0.5 \), example \( x_{qj} \) is closer to \( C_p \) than to \( C_q \), and example \( x_{qj} \) is likely to be misclassified to \( C_p \). Therefore, we define a fuzzy region to contain the minority examples that have low membership degrees to the minority class and are likely to be misclassified. It is important to determine the range of the fuzzy region. When the range is small, the classification region biased to the majority class is little changed. In contrast, when the range is large, some redundant examples are generated. We will explore the proper range of the fuzzy region through the experiments.

In addition, examples in which the value of \( \mu_{C_p}(x_{qj}) \) is smaller should be given more attention, and additional synthetic samples should be generated for them.

**Step 3: oversampling rate.** For every minority example \( x_{qj} \) of the training minority class in the fuzzy region, the oversampling rate \( K_{qj} \) can be calculated as follows

\[
K_{qj} = \left[ 1 - \frac{\mu_{C_p}(x_{qj})}{\mu_{C_q}(x_{qj})} \right] \times K
\] (8)

When \( \mu_{C_q}(x_{qj}) \) is smaller, the value of \( K_{qj} \) is larger. Therefore, more synthetic examples can be generated for the minority examples with small membership degrees.

**Step 4: oversampling.** First, we find the KNNs for each sample \( x_{qj} \) in minority class based on Euclidean distance. Here, \( k \) is set to 5, as in SMOTE. Second, we randomly select one neighbor \( \tilde{x}_{qj} \) from the KNNs and calculate the difference \( \text{diff}_j \) between \( x_{qj} \) and \( \tilde{x}_{qj} \) in formula (9). Then, \( \text{diff}_j \) is multiplied by a random integer 0 or 1. Finally, a new synthetic minority example is generated between \( x_{qj} \) and its nearest neighbor \( \tilde{x}_{qj} \) is shown in formula (10).

\[
\text{diff}_j = x_{qj} - \tilde{x}_{qj}, \quad j = 1, 2, \ldots, K_{qj}
\] (9)

\[
x_{q_j, \text{new}} = x_{qj} + \text{rand} \cdot \text{int}(0, 1) \cdot \text{diff}_j
\] (10)

The above procedure is repeated for each minority example in the fuzzy region until \( \sum K_{qj} \) synthetic minority examples have been generated. Then, the new synthetic minority examples are added to the original training set, which is then sent to the classifier and used to train classification models.

**Experiment evaluation**

To evaluate the performance of our proposed method, the datasets are divided into training sets and testing sets using 10-fold cross-validation. In this method, a full dataset is split into 10 independent folds. In turn, nine folds are used to train the classification models, and the remaining fold is applied as the testing set to validate and assess the model. Consequently, each fold is used once as a testing set. Formally, the complete cross-validation estimate is the average of the 10-fold estimates computed in a loop. While 10-fold cross-validation can be computationally expensive, it does not require as much data as a fixed arbitrary test set and is quite suitable for small datasets.

Traditionally, with balanced datasets, machine learning classifiers are evaluated by overall accuracy. However, to meet the special situations of imbalanced datasets, other evaluation measures are adopted to provide comprehensive assessments. In our work, the negative class (malicious applications) is the minority class and the positive class (benign applications) is the majority class. The evaluation metrics are defined as follows

\[
\text{acc}_+ = \text{recall} = \frac{TP}{TP + FN}
\]

\[
\text{acc}_- = \frac{TN}{TN + FP}
\]

\[
\text{overall} - \text{acc} = \frac{TP + TN}{TP + FN + FP + TN}
\]

\[
\text{precision} = \frac{TP}{TP + FP}
\]

\[
F\text{-measure} = \frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}} = \frac{2 \times TP}{2 \times TP + FN + FP} (\beta = 1)
\]

\[
G\text{-mean} = \sqrt{\text{acc}_+ \times \text{acc}_-} = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}}
\]
where \( \text{acc}^+ \) denotes classifier accuracy for the majority class (positive examples), and \( \text{acc}^- \) denotes classifier accuracy for the minority class (negative examples). Thus, \( \text{overall} - \text{acc} \) represents the overall accuracy of the classifier on the complete dataset, and \( \text{precision} \) is the percentage of predicted positive applications that are actually benign. Here, \( \text{recall} \) is the percentage of correctly classified positive examples. In addition, \( \text{F-measure} \) integrates both \( \text{recall} \) and \( \text{precision} \) as a measure of the effectiveness of the classification models. When both \( \text{recall} \) and \( \text{precision} \) are high, the \( \text{F-measure} \) value is also high. The parameter \( \beta \) reflects the relative importance of \( \text{recall} \) and \( \text{precision} \) and is usually set to 1.0.\(^{36}\) Furthermore, \( \text{G-mean} \) evaluates the degree of inductive bias in terms of positive and negative accuracy. When the difference between \( \text{acc}^+ \) (\( \text{recall} \)) and \( \text{acc}^- \) is small, the value of \( \text{G-mean} \) is high.\(^{59}\)

### Experiment and discussion

This section presents the detailed methodology and results of the experiments. For each method, the imbalanced evaluation values (such as \( \text{acc}^+ \), \( \text{acc}^- \), \( \text{F-measure} \), and \( \text{G-mean} \)) are the average values of the 10-fold cross-validation experiments.

#### Data source

We used 10 datasets in the experiments as described in Table 1. Eight of these datasets, termed S1–S8, include both benign examples and malicious examples. We collected 1017 benign examples from the Google Play Store in March 2015. These include top applications from 14 categories including Office, Lifestyle, Travel, Children, Shopping, Education, Finance, Photography, Social, Tools, Reading, Multimedia, Sports, and Themes. We chose the most frequently downloaded applications in each category. The malicious examples were collected from the UNB ISCX Android Botnet Dataset and span a period from 2010 to 2014.\(^{60}\) These malicious examples are analyzed in Kadir et al.\(^{61}\) We used only four Android malware sets: DroidDream, Nickspay, MisoSMS, and SandDroid. The original number of examples in the DroidDream family was 343. To achieve our goals, we randomly duplicated 217 examples to generate a DroidDream1 set and randomly selected some examples to generate the DroidDream2, DroidDream3, and DroidDream4 datasets. Finally, the benign samples and the malware sets were integrated to form the S1–S8 datasets.

The other two datasets, termed S9 and S10, were collected from University of California, Irvine (UCI) Machine Learning Repository (http://archive.ics.uci.edu/ml).\(^{62}\) S9 originally stems from UCI. S10 contains 1528 majority examples (from the “M” class in Abalone) and 396 minority examples (some random examples from the “I” class in Abalone). Their descriptions are also shown in Table 1.

The Android permission mechanism is used to restrict application access to system resources and guarantee runtime security. Permissions are declared in an AndroidManifest.xml file and requested when the applications perform sensitive behaviors through a corresponding set APIs. Malware and benign applications tend to request permissions differently. Malware applications often request higher-risk permissions such as SEND_SMS, RECEIVE_SMS, and READ_SMS. Therefore, to some degree, malware can be differentiated from benign applications based on the permissions listed in the AndroidManifest.xml file. Previous works have demonstrated that permission features can identify whether an application is malicious.\(^{63-65}\) Therefore, in our work, we extract the permissions from decompiled AndroidManifest.xml files and use them as features for detecting malware. In each dataset, the permission features are the union of permissions declared in the AndroidManifest.xml file. The numbers of features in the different datasets are listed in Table 1.

#### Accuracy bias to the majority class

For imbalanced datasets, machine learning algorithms create a decision boundary biased to the majority class. Consequently, the minority class examples are more

| Datasets | Content | Features | Size (majority/minority) | Imbalanced ratio |
|----------|---------|----------|--------------------------|-----------------|
| S1       | Benign + DroidDream1 | 126      | 1017/560                 | 1.82            |
| S2       | Benign + DroidDream  | 126      | 1017/343                 | 2.97            |
| S3       | Benign + Nickspay     | 126      | 1017/199                 | 5.11            |
| S4       | Benign + DroidDream2 | 126      | 1017/120                 | 8.48            |
| S5       | Benign + MisoSMS      | 124      | 1017/100                 | 10.17           |
| S6       | Benign + DroidDream3 | 126      | 1017/80                  | 12.71           |
| S7       | Benign + DroidDream4 | 126      | 1017/54                  | 18.83           |
| S8       | Benign + SandDroid    | 116      | 1017/44                  | 23.11           |
| S9       | Pima                 | 8        | 500/268                  | 1.87            |
| S10      | Abalone              | 8        | 1528/396                 | 3.94            |
likely to be misclassified. In this section, we use S1, S2, S4, S6, and S7 datasets to explore the accuracy bias to the majority class in imbalanced datasets. The minority classes are similar in the S1, S2, S4, S6, and S7 datasets except for their number, so the datasets are comparable. We used SVM as the classifier. The results are shown in Table 2.

As shown in Table 2, the results indicate that as the IR grows, the accuracy of minority class \( \text{acc}_- \) classification decreases while the accuracy of majority class \( \text{acc}_+ \) classification and the overall accuracy of the dataset \( \text{overall} - \text{acc} \) increase. This trend is consistent because the classifiers tend to be biased to the majority class. When the IR is larger than 18, the value of \( \text{acc}_- / \text{acc} \) decreases by more than 20%. In this case, misclassifications of the minority examples (malware applications) may result in serious consequences. Meanwhile, \( \text{recall} \) and \( \text{precision} \) reflect the correct classification of the majority class (positive), and their values increase as the IR increases. The \( \text{F-measure} \) value also increases because the \( \text{F-measure} \) integrates \( \text{recall} \) and \( \text{precision} \). In addition, \( \text{G-mean} \) is the square of \( \text{acc}_- \times \text{acc}_+ \). When the difference between \( \text{acc}_- \) and \( \text{acc}_+ \) is small, the value of \( \text{G-mean} \) is large. As the IR increases, the difference between \( \text{acc}_- \) and \( \text{acc}_+ \) also increases; therefore, the value of \( \text{G-mean} \) decreases.

### Table 2. SVM performance on the imbalanced datasets.

| Datasets | IR  | \( \text{acc}_+ \) | \( \text{acc}_- \) | \( \text{overall} - \text{acc} \) | \( \text{precision} \) | \( \text{recall} \) | \( \text{F-measure} \) | \( \text{G-mean} \) |
|----------|-----|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| S1       | 1.82| 0.9656          | 0.9536          | 0.9613          | 0.9744          | 0.9656          | 0.9699          | 0.9595          |
| S2       | 2.97| 0.9695          | 0.9152          | 0.9558          | 0.9715          | 0.9695          | 0.9705          | 0.9418          |
| S4       | 8.48| 0.9832          | 0.8151          | 0.9656          | 0.9786          | 0.9833          | 0.9808          | 0.8940          |
| S6       | 12.71| 0.9882         | 0.7625          | 0.9717          | 0.9815          | 0.9882          | 0.9848          | 0.8661          |
| S7       | 18.83| 0.9931          | 0.7400          | 0.9839          | 0.9902          | 0.9931          | 0.9916          | 0.8653          |

SVM: support vector machine; IR: imbalance ratio.

### Performance for fuzzy region

In section “Fuzzy-SMOTE,” the fuzzy region in fuzzy-SMOTE method is defined to contain the minority class examples, which are easily misclassified to the wrong class and should be given more attention. In this section, we explore the performance of the fuzzy region at different ranges. First, we calculate the membership degree \( \mu_{C_q}(x_q) \) of each minority example to the minority class. Using the S1 and S8 datasets as examples, the distribution diagrams of the membership degree are shown in Figure 2. The results show that the range of the membership degree of some minority examples is concentrated in the range \([0.0, 0.5]\); these examples are more likely to be misclassified to the majority class. Therefore, these minority examples should be included in the fuzzy region. This part of the experiments explored the proper range for the fuzzy region. Then, we used fuzzy-SMOTE to create the synthetic samples and used SVM as the classifier. The algorithms with different range parameters are executed 10 times. The results that appear repeatedly are taken as the final results.

The \( \text{G-mean} \) performances of SVM using fuzzy-SMOTE with different fuzzy region ranges are presented in Figure 3. Figure 3(a) shows that when the range of the fuzzy region is smaller than 0.7, the
number of synthetic samples generated by fuzzy-SMOTE increases. However, after the range of the fuzzy region exceeds 0.7, the numbers change very little. Similarly, Figure 3(b)–(d) shows that as the fuzzy region range increases, the classifier achieves better performance on $\text{acc}/C_0$, $F$-measure, and $G$-mean. However, after the range of the fuzzy region exceeds 0.7, the results of $\text{acc}$ and $G$-mean change very little. We can conclude that the proper range of the fuzzy region is [0.0, 0.7]. Moreover, fuzzy-SMOTE does not need to create synthetic samples for examples in the minority class whose membership degree is high because they are already highly likely to be classified correctly.

Comparison with existing oversampling methods

SMOTE\textsuperscript{10} and Borderline-SMOTE\textsuperscript{56} have been shown to improve classifier accuracy for the minority class. In this section, we compare several sample synthesis methods including SMOTE, Borderline-SMOTE, and fuzzy-SMOTE with the fuzzy region [0.0, 1.0]. The oversampling rate of each training dataset is $|IR|$ in the SMOTE and Borderline-SMOTE methods. The goal is to try to balance the number of the minority and majority classes in the training set. For the fuzzy-SMOTE method, the oversampling rate of each minority example is determined by its membership degree to the minority class. After performing oversampling of the original training sets using the different synthesis methods, we applied SVM as the classifier.

The comparison results are illustrated in Tables 3–7. As shown in Tables 4 and 7, all the oversampling methods improve $\text{acc}_-$ and $G$-mean. Fuzzy-SMOTE achieves the best performance on $\text{acc}_-$ and $G$-mean, and it improves the $\text{acc}_-$ scores on S7 by 16%. On datasets S2, S3, S5, S7, S8, and S9 Borderline-SMOTE achieves higher scores than SMOTE on $\text{acc}_-$ and $G$-mean. Meanwhile, both Borderline-SMOTE and SMOTE sacrifice the performance of $\text{acc}_+$ and $F$-measure. On datasets S2, S3, and S8, the losses to $\text{acc}_+$ and $F$-measure are smallest from fuzzy-SMOTE. Moreover, as shown in Table 3, fuzzy-SMOTE usually generated
the smallest number of synthetic samples for datasets S1–S8. Overall, fuzzy-SMOTE is superior to SMOTE and Borderline-SMOTE because it improves $acc_\text{c}$ the most while using the fewest synthetic samples. Combining the evaluation results from Tables 3–7, for most datasets, Borderline-SMOTE outperforms SMOTE because it improves $acc_\text{c}$ while requiring fewer synthetic samples. These results may have occurred

Table 3. The number of synthetic samples generated by several oversampling methods.

| Datasets | Original IR | The number of generated synthetic samples |
|----------|-------------|------------------------------------------|
|          | None | SMOTE | Borderline-SMOTE | Fuzzy-SMOTE |
| S1       | 1.82 | 0 | 505 | 566 | 507 |
| S2       | 2.97 | 0 | 619 | 662 | 599 |
| S3       | 5.11 | 0 | 897 | 744 | 626 |
| S4       | 8.48 | 0 | 858 | 833 | 699 |
| S5       | 10.17 | 0 | 901 | 832 | 453 |
| S6       | 12.71 | 0 | 865 | 859 | 727 |
| S7       | 18.83 | 0 | 888 | 906 | 699 |
| S8       | 23.11 | 0 | 888 | 845 | 520 |
| S9       | 1.87 | 0 | 243 | 218 | 401 |
| S10      | 3.94 | 0 | 1048 | 1320 | 1703 |

IR: imbalance ratio; SMOTE: synthetic minority oversampling technique.
The bold values significance that the values are bigger when the methods are compared.

Table 4. $acc_\text{c}$ comparison of several oversampling methods.

| Datasets | $acc_\text{c}$ |
|----------|----------------|
|          | None | SMOTE | Borderline-SMOTE | Fuzzy-SMOTE |
| S1       | 0.9536 | 0.9607 | 0.9571 | 0.9625 |
| S2       | 0.9152 | 0.9385 | 0.9480 | 0.9445 |
| S3       | 0.9197 | 0.9450 | 0.9530 | 0.9550 |
| S4       | 0.8151 | 0.9076 | 0.9159 | 0.9242 |
| S5       | 0.8600 | 0.8900 | 0.9000 | 0.9200 |
| S6       | 0.7625 | 0.9000 | 0.8875 | 0.9000 |
| S7       | 0.7450 | 0.8800 | 0.9000 | 0.9050 |
| S8       | 0.8100 | 0.9267 | 0.9467 | 0.9633 |
| S9       | 0.5635 | 0.7199 | 0.7088 | 0.7803 |
| S10      | 0.5723 | 0.8692 | 0.9179 | 0.9641 |

SMOTE: synthetic minority oversampling technique.
The bold values significance that the values are bigger when the methods are compared.

Table 5. $acc_\text{c}$+ comparison of several oversampling methods.

| Datasets | $acc_\text{c}$+ |
|----------|----------------|
|          | None | SMOTE | Borderline-SMOTE | Fuzzy-SMOTE |
| S1       | 0.9656 | 0.9646 | 0.9638 | 0.9626 |
| S2       | 0.9695 | 0.9617 | 0.9503 | 0.9617 |
| S3       | 0.9725 | 0.9538 | 0.9549 | 0.9568 |
| S4       | 0.9832 | 0.9597 | 0.9686 | 0.9410 |
| S5       | 1.000 | 0.9754 | 0.9728 | 0.9715 |
| S6       | 0.9882 | 0.9489 | 0.9519 | 0.9479 |
| S7       | 0.9931 | 0.9489 | 0.9626 | 0.9459 |
| S8       | 0.9961 | 0.9823 | 0.9814 | 0.9823 |
| S9       | 0.8740 | 0.7640 | 0.6580 | 0.7800 |
| S10      | 0.9156 | 0.7684 | 0.5145 | 0.6964 |

SMOTE: synthetic minority oversampling technique.
The bold values significance that the values are bigger when the methods are compared.
because fuzzy-SMOTE pays particular attention to the minority examples in the fuzzy region—those most likely to be misclassified. In contrast, SMOTE treats each minority example equally, and all the minority examples have the same oversampling rate, while Borderline-SMOTE oversamples only the part of the minority examples that are near the classification borderline. Moreover, Borderline-SMOTE calculates nearest neighbors from the whole dataset rather than only from the minority class. Therefore, its computation cost and time increase as the dataset size increases.

Comparison with machine learning methods

The DTs C4.5, NB, SVM, and AdaBoost have all been applied as classifiers in imbalanced dataset experiments.\textsuperscript{10,35,36,56} In the experiments described in the previous subsection, we showed that combining fuzzy-SMOTE and SVM is better than SVM combined with other oversampling methods. In this section, we explore how well fuzzy-SMOTE works with other machine learning methods. The principle of NB is to calculate the posterior probability for each class of testing samples; the class with the highest posterior probability is the outcome of the prediction. C4.5 algorithm is constructed using a recursive partitioning operation that splits the examples into successive subsets based on information gain. SVM constructs a hyperplane to divide the sample space into two parts: a positive part and a negative part. AdaBoost learns a series of weak classifiers using the training dataset and then conjoins the weak classifiers to create a boosted classifier. This section compares the performances of these methods with fuzzy-SMOTE oversampling.

The performances of four basic machine learning methods with fuzzy-SMOTE are presented in Tables 8–11. First, comparing the classifier performances with and without fuzzy-SMOTE, we can conclude that these four machine learning methods achieve better performance on $\text{acc}_-$ and worse performance on $\text{acc}_+$ when using fuzzy-SMOTE. The greatest improvement of $\text{acc}_-$ occurred with fuzzy-SMOTE and SVM on S10; the improvement was 39.18% compared to the basic SVM method without fuzzy-SMOTE.
summary, fuzzy-SMOTE can improve the accuracy of classifiers on the minority class and their overall performance on $G\text{-}mean$, although the accuracy of the majority class is sacrificed. Comparing the machine learning methods, on the whole, NB had the worst performance. Without an oversampling method, AdaBoost outperforms SVM on $acc_{+}/C_{0}$, $F$-measure, and $G\text{-}mean$, while when using fuzzy-SMOTE, SVM outperforms AdaBoost.

### Conclusion

Previous works have paid little attention to the problem of imbalanced datasets in Android malware detection, where the number of malware examples is much smaller than that of benign examples; however, these imbalanced datasets can cause classifier decision boundaries to be biased toward the majority class. To solve this problem, we propose a new oversampling method called fuzzy-SMOTE, which is based on fuzzy set theory and the SMOTE method. Fuzzy-SMOTE generates synthetic examples for the minority class in the fuzzy region where the minority examples have low membership degrees and are more likely to be misclassified. Compared with traditional SMOTE and the Borderline-SMOTE methods, classifiers trained with datasets oversampled by fuzzy-SMOTE achieve better accuracy on the minority class and on the overall accuracy of the entire dataset.

Fuzzy-SMOTE improves classifier performances on imbalanced datasets because it broadens the minority decision boundary by generating additional minority class training examples. Based on the original class distribution of the training datasets, fuzzy-SMOTE calculates the membership degree of every minority example to the minority class. Then, it generates additional synthetic examples for those examples that have lower membership degrees. The new synthetic examples, which belong to the minority class, are then added to the original training dataset to create a new training set. When this new training set is used to train new classification models, the trained models pay more attention to the enhanced minority class. Consequently, the decision boundary of the minority class is enlarged and the classifier is no longer biased toward the majority class. The outcome is that classifiers trained with fuzzy-SMOTE classify more minority class examples correctly.
Table 10. The performances of C4.5 and C4.5 with fuzzy-SMOTE.

| Datasets | C4.5 | Fuzzy-SMOTE + C4.5 |
|----------|------|--------------------|
|          | acc+ | acc- | F-measure | G-mean | acc+ | acc- | F-measure | G-mean |
| S1       | 0.9705 | 0.9661 | 0.9758 | 0.9682 | 0.9459 | 0.9696 | 0.9639 | 0.9575 |
| S2       | 0.9725 | 0.9154 | 0.9720 | 0.9432 | 0.9400 | 0.9210 | 0.9550 | 0.9271 |
| S3       | 0.9734 | 0.9545 | 0.9821 | 0.9635 | 0.9606 | 0.9750 | 0.9773 | 0.9674 |
| S4       | 0.9784 | 0.9667 | 0.9812 | 0.9192 | 0.9450 | 0.8826 | 0.9648 | 0.9120 |
| S5       | 0.9853 | 0.9000 | 0.9877 | 0.9402 | 0.9489 | 0.9500 | 0.9712 | 0.9483 |
| S6       | 0.9852 | 0.9000 | 0.9886 | 0.9393 | 0.9390 | 0.9250 | 0.9656 | 0.9310 |
| S7       | 0.9901 | 0.8350 | 0.9916 | 0.9027 | 0.9390 | 0.9250 | 0.9670 | 0.9278 |
| S8       | 0.9892 | 0.9067 | 0.9921 | 0.9444 | 0.9744 | 0.9300 | 0.9850 | 0.9507 |
| S9       | 0.7900 | 0.5594 | 0.7793 | 0.6621 | 0.688 | 0.6496 | 0.7328 | 0.6667 |
| S10      | 0.8809 | 0.4664 | 0.8727 | 0.6237 | 0.7867 | 0.5773 | 0.8271 | 0.6660 |

SMOTE: synthetic minority oversampling technique.
The bold values significance that the values are bigger when the methods are compared.

Table 11. The performances of AdaBoost and AdaBoost with fuzzy-SMOTE.

| Datasets | AdaBoost | Fuzzy-SMOTE + AdaBoost |
|----------|----------|------------------------|
|          | acc+ | acc- | F-measure | G-mean | acc+ | acc- | F-measure | G-mean |
| S1       | 0.9676 | 0.9518 | 0.9705 | 0.9596 | 0.9410 | 0.9571 | 0.9580 | 0.9490 |
| S2       | 0.9715 | 0.9153 | 0.9715 | 0.9426 | 0.9469 | 0.9386 | 0.9625 | 0.9425 |
| S3       | 0.9823 | 0.8997 | 0.9813 | 0.9396 | 0.9617 | 0.9400 | 0.9745 | 0.9505 |
| S4       | 0.9794 | 0.8227 | 0.9793 | 0.8969 | 0.9400 | 0.9242 | 0.9646 | 0.9317 |
| S5       | 0.9902 | 0.8500 | 0.9877 | 0.9166 | 0.9705 | 0.9000 | 0.9801 | 0.9334 |
| S6       | 0.9774 | 0.8500 | 0.9827 | 0.9100 | 0.9528 | 0.9125 | 0.9724 | 0.9303 |
| S7       | 0.9902 | 0.8850 | 0.9921 | 0.9128 | 0.9548 | 0.8800 | 0.9744 | 0.9092 |
| S8       | 0.9951 | 0.8567 | 0.9936 | 0.9215 | 0.9813 | 0.9100 | 0.9880 | 0.9421 |
| S9       | 0.8460 | 0.5897 | 0.8179 | 0.7036 | 0.7050 | 0.7614 | 0.7702 | 0.7328 |
| S10      | 0.9156 | 0.5879 | 0.9063 | 0.7043 | 0.6709 | 0.8999 | 0.7754 | 0.7615 |

SMOTE: synthetic minority oversampling technique.
The bold values significance that the values are bigger when the methods are compared.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the National Natural Science Foundation of China (No. 61602052).

References
1. McAfee. McAfee lab threats report (fourth quarter), 2013. https://www.mcafee.com/in/resources/reports/rp-quarterly-threat-q4-2013.pdf
2. Zhou Y and Jiang X. Dissecting android malware: characterization and evolution. In: Proceedings of the 2012 IEEE symposium on security and privacy (SP), San Francisco, CA, 20–23 May 2012, pp.95–109. New York: IEEE.
3. Ali F, Nor BA, Rosli S, et al. A review on feature selection in mobile malware detection. Digit Invest 2015; 13: 22–37.
4. Aafer Y, Du W and Yin H. DroidAPIMiner: mining API-level features for robust malware detection in android. In: Proceedings of the international conference on security and privacy in communication systems, Sydney, NSW, Australia, 25–27 September 2013, pp.86–103. Cham: Springer.
5. Sanz B, Santos I, Laorden C, et al. Puma: permission usage to detect malware in android. In: Herrero A, Sanzczynski J and Stefanowski J. Neighbourhood sampling in bagging for imbalanced data. Source code.
6. Cen L, Gates CS, Si L, et al. A probabilistic discriminative model for android malware detection with decompiled source code. IEEE T Depend Secure 2015; 12(4): 400–412.
7. Blaszczyński J and Stefanowski J. Neighbourhood sampling in bagging for imbalanced data. Neurocomputing 2015; 150: 529–542.
8. Phung SL, Bouzerdoum A and Nguyen GH. Learning pattern classification tasks with imbalanced data sets. Pattern Recogn 2009: 193–208, http://ro.uow.edu.au/cgi/viewcontent.cgi?article=1806&context=infopapers
9. Gao M, Hong X, Chen S, et al. A combined SMOTE and PSO based RBF classifier for two-class imbalanced problems. Neurocomputing 2011; 74(17): 3456–3466.
10. Chawla NV, Bowyer KW, Hall LO, et al. SMOTE: synthetic minority over-sampling technique. *J Artif Intell Res* 2012; 16: 321–357.

11. Verbiest N, Ramentol E, Cornelis C, et al. Improving SMOTE with fuzzy rough prototype selection to detect noise in imbalanced classification data. In: *Proceedings of Ibero-American conference on artificial intelligence*, Cartagena, Colombia, 13–16 November 2012, pp.169–178. Berlin, Heidelberg: Springer.

12. García V, Sánchez J and Mollineda R. An empirical study of the behavior of classifiers on imbalanced and overlapped data sets. In: *Proceedings of the Iberoamerican congress on pattern recognition*, Viña del Mar, Chile, 13–16 November 2007, pp.397–406. Berlin, Heidelberg: Springer.

13. Prati RC, Batista GE and Monard MC. Class imbalances versus class overlapping: an analysis of a learning system behavior. In: *Proceedings of the Mexican international conference on artificial intelligence*, Mexico City, Mexico, 26–30 April 2004, pp.312–321. Berlin, Heidelberg: Springer.

14. Napierala K, Stefanowski J and Wilk S. Learning from imbalanced data in presence of noisy and borderline examples. In: *Proceedings of the international conference on rough sets and current trends in computing*, Warsaw, 28–30 June 2010, pp.158–167. Berlin, Heidelberg: Springer.

15. Kim DW, Lee KH and Lee D. On cluster validity index for estimation of the optimal number of fuzzy clusters. *Pattern Recogn* 2004; 37(10): 2009–2025.

16. Zimmermann HJ. Fuzzy set theory. *IEEE Trans. Syst. Man Cybern.* 1987; 17(2): 69–82.

17. Napierala K and Stefanowski J. BRACID: a comprehensive approach to learning rules from imbalanced data. *Intell Inf Syst* 2012; 39(2): 335–373.

18. Enck W, Ongtang M and McDaniel P. On lightweight mobile phone application certification. In: *Proceedings of the 16th ACM conference on computer and communications security*, Chicago, IL, 9–13 November 2009, pp.235–245. New York: ACM.

19. Gibler C, Crussell J, Erickson J, et al. AndroidLeaks: automatically detecting potential privacy leaks in android applications on a large scale. In: *Proceedings of international conference on trust and trustworthy computing*, Vienna, 13–15 June 2012, pp.291–307. Berlin, Heidelberg: Springer.

20. Felt AP, Chin E, Hanna S, et al. Android permissions demystified. In: *Proceedings of the 18th ACM conference on computer and communications security*, Chicago, IL, 17–21 October 2011, pp.627–638. New York: ACM.

21. Lin YD, Lai YC, Chen CH, et al. Identifying android malicious repackaged applications by thread-grained system call sequences. *Comput Secur* 2013; 39: 340–350.

22. Zheng M, Sun M and Lui JCS. Droid analytics: a signature based analytic system to collect, extract, analyze and associate android malware. In: *Proceedings of 2013 12th IEEE international conference on trust, security and privacy in computing and communications*, Melbourne, VIC, Australia, 16–18 July 2013, pp.163–171. New York: IEEE.

23. Chin E, Felt AP, Greenwood K, et al. Analyzing inter-application communication in android. In: *Proceedings of the 9th international conference on mobile systems, applications, and services*, Bethesda, MA, 28 June–1 July 2011, pp.239–252. New York: ACM.

24. Ham HS and Choi MJ. Analysis of android malware detection performance using machine learning classifiers. In: *Proceedings of 2013 IEEE international conference on ICT convergence (ICTC)*, Jeju, Korea, 14–16 October 2013, pp.490–495. New York: IEEE.

25. Shabtai A, Fleed Y and Elovici Y. Automated static code analysis for classifying android applications using machine learning. In: *Proceedings of 2010 international conference on computational intelligence and security (CIS)*, Beijing, China, 11–14 December 2010, pp.329–333. New York: IEEE.

26. Shabtai A, Kanonov U, Elovici Y, et al. “Andromaly”: a behavioral malware detection framework for android devices. *J Intell Inf Syst* 2012; 38(1): 161–190.

27. Burguera I, Zurutuza U and Nadjm-Tehrani S. Crowdroid: behavior-based malware detection system for android. In: *Proceedings of the 1st ACM workshop on security and privacy in smartphones and mobile devices*, Chicago, IL, 17 October 2011, pp.15–26. New York: ACM.

28. Zhang M, Duan Y, Yin H, et al. Semantics-aware android malware classification using weighted contextual API dependency graphs. In: *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, Scottsdale, AZ, 3–7 November 2014, pp.1105–1116. New York: ACM.

29. Enck W, Gilbert P, Han S, et al. TaintDroid: an information-flow tracking system for real time privacy monitoring on smartphones. *ACM T Comput Syst* 2014; 12(2): 5–19.

30. Dietz M, Shekhar S, Pisetsky Y, et al. QUIRE: lightweight provenance for smart phone operating systems. In: *Proceedings of USENIX security symposium*, San Francisco, CA, 8–12 August 2011, p.31. New York: ACM.

31. Felt AP, Wang HJ, Moshchuk A, et al. Permission redelegation: attacks and defenses. In: *Proceedings of USENIX security symposium*, San Francisco, CA, 8–12 August 2011, pp.12–16. New York: ACM.

32. Jang J, Kang H, Woo J, et al. Andro-AutoPsy: anti-malware system based on similarity matching of malware and malware creator-centric information. *Digit Invest* 2015; 14: 17–35.

33. Barandela R, Valdovinos RM, Sánchez JS, et al. The imbalanced training sample problem: under or over sampling? In: *Proceedings of joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)*, Lisbon, 18–20 August 2004, pp.806–814. Berlin, Heidelberg: Springer.

34. Kotssiantis S, Kanellopoulos D and Pintelas P. Handling imbalanced datasets: a review. *GESTS Int Trans Comput Sci Eng* 2006; 30(1): 25–36.

35. Han H, Wang WY and Mao BH. Borderline-SMOTE: a new over-sampling method in imbalanced data sets...
learning. In: Proceedings of the international conference on intelligent computing, Heifei, China, 23–26 August 2005, pp.878–887. Berlin, Heidelberg: Springer.
36. Chawla NV, Lazarevic A, Hall LO, et al. SMOTEBoost: improving prediction of the minority class in boosting. In: Proceedings of European conference on principles of data mining and knowledge discovery, Cavtat, Croatia, 22–26 September 2003, pp.107–119. Berlin, Heidelberg: Springer.
37. Bunkhumpornpat C, Sinapiromsaran K and Lursinsap C. Safe-level-SMOTE: safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In: Proceedings of Pacific-Asia conference on knowledge discovery and data mining, Bangkok, 27–30 April 2009, pp.475–482. Berlin, Heidelberg: Springer.
38. Guo H and Viktor HL. Learning from imbalanced data sets with boosting and data generation: the databoost-1M approach. ACM SIGKDD Explor Newsletter 2004; 6(1): 30–39.
39. Qiong GU, Ming WX, Zhao WU, et al. An improved SMOTE algorithm based on genetic algorithm for imbalanced data classification. J Digit Inform Manag 2016; 14(2): 93–103.
40. Verbiest N, Ramentol E, Cornelis C, et al. Preprocessing noisy imbalanced datasets using SMOTE enhanced with fuzzy rough prototype selection. Appl Soft Comput 2014; 14: 511–517.
41. Peng L, Zhang H, Yang B, et al. SMOTE-DGC: an imbalanced learning approach of data gravitation based classification. In: Proceedings of International conference on intelligent computing, Lanzhou, China, 2–5 August 2016, pp.33–144. Cham: Springer.
42. Wang KJ, Makond B, Chen KH, et al. A hybrid classifier combining SMOTE with PSO to estimate 5-year survivalability of breast cancer patients. Appl Soft Comput 2014; 20: 15–24.
43. Hambali M and Gbolagade M. Ovarian cancer classification using hybrid synthetic minority over-sampling technique and neural network. J Adv Comput Res 2016; 7: 109–124.
44. Tahir MA, Kitterl J and Yan F. Inverse random undersampling for class imbalance problem and its application to multi-label classification. Pattern Recogn 2015; 45(10): 3738–3750.
45. Liu XY, Wu J and Zhou ZH. Exploratory undersampling for class-imbalance learning. IEEE T Syst Man Cy B 2015; 39(2): 539–550.
46. He H and Garcia EA. Learning from imbalanced data. IEEE T Knowl Data En 2009; 21(9): 1263–1284.
47. Antonelli M, Ducange P and Marcelloni F. An experimental study on evolutionary fuzzy classifiers designed for managing imbalanced datasets. Neurocomputing 2014; 146: 125–136.
48. Elkan C. The foundations of cost-sensitive learning. In: Proceedings of the international joint conference on artificial intelligence, Seattle, WA, 4–10 August 2001, vol. 17, pp.973–978. New York: ACM.
49. Sun Y, Kamel MS, Wong AKC, et al. Cost-sensitive boosting for classification of imbalanced data. Pattern Recogn 2007; 40(12): 3358–3378.
50. Song J, Lu X and Wu X. An improved AdaBoost algorithm for unbalanced classification data. In: Proceedings of the sixth international conference on fuzzy systems and knowledge discovery, Tianjin, China, 14–16 August 2009, pp.109–113. New York: IEEE.
51. Drummond C and Holte RC. Exploiting the cost (in) sensitivity of decision tree splitting criteria. In: Proceedings of the seventeenth international conference on machine learning (ICML), Stanford, CA, 29 June–2 July 2000, pp.239–246. New York: ACM.
52. Kukar M and Kononenko I. Cost-sensitive learning with neural networks. In: Proceedings of the European conference on artificial intelligence (ECAI), Brighton, UK, 23–28 August 1998, pp.445–449. New York: Wiley.
53. Akbani R, Kwek S and Japkowicz N. Applying support vector machines to imbalanced datasets. In: Proceedings of European conference on machine learning, Pisa, 20–24 September 2004, pp.39–50. Berlin, Heidelberg: Springer.
54. Tang Y, Zhang YQ, Chawla NV, et al. SVMs modeling for highly imbalanced classification. IEEE T Syst Man Cy B 2009; 39(1): 281–288.
55. Zhou ZH and Liu XY. Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE T Knowl Data En 2006; 18(1): 63–77.
56. Jian C, Gao J and Ao Y. A new sampling method for classifying imbalanced data based on support vector machine ensemble. Neurocomputing 2016; 193: 115–122.
57. Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th international joint conference on artificial intelligence (IJCAI), Montreal, QC, Canada, 20–25 August 1995, vol. 14, pp.1137–1145. New York: ACM.
58. Cross validation, http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation
59. Maciejewski T and Stefanowski J. Local neighbourhood extension of SMOTE for mining imbalanced data. In: Proceedings of 2011 IEEE symposium on computational intelligence and data mining (CIDM), Paris, 11–15 April 2011, pp.104–111. New York: IEEE.
60. ISCX android botnet dataset, http://www.unb.ca/research/isxcs/dataset/isxcs-android-botnet-dataset.html
61. Kadir AFA and Stakhanova N Ghorbani AA. Android permissions and applications. In: Proceedings of the 17th ACM symposium on access control models and technologies, New York, 3–5 November 2015, pp.78–91. Cham: Springer.
62. UCI machine learning dataset, http://archive.ics.uci.edu/ml/datasets.html
63. Sarma BP, Li N, Gates C, et al. Android permissions: a perspective combining risks and benefits. In: Proceedings of the 17th ACM symposium on access control models and technologies, Newark, NJ, 20–22 June 2012, pp.13–22. New York: ACM.
64. Wang Y, Zheng J, Sun C, et al. Quantitative security risk assessment of android permissions and applications. In: Proceedings of IFIP annual conference on data and applications security and privacy, Newark, NJ, 15–17 July 2013, pp.226–241. Berlin, Heidelberg: Springer.
65. Yerima SY, Sezer S and McWilliams G. Analysis of Bayesian classification-based approaches for Android malware detection. IET Inform Secur 2014; 8(1): 25–36.