Fast-RCM: Fast Tree-Based Unsupervised Rare-Class Mining

Haiqin Weng, Shouling Ji, *Member, IEEE*, Changchang Liu, Ting Wang, Qinming He, and Jianhai Chen, *Member, IEEE*

***Abstract*—Rare classes are usually hidden in an imbalanced dataset with the majority of the data examples from major classes. Rare-class mining (RCM) aims at extracting all the data examples belonging to rare classes. Most of the existing approaches for RCM require a certain amount of labeled data examples as input. However, they are ineffective in practice since requesting label information from domain experts is time consuming and human-labor extensive. Thus, we investigate the unsupervised RCM problem, which to the best of our knowledge is the first such attempt. To this end, we propose an efficient algorithm called Fast-RCM for unsupervised RCM, which has an approximately linear time complexity with respect to data size and data dimensionality. Given an unlabeled dataset, Fast-RCM mines out the rare class by first building a rare tree for the input dataset and then extracting data examples of the rare classes based on this rare tree. Compared with the existing approaches which have quadric or even cubic time complexity, Fast-RCM is much faster and can be extended to large-scale datasets. The experimental evaluation on both synthetic and real-world datasets demonstrate that our algorithm can effectively and efficiently extract the rare classes from an unlabeled dataset under the unsupervised settings, and is approximately five times faster than that of the state-of-the-art methods.**

***Index Terms*—Clustering methods, data mining, tree data structures.**

Manuscript received February 20, 2019; revised April 27, 2019; accepted June 11, 2019. This work was supported in part by NSFC under Grant 61772466, Grant U1836202, and Grant 61472359, in part by the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under Grant LR19F020003, in part by the Provincial Key Research and Development Program of Zhejiang, China, under Grant 2017C01055, and in part by the Alibaba-ZJU Joint Research Institute of Frontier Technologies. This paper was recommended by Associate Editor A. F. S. Gomez. *(Corresponding author: Jianhai Chen.)*

H. Weng, Q. He, and J. Chen are with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China (e-mail: hq_weng@zju.edu.cn; hqm@zju.edu.cn; chenjh919@zju.edu.cn).

S. Ji is with the Institute of Cyberspace Research and College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China, and also with the Alibaba-Zhejiang University Joint Institute of Frontier Technologies, Zhejiang University, Hangzhou 310027, China (e-mail: sji@zju.edu.cn).

C. Liu is with the Department of Distributed AI, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: changchang.liu33@ibm.com).

T. Wang is with the Department of Computer Science, Lehigh University, Bethlehem, PA 18015 USA (e-mail: inbox.ting@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2019.2924804

I. Introduction

THE CONCEPT of rare class was first proposed by Pelleg and Moore [1], where rare classes were defined as tiny clusters of similar anomalies. For example, in finance, a rare class usually represents a small fraction of fraud transactions generated from the same template [2]. As time goes on, rare classes are not limited to similar anomalies [2]–[5]. All the interesting data examples from undiscovered classes can be regarded as rare classes. For instance, in medical diagnosis, a rare class can be a new kind of disease which is unknown to the current doctors [3].

In many real-world applications, the proportion of data examples in different classes is highly skewed: major classes dominate the dataset, and rare classes may have only a few examples. However, we are usually more interested in rare classes since analyzing rare classes can help us to understand the underlying patterns of those real-world applications. For example, finding out the fraud transactions generated from the same malicious template helps users to analyze the security leak of a financial system. Hence, rare-class mining (RCM) is proposed for discovering the data examples of the rare classes from a given dataset [2], [4]–[8]. The difference between RCM and imbalance learning is that: RCM mainly focuses on discovering the data examples of the rare classes, while imbalanced classification trains a classifier over the entire dataset. Traditional RCM techniques require label information, and are normally carried out by two semisupervised subtasks [2], [4]–[7]: 1) rare-class detection (RCD) and 2) rare-class exploration (RCE). RCD aims at identifying at least one data example of a rare class to provide evidence for its existence with certain confidence by using a labeling Oracle [2], [6], [7], [9], [10]. RCE aims at finding out the remaining data examples of a rare class by using at least one labeled data example of that class as a seed [4], [5]. Both of the two subtasks require label information, which is time consuming and human-labor extensive for real-world applications. The state-of-the-art RCD approaches [2], [6], [7] and RCE methods [4], [5] are hardly to work properly if no label information is available [11].

To address the above challenge, we therefore investigate RCM under the unsupervised setting (unsupervised RCM for short), which to the best of our knowledge is the first such attempt. Unsupervised RCM is particularly useful in real-world applications since it has two advantages.

1) It can avoid potential mismatch between true data labels and their corresponding labels predicted by human

experts. For example, in medical diagnosis, each of the diseases is very complicated and hard to be diagnosed perfectly even for an experienced doctor. In this case, traditional semisupervised RCM may not work while in contrast unsupervised RCM is possible to detect a rare class of diseases which has the potential to help doctors find a new disease.

2) It will reduce the time-consuming process of label collection especially for those involving online data sources. Furthermore, unsupervised RCM is the only technique that can be applied for RCM when no labeled data is available.

In this paper, we propose an efficient algorithm Fast-RCM for unsupervised RCM, which has an approximately linear time complexity with respect to the dataset size and dimensionality. Compared with the existing approaches which have quadric or even cubic time complexity [4], [12]–[14], Fast-RCM is much faster in runtime behavior. Given a dataset, Fast-RCM works as follows. It first builds a rare tree for this dataset, which reorganizes the data for capturing the characteristics of the rare classes. Then, it extracts rare classes leveraging the searching results of the data over this rare tree. The rare tree is a specific binary tree, which reduces the runtime complexity into approximately linear. Moreover, the rare tree can also be used as a density estimator for imbalanced datasets. Experimental results show that Fast-RCM can effectively extract the rare classes under the unsupervised setting and is approximately five times faster than that of the state-of-the-art methods.

Our contributions are summarized as follows.

1) We provide a novel rare tree for characterizing the structures of an imbalanced dataset, which can also be utilized to select rare class candidates as well as estimating the density of input data. Further, we theoretically analyze the performance of the rare tree as a density estimator. We find that the deviation between the estimated and true density of input data is reduced by our rare tree construction process, and the variance of the estimated density decreases with larger size of the input data.
2) Based on the rare tree, we propose an unsupervised RCM technique, called Fast-RCM, which to our best knowledge is the first such attempt. Compared with the state-of-the-art approaches, Fast-RCM is much faster and can be applicable to large-scale datasets.
3) Finally, we apply Fast-RCM on two real-word datasets—CALLS and DSAA—to discover the interesting patterns of rare classes hidden in them. CALLS contains three kinds of emergency calls from the Montgomery County, and DSAA contains the course records of the registered students on an online course platform. On these datasets, Fast-RCM achieves five times faster than that of the state-of-the-art supervised methods and discovers the rare class of a natural disaster happening in Montgomery County and the rare class of the certificated students who pass the final examination.

The remaining sections are organized as follows. We review the related work in Section II and present the preliminaries in Section III. In Section IV, we first introduce the overview of our Fast-RCM algorithm, and then we present the two main steps of Fast-RCM, that is, rare tree construction and rare-class extraction. Especially, in Section IV-A4, we theoretically analyze the upper bound of the rare tree as a density estimator of the input dataset. Finally, we show the experimental results in Section V and conclude this paper in Section VI.

II. Related Work

Generally, previous work of RCM can be classified into three categories: 1) anomaly detection in RCM; 2) RCD; and 3) RCE.

A. Anomaly Detection in RCM

Anomaly detection refers to the problem of finding patterns in given data that do not conform to the expected behavior [15]. Over time, a variety of anomaly detection techniques have been developed [13], [14], [16]–[25], and most of which are specifically designed for certain application domains while the others are for the more general cases. These detection techniques can be further classified into four categories: 1) classification-based schemes [16]–[20]; 2) clustering-based schemes [14], [21], [22]; 3) nearest neighbors-based schemes [13], [23]; and 4) statistical schemes [24], [25]. A thread of the anomaly detection techniques can be applied into unsupervised RCM by returning data examples that are identified as a cluster of anomalies [13], [14], [21]. Nonetheless, since these detection techniques are not specifically designed for the application domain of unsupervised RCM, they are usually hard to capture the fine-grained characteristics of the rare classes and thus cannot perform well in practice.

Among these detection techniques, the most related work to ours is isolation forest (IF) [14], which is a clustering-based method. IF builds an ensemble of *iTrees* for a given dataset, then reports data examples that have short average path lengths on the *iTrees* as anomalies. Compared with IF, our Fast-RCM, has two advantages: 1) Fast-RCM builds only one tree for the input dataset, making it much faster and 2) Fast-RCM builds the rare tree based on the compactness characteristics of the rare classes, making it more accurate. Except for this paper, He *et al.* extended the one-class SVM [19] to semisupervised RCM in [12], where they tried to enclose a rare class with a minimum-radius hyperball by using a small training set.

B. Rare-Class Detection

RCD aims at identifying at least one data example of a rare class to provide evidence for its existence by using a labeling Oracle [1], [3], [7]–[10], [26]–[28]. Since RCD can find at least one data example for a rare class, it is often followed by an RCE process to discover the remaining data examples of this rare class. The problem of RCD is first formalized by Pelleg and Moore [1], where they introduced a novel active-learning scenario in which a user wants to work with a learning algorithm to identify useful anomalies. To solve this problem, they proposed a mix-model-based algorithm *interleave* for identifying rare class examples existing in tiny classes of similar anomalies. Methods proposed so far can be classified into three categories: 1) the mixture-model-based [1], [10]; 2) the

nearest-neighborhood-based [3], [8], [9]; and 3) the density-based [7], [26]–[29]. Most recently, Wang *et al.* [9] presented a k-nearest-neighbors-based RCD algorithm along with an automatically selection strategy for k. Tu *et al.* [10] proposed a mixed-model-based novel RCD framework, which combined active learning and hierarchical density-based clustering to find initial data examples of the rare classes.

C. Rare-Class Exploration

With one or more labeled data examples as initializing seeds, RCE targets on discovering the remaining data examples belonging to the same rare class of those seeds. Recently, there have been a few algorithms proposed for RCE. He *et al.* [12] for the first time proposed an optimization framework to discover the rare class by using a small training set. Huang *et al.* [4] proposed an RCE algorithm FRANK which transformed the problem of RCE to a local community detection problem. Liu *et al.* [5] proposed an effective and efficient algorithm known as FREE to explore rare classes of arbitrary shapes. Recently, Liu *et al.* [30] proposed a novel approach called RCEWA, which achieves a higher F-score compared with the existing algorithms.

D. Remarks

In summary, the anomaly detection techniques [13], [14], [21] cannot perform well for our applications since they are hard to capture fine-grained characteristics of a rare class. Both the state-of-the-art RCD approaches [7], [8] and RCE methods [4], [5] are ineffective in practice since it can be time consuming to request label information from domain experts. Moreover, these algorithms are hardly to work properly if no label information is available [11]. To address the limitations of existing work, we propose Fast-RCM for unsupervised RCM in this paper.

III. Preliminaries

In this section, we first formulate the problem of unsupervised RCM. Then, we give several assumptions of rare classes, which motivate our design of Fast-RCM. Finally, we briefly introduce the dip test used for rare tree construction.

A. Problem Formulation and Assumptions

Let $L = \{1, 2, \ldots, m_1, \ldots, m\}$ denote the set of m distinct classes, where the first m_1 classes are rare classes and the remaining are major classes. Let $X = \{X_i | i = 1, 2, \ldots, n\} \subset R^d$ denote the set of unlabeled data examples, where d is the dimensionality of each X_i, and $Y = \{Y_i | i = 1, 2, \ldots, n\}$ denote the set of the true labels of X, which is usually not available in practice.

The unsupervised RCM is formulated as follows.

Definition 1 (Unsupervised RCM): Given an unlabeled dataset X, which come from the class set L, that is, $Y_i \in L$, unsupervised RCM aims to extract the set of data examples from X that belong to the rare classes, that is, $\mathbb{R} = \{X_i | \exists X_i \in X, Y_i \leq m_1\}$.

Usually, the number of data examples in the rare class is extremely small following the assumption that are commonly used by the existing work [1], [3], [4], [7], [8], [12], [26], [31], [32]. Different from both RCD and RCE, unsupervised RCM neither requests any label information from a labeling Oracle nor requires labeled data examples as its input. Unsupervised RCM can find out the rare classes in practice, and further help us to understand the underlying patterns contained in those rare classes. Again, let us consider the dataset of financial transactions collected from a financial system as an example. In this scenario, unsupervised RCM is expected to discover all the fraud transactions without requiring the help from security experts. These discovered fraud transactions further help us to analyze the potential vulnerabilities of the financial system.

For each of the rare classes, we adopt the following assumptions which are commonly used in existing research [1], [3], [4], [7], [8], [12], [26], [31], [32].

Assumption 1 (Compactness): The data examples of a rare class form a tiny and compact cluster on partial or the whole dimensions in the feature space.

This *compactness* assumption was first proposed by Pelleg and Moore [1], where a rare class exists within tiny clusters of similar anomalies. In recent work [3], [4], [8], all the compact clusters of interesting data examples from undiscovered classes are regarded as rare classes. In addition to the *compactness* assumption, the existing work [1], [3], [4], [8] also adopted the *isolation* assumption where a rare class is isolated from the major classes. To relax this assumption and make our analysis generalizable, we assume that a rare class can be overlapped with the major classes.

The *compactness* assumption implies that rare classes are tightly grouped on partial or the whole feature space, and therefore their data distributions form local maximums. A mode of a data distribution is considered to be any value where the probability density function (*pdf*) has a local maximum [33]. Based on this analysis, each of the rare classes can be seen as a mode on partial dimensions or the whole feature space, as shown below.

Assumption 2 (Unimodality): Each of the rare class corresponds to one mode on partial dimensions or on the whole feature space.

Maurus and Plant [32] adopted a more strict *unimodality* assumption where each cluster in the feature space forms a unimodal shape on all coordinate directions. Under this *unimodality* assumption, we can use the dip test [34] to recursively choose the split value for the rare tree, which will be described in detail in Section IV.

B. Dip Test

The dip test [34] is a formal statistical test that measures the unimodality of a data distribution on one dimension. In this paper, we exploit the dip test to discover the presence of modes in the data distribution of an input dataset.

Given an unlabeled dataset $X \subset R^d$, let $X^j = \{X_i^j | i = 1, 2, \ldots, n\}$ be the vertical projection of X on its jth dimension. For each X^j, let F_{X^j} be its empirical cumulative distribution

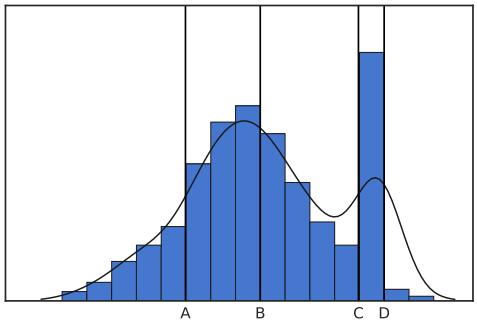


Fig. 1. Dip test.

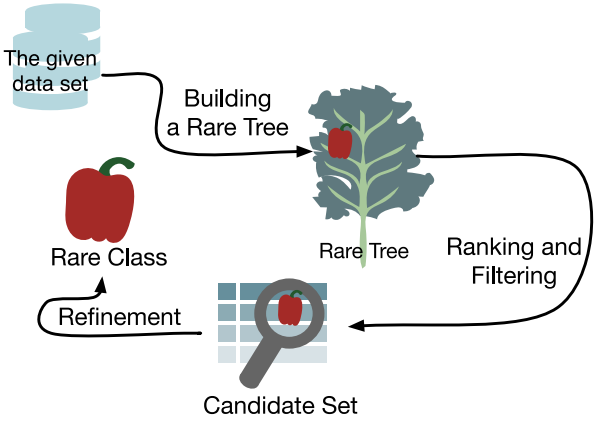


Fig. 2. Overview of Fast-RCM.

function, and G_{X^j} be the optimal piecewise-linear function that is used to fit F_{X^j}.

Mechanism of Dip Test: To measure the unimodality of X^j, the dip test uses a common definition of unimodality: the distribution of X^j is unimodal if F_{X^j} takes a convex form up to its mode and a concave form after the mode. Based on this definition, the dip test tries to fit some piecewise-linear function G_{X^j} on F_{X^j}. Then, dip test measures the unimodality of X^j as the maximum difference between G_{X^j} and F_{X^j}. Note that since a group of piecewise-linear functions can be used to fit F_{X^j}, the dip test selects the one that has the minimum–maximum distance as the optimal fit function. In summary, the dip test measures the unimodality of X^j as the maximum difference between F_{X^j} and the unimodal distribution function that minimizes the maximum difference.

For simplicity, the dip test tries to assert that F_{X^j} is unimodal by fitting a piecewise-linear function on it. Then, it gives us a measure for F_{X^j}'s departure from unimodality. The farther F_{X^j} stays from unimodality, the larger value of this measure. Consider X^j with two modes as an example. In this case, the multimodal distribution of X^j makes G_{X^j} stay far from F_{X^j}. Therefore, the dip test reports a large value of this measure for X^j. More importantly, the dip test gives the locations of the largest mode: the left and right interval of the mode.

Now, for each X^j, we define p as the measure of its unimodality. We further denote x_L and x_R as the left and right interval, respectively, of the largest mode of X^j. According to Hartigan and Hartigan [34], the dip test is defined as follows.

Definition 2 (Dip Test): Given a dataset $X \subset R^d$, the dip test of X on its jth dimension is defined as

$$x_L, x_R, p = \text{diptest}(X^j).$$

Here, diptest(X^j) can be utilized to: 1) show whether X^j is unimodal shaped or not. Specifically, a small value of p indicates that X^j is unimodal shaped and a large value of p indicates that X^j has more than one mode and 2) detect the locations of the largest mode on the jth dimension, that is, x_L and x_R. Let us consider an example in Fig. 1 which shows the histogram and density distribution of a 1-D dataset consisting of one rare class and one major class. The dip test here shows that the distribution of this dataset is multimodal and detects the largest mode ranging from C to D.

IV. Fast Rare-Class Mining

In this section, we first introduce the overview of Fast-RCM, and then describe the detailed steps of Fast-RCM. As aforementioned, Fast-RCM consists of two key steps: 1) rare tree construction and 2) rare-class extraction. The details of these two steps are described in Sections IV-A and IV-B, respectively.

The *compactness* and *unimodality* assumptions of a rare class imply that the compact cluster of a rare class usually corresponds to the most significant mode in partial or the whole dimension. Take Fig. 1 as an example again. The rare class corresponds to the most significant mode ranging from C to D while the part of the major class corresponds to another mode ranging from A to B. Moreover, the dip test can tell that the dataset illustrated in Fig. 1 is multimodal shaped and can find the range of the most significant mode of the rare class. In other words, the dip test provides some guidance to help us find the coarse shape of the rare class.

Fast-RCM Overview: Based on the above observation and analysis, we come up with the idea of Fast-RCM, as illustrated in Fig. 2. Fast-RCM consists of two main steps: 1) rare tree construction and 2) rare-class extraction.

For the rare tree construction step, Fast-RCM builds a specific binary tree, called rare tree, to reorganize the given dataset and to be used further to find the coarse shape of the rare class. Specifically, in the tree construction process, we utilize the dip test to decide that whether the input dataset should be split or not and where to be split. As shown in Fig. 2, the rare class is then reorganized into a *red apple* hidden in the rare tree.

For the rare-class extraction step, we use the constructed rare class as a ranking tool to assign a score for each example. In the constructed rare tree, data examples of the rare class have two dominate features: 1) they have a short traversing path because they are separated from the rest in the previous building process and 2) they usually fall into leaf nodes with high density. Thus, Fast-RCM uses the combination of these two features to discriminate the data examples of the rare class from the major ones. Fast-RCM first ranks each data example using their traversing results computed over the rare tree. Then, it selects a few data examples with high scores as a candidate set of the rare class. In the end, it refines the candidate set to achieve high accuracy for detecting the rare class.

Algorithm 1 RareTuple

1: **Input:** The unlabelled dataset X
2: **Output:** The rare tuple of the input dataset
3: Initialise $D \leftarrow \{1, \ldots, d\}$
4: Initialise $p \leftarrow +\infty$
5: **while** $p > 10^{-10}$ *and* D is not empty **do**
6: $\quad t \leftarrow \arg\max_{j \in D} \big(max(X^j) - min(X^j)\big)$
7: $\quad D \leftarrow D \backslash \{t\}$
8: $\quad x_L, x_R, p \leftarrow \text{diptest}(\mathbb{X}, t)$
9: **end while**
10: $split\_dim \leftarrow t$
11: $L \leftarrow \left\{X_i | \forall X_i \in X, X_i^t \leq x_L\right\}$
12: $R \leftarrow \left\{X_i | \forall X_i \in X, X_i^t \geqslant x_R\right\}$
13: **if** $|L| > |R|$ **then**
14: $\quad split\_value \leftarrow x_L$
15: **else**
16: $\quad split\_value \leftarrow x_R$
17: **end if**

Algorithm 2 RareTree

1: **Input:** The unlabelled dataset X
2: **Output:** The Rare Tree T
3: $T \leftarrow \emptyset$ ▷ T is an static variable and initialized only once
4: **if** $|X| < \theta$ **then**
5: $\quad \ell \leftarrow$ new LeafNode ▷ Generate a leaf node
6: $\quad T \leftarrow T \cup \ell$
7: $\quad \ell.size \leftarrow |X|$, $\ell.density \leftarrow p(\ell)$
8: $\quad$ **Return** ℓ
9: **else**
10: $\quad \ell \leftarrow$ new InnerNode ▷ Generate an inner node
11: $\quad T \leftarrow T \cap \ell$
12: $\quad (j, v) \leftarrow \text{RareTuple}(X)$
13: $\quad L \leftarrow \left\{X_i | \forall X_i \in X, X_i^j \leq v\right\}$, $R \leftarrow X \backslash L$
14: $\quad \ell.rare\_tuple \leftarrow (j, v)$
15: $\quad \ell.left \leftarrow \text{RareTree}(L)$
16: $\quad \ell.right \leftarrow \text{RareTree}(R)$
17: $\quad$ **Return** ℓ
18: **end if**

A. Rare Tree Construction

Given an unlabeled dataset X, for building a rare tree over X, we recursively divide X by selecting a split dimension i and a split value v until the size of the input dataset is less than a threshold. For simplicity, we define a *rare tuple* consisting of both the split dimension and value, denoted as (i, v). In the following text, we first describe how to select a rare tuple, and then present detailed steps for building a rare tree. In the end of this section, we explore the several interesting characteristics of the rare tree.

1) Choosing Rare Tuple: Given a dataset $X \subset R^d$, let $D = \{1, 2, \ldots, d\}$ denote the set of dimension indicators of X. Let $|\cdot|$ denote the number of data examples in a dataset, for example, $|X|$ denotes the number of data examples in X. Now, following the *compactness* and *unimodality* assumptions of a rare class, we choose a rare tuple by using the dip test in Definition 2. Algorithm 1 shows the detailed steps of rare tuple selection.

Algorithm 1 works as follows.

1) Choose a split dimension using a while loop (lines 5–9). This while loop is to search such a dimension that has a large domain range and at least one mode. In this while loop, we first choose a dimension t with the largest domain range as the split dimension, that is, $t = \arg\max_{j \in D}(\max(X^j) - \min(X^j))$. We implement the dip test of X on the chosen dimension t. Then, p-value is reported by the dip test for the unimodality measure of X^t, and the locations of the largest mode in X^t. As a commonly used strategy in the dip test [32], we consider that X^t is multimodal shaped (has more than one mode) if its p-value is larger than 10^{-10}. This process continues until the p-value returned by the dip test is larger than 10^{-10} or each dimension has been tested.
2) Choose a split value (lines 11–17). As mentioned earlier, the dip test returns the left and right boundaries of the largest mode, that is, x_L and x_R. From these boundaries, we select the one that can split the input dataset more equally as the split value. We use this selection strategy because a more balanced binary tree can reduce the time complexity of searching over this tree. For example, in Fig. 1, we select C from the locations of the largest mode $[C, D]$ as the split value.

2) Rare Tree Construction: After choosing the split dimension and value according to Algorithm 1, we now describe how to build a rare tree. The rare tree is constructed by recursively splitting the input dataset until the size of the dataset is less than a threshold θ. Let $T = \{T_i | i = 1, 2, \ldots\}$ denote the rare tree, where T_i is either a leaf node with no child or an inner node with two children. For each T_i, let $\mathcal{C}_{T_i} = \{X_i | \forall X_j \in X, X_j$ falls in T_i during the split process$\}$ denote the set of data examples in T_i. Before introducing how to build a rare tree, we first show the definition of node density as follows.

Definition 3 (Node Density): Given a rare tree T and a node $T_i \in T$, the node density of T_i, denoted as $p(T_i)$, is defined as follows:

$$p(T_i) = \frac{\left|\mathcal{C}_{T_i}\right|}{nV(T_i)} \tag{1}$$

where $|\mathcal{C}_{T_i}|$ denotes the number of data examples in the node, or called *node size*, and $V(T_i)$ denotes the node volume of the hyper-rectangle that T_i represents.

Algorithm 2 presents the pseudocode for rare tree construction. Given a dataset X, we recursively split the input dataset using a rare tuple (i, v) until the termination condition, $|X| < \theta$, is satisfied, as shown in line 3. In lines 4–7, we stop the split process and generate a leaf node using the current dataset. In lines 9–15, we continue to split the input dataset as follows. We first calculate the rare tuple according to Algorithm 1. Then, we split the current node using this rare tuple into two subdatasets. The depth of a rare tree depends

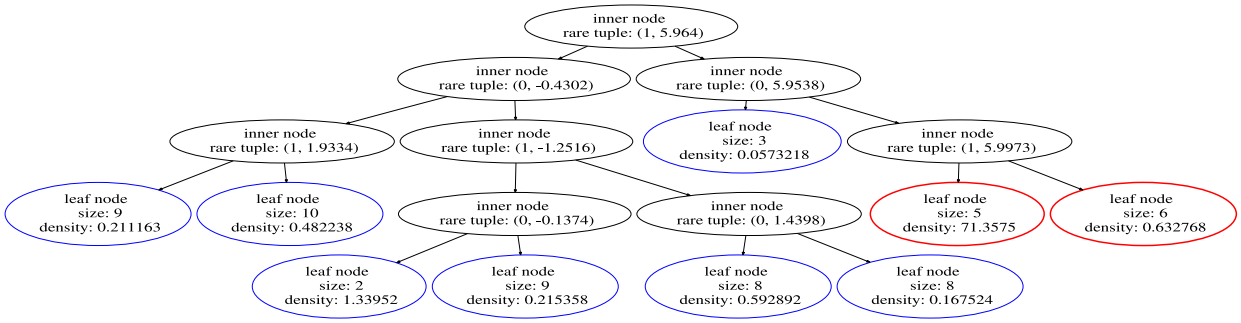


Fig. 3. Internal structure of a rare tree.

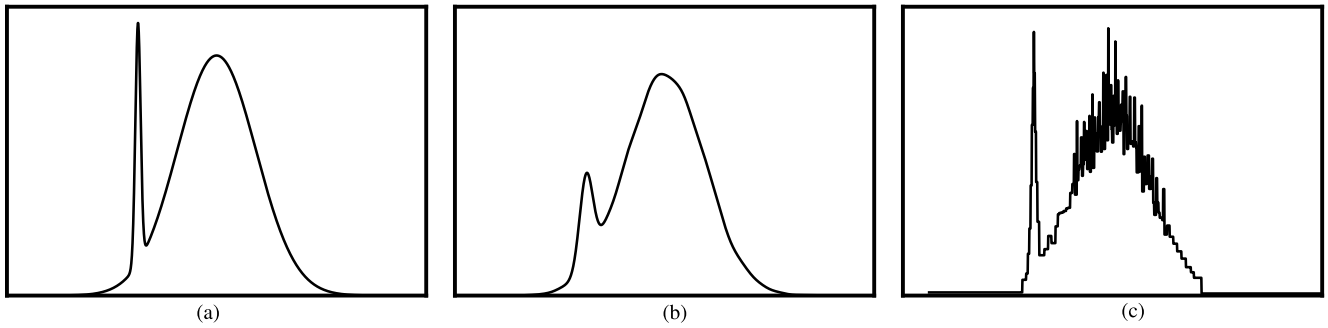


Fig. 4. Density estimation. (a) True. (b) KDE. (c) Fast-RCM.

on the number of data examples and the dimensionality of the dataset. Usually, the depth of the rare tree is larger than six.

The goal of the rare tree construction is to reorganize all the data examples of the same rare class into the same leaf node. The threshold θ should be larger than the size of a rare class. This guarantees that all the data examples of a rare class can be split into the same leaf node. Since rare classes are usually hidden in a dataset with over 99% data examples from the major classes in practice, this threshold θ is empirically set based on the size of the input dataset according to $\theta = |X| \times 5\%$. Fig. 3 shows the internal structure of the rare tree built over a 2-D dataset which contains one rare class and one major class. In Fig. 3, the black circles represent the inner nodes, and the red and blue circles represent the leaf nodes. We can see that most of the data examples from the rare class fall into red circles with high density and short traversing path. Note that the two red circles have significantly different density since they come from different regions of the rare class: the one with larger density is from the centroid of the rare class while the other with smaller density is from the boundary of the rare class.

Our rare tree is different from the common decision tree from many perspectives. Common decision tree is used for supervised learning, including regression and classification while the rare tree is used for RCM under the unsupervised setting. The training procedure of a common decision tree requires learning strategies to minimize the training error, for example, GBDT [35] constructs an ensemble of weak decision trees through gradient boosting. Compared with the common decision tree, the rare tree leverages the dip test to select the split dimension and value and requires no training procedure.

3) Exploration of the Rare Tree: Until now, we have built the rare tree. In this section, we discuss some interesting characteristics of this rare tree.

Density Estimator: As a side function, the rare tree can be used as a density estimator for imbalanced datasets. Specifically, we use the density of leaf nodes to estimate the density over a single data example $X_i \in X$. The density estimation function over X_i is defined as follows.

Definition 4 (Estimated Density): Given a rare tree T and a data example $X_i \in X$, the estimated density of X_i, denoted as $\hat{f}(X_i)$, is defined as

$$\hat{f}(X_i) = \sum_{T_i \in T, T_i \text{ is a leaf node}} \delta(X_i \in \mathcal{C}_{T_i}) p(T_i) \tag{2}$$

where $\delta(\cdot)$ denotes the Kronecker delta function [36] that gives the value 1 when its argument is true and 0 otherwise.

Fig. 4 illustrates the effectiveness of a rare tree as a density estimator on a dataset of 26 000 data examples. In particular, Fig. 4(a) shows the true density of the dataset where the left peak represents the area of the major class and the right peak is the rare class. For the rare tree, we set $\gamma = n \times 5\%$ and $\theta = n \times 5\%$. Fig. 4(b) shows the density estimation results of the kernel density estimation (KDE) [37], and Fig. 4(c) shows the density estimation results of the rare tree. From Fig. 4(b), we observe that although KDE smoothly estimates the density, it performs a poorer estimation for the rare class than that of a rare tree as depicted in Fig. 4(c). Therefore, we conclude that our constructed rare tree can perform more accurate estimation than that of KDE in RCM.

Searching Behaviors: First, we give the definition of searching path length. The searching path length is measured by the length of the path starting from the root node to a leaf node

that terminates the searching process. We formally define the searching path length as follows.

Definition 5 (Path Length): Given a rare tree T and a data example $X_i \in X$, the path length of X_i, denoted as $h(X_i)$, is defined as

$$h(X_i) = \sum_{T_i \in T} \delta(X_i \in \mathcal{C}_{T_i}). \tag{3}$$

Then, we search all the data examples over the rare tree and explore their searching behaviors. Based on the *compactness* and *unimodality* assumptions that the data examples of a rare class form a mode on the feature space, we conclude that those data examples tend to be separated from the remaining in the previous split process by a rare tuple. Therefore, those data examples will have short searching path on average. Besides, the compact cluster of those data examples tends to be separated from the remaining as a whole. Hence, those data examples will fall into the leaf nodes with high estimated density. Based on such analysis, we summarize that the data examples of the rare classes have two dominating features: 1) high estimated density and 2) short searching path length.

Let us consider Fig. 3 as an example again. Most of the data examples from the rare class have path length 3 while most of the data examples from the major class have path length larger than 3. It is obvious that the data examples from the rare class have a shorter path length on average than that of those from the major class.

4) Analysis of the Rare Tree: The estimated density is of key importance for rare-class extraction since it is further used to select the candidate set of rare classes (will be introduced in Section IV-B for details). In this section, we theoretically measure the performance of the rare tree as a density estimator.

Given an unlabeled dataset X, let us denote $\hat{f}(\cdot)$ and $f(\cdot)$ as the estimated and true probability density of $X_i \in X$, respectively. Given the rare tree T built over X, we further denote $T(X_i) \in T$ as the leaf node that contains X_i as well as the region it represents.

To measure the performance of the rare tree, we use *mean squared error* denoted as e to quantify the deviation between $\hat{f}(X_i)$ and the true probability density $f(X_i)$, which is defined as follows:

$$e = \mathbb{E}\left[\left(\hat{f}(X_i) - f(X_i)\right)^2\right]. \tag{4}$$

Let us first simplify (4) as follows:

$$\begin{aligned} e &= \mathrm{Var}\left(\hat{f}(X_i)\right) + \left(\mathbb{E}\left[\hat{f}(X_i)\right] - f(X_i)\right)^2 \\ &= \sigma^2 + \beta^2. \end{aligned}$$

In the following text, we theoretically analyze σ^2 and β^2, respectively. Before analyzing them, we introduce the definition of the second-order Taylor approximation [38] around the leaf node.

Definition 6 (Taylor Approximation): Given $X_i \in X$ and the leaf node $T(X_i)$ where X_i falls, the second-order Taylor approximation of $f(X_i)$ within the region represented by $T(X_i)$ is defined as

$$\begin{aligned} f(X_i) \approx{}& f(\bar{X}_i) + (\bar{X}_i - X_i)^T J(\bar{X}_i) \\ &+ \frac{1}{2}(X_i - \bar{X}_i)^T H(\bar{X}_i)(X_i - \bar{X}_i) \end{aligned} \tag{5}$$

where $\bar{X}_i$ is the middle data example of the region defined by $T(X_i)$, $J(\bar{X}_i)$ is the gradient of $\bar{X}_i$, and $H(\bar{X}_i)$ is the Hessian matrix [37] of $\bar{X}_i$.

According to the second-order Taylor approximation of $f(X_i)$, we have the following lemma which approximates the probability of a data example falling into a leaf node, denoted as $P(T(X_i))$.

Lemma 1: Given a rare tree T and a data example $X_i \in X$, the approximate probability of X_i falling into a leaf node is

$$\begin{aligned} P(T(X_i)) &= \int_{x \in T(X_i)} f(x)dx \\ &\approx f(\bar{X}_i)V(T(X_i)) + \frac{1}{24}\sum_{j=1}^{d} \frac{\partial^2 f(\bar{X}_i)}{\partial \bar{X}_i^{j\,2}} \nabla_j{}^2 V(T(X_i)) \end{aligned} \tag{6}$$

where ∇_j is the jth range of the region that $T(X_i)$ represents.

Proof: After applying the Taylor approximation of (5) on $f(x)$, the right part of (6) can be written as

$$\begin{aligned} \int_{x \in T(X_i)} f(x)dx \approx{}& f(\bar{X}_i)V(T(X_i)) \\ &+ \sum^{d} \int \frac{\partial f(\bar{X}_i)}{\partial \bar{X}_i^j}\left(x^j - \bar{X}_i^j\right)dx \\ &+ \sum_{k,j=1}^{d} \int \frac{\partial^2 f(\bar{X}_i)}{\partial \bar{X}_i^k \partial \bar{X}_i^j}\left(x^k - \bar{X}_i^k\right)\left(x^j - \bar{X}_i^j\right)dx. \end{aligned}$$

Since $(x^k - \bar{X}_i^k)$ is an odd function in the region of the leaf node, $\sum_{j=1}^{d} \int (\partial f(\bar{X}_i)/\partial \bar{X}_i^j)(x^j - \bar{X}_i^j)dx$ equals to zero. In addition, $(x^k - \bar{X}_i^k)(x^j - \bar{X}_i^j)$ is an odd function in this region as well if $k \neq j$. Therefore, $\sum_{k,j=1}^{d} \int [(\partial^2 f(\bar{X}_i))/(\partial \bar{X}_i^k \partial \bar{X}_i^j)](x^k - \bar{X}_i^k)(x^j - \bar{X}_i^j)dx$ can be written as

$$\begin{aligned} &\sum_{j=1}^{d} \int \frac{\partial^2 f(\bar{X})}{\partial \bar{X}_i^{j\,2}}\left(x^j - \bar{X}_i^j\right)^2 dx \\ &= \frac{1}{6}\sum_{j=1}^{d} \frac{\partial^2 f(\bar{X}_i)}{\partial \bar{X}_i^{j\,2}} \cdot \frac{\nabla_j{}^2}{4} V(T(X_i)). \end{aligned}$$

Then, Lemma 1 holds. ∎

According to Lemma 1, we have the following lemma on the expectation of the estimated density around a leaf node.

Lemma 2: The expectation of the estimated density is

$$\mathbb{E}\left[\hat{f}(X_i)\right] \approx f(\bar{X}_i) + \frac{1}{24}\sum_{j=1}^{d} \frac{\partial^2 f(\bar{X}_i)}{\partial \bar{X}_i^{j\,2}} \nabla_j{}^2.$$

Proof: The expectation of the estimated density, denoted as $\mathbb{E}[\hat{f}(x)]$, can be simply defined as the ratio between the expected probability of data examples within the leaf node, denoted as $[(\mathbb{E}[|\mathcal{C}_{T(X_i)}|])/n]$, and the volume of the leaf

node, denoted as $V(T(X_i))$. Further, the expected probability $[(\mathbb{E}[|\mathcal{C}_{T(X_i)}|])/n]$ equals to $nP(T(X_i))$. Therefore, we have

$$\mathbb{E}\Big[\hat{f}(X_i)\Big] = \frac{nP(T(X_i))}{V(T(X_i))}. \tag{7}$$

After applying Lemma 1 to the above equation, Lemma 2 is proven. ∎

According to Lemma 2 and the second-order Taylor approximation, we have the following theorem which captures the upper bound of the density estimation bias, that is, β^2.

Theorem 1: The upper bound of the density estimation bias is

$$\begin{aligned}\beta^2 \le & \left(\frac{1}{24}\sum_{j=1}^{d}\frac{\partial^2 f(\bar{X}_i)}{\partial \bar{X}_i^{j^2}}\nabla_j{}^2\right)^2 + \frac{1}{4}\sum_{j=1}^{d}\left(\frac{\partial f(\bar{X}_i)}{\partial \bar{X}_i^j}\nabla_j\right)^2 \\ & + \frac{1}{64}\sum_{k,j=1}^{d}\left(\frac{\partial^2 f(\bar{X}_i)}{\partial \bar{X}_i^k \partial \bar{X}_i^j}\nabla_k\nabla_j\right)^2 + |C_1C_2| + \frac{1}{4}|C_1C_3| \\ & + \frac{1}{8}|C_2C_3|\end{aligned}$$

where $C_1 = (1/24)\ \sum_{j=1}^{d}\ [(\partial^2 f(\bar{X}_i))/(\partial \bar{X}_i^{j^2})]\ \nabla_j{}^2$, $C_2 = \sum_{j=1}^{d}\ |([\partial f(\bar{X}_i)]/[\partial \bar{X}_i^j])\nabla_j|$, and $C_3 = \sum_{k,j=1}^{d}\ |([\partial^2 f(\bar{X}_i)]/[\partial \bar{X}_i^k \partial \bar{X}_i^j])\nabla_k\nabla_j|$.

Proof: After applying Lemma 2 to the equation $\beta^2 = (\mathbb{E}[\hat{f}(X_i)] - f(X_i))^2$, we have

$$\beta^2 \approx \big(C_1 + f(\bar{X}_i) - f(X_i)\big)^2. \tag{8}$$

Recall that $C_1 = (1/24)\sum_{j=1}^{d}([\partial^2 f(\bar{X}_i)]/[\partial \bar{X}_i^{j^2}])\nabla_j{}^2$, which is given in Theorem 1. Using the second-order Taylor approximation to substitute $f(X_i)$, the above equation can be written as the following:

$$\beta^2 \approx \left(C_1 - (X_i - \bar{X}_i)^T J(\bar{X}_i) - \frac{1}{2}(X_i - \bar{X}_i)^T H(\bar{X}_i)(X_i - \bar{X}_i)\right)^2.$$

Expanding all the terms in $J(\bar{X}_i)$ and $H(\bar{X}_i)$ of the above equation, we have

$$\begin{aligned}\beta^2 \approx & \left(C_1 - \sum_{j=1}^{d}\frac{\partial f(\bar{X}_i)}{\partial \bar{X}_i^j}(X_i^j - \bar{X}_i^j)\right. \\ & \left. - \frac{1}{2}\sum_{k,j=1}^{d}\frac{\partial^2 f(\bar{X}_i)}{\partial \bar{X}_i^k \partial \bar{X}_i^j}(X_i^k - \bar{X}_i^k)(X_i^j - \bar{X}_i^j)\right)^2.\end{aligned} \tag{9}$$

Expanding the quadratic term of the above equation, we have

$$\begin{aligned}\beta^2 \approx & C_1^2 + \sum_{j=1}^{d}\left(\frac{\partial f(\bar{X}_i)}{\partial \bar{X}_i^j}(X_i^j - \bar{X}_i^j)\right)^2 \\ & + \frac{1}{4}\sum_{k,j=1}^{d}\left(\frac{\partial^2 f(\bar{X}_i)}{\partial \bar{X}_i^k \partial \bar{X}_i^j}(X_i^k - \bar{X}_i^k)(X_i^j - \bar{X}_i^j)\right)^2 \\ & - 2C_1\sum_{j=1}^{d}\frac{\partial f(\bar{X}_i)}{\partial \bar{X}_i^j}(X_i^j - \bar{X}_i^j) \\ & - C_1\sum_{k,j=1}^{d}\frac{\partial^2 f(\bar{X}_i)}{\partial \bar{X}_i^k \partial \bar{X}_i^j}(X_i^k - \bar{X}_i^k)(X_i^j - \bar{X}_i^j) \\ & + \left(\sum_{j=1}^{d}\frac{\partial f(\bar{X}_i)}{\partial \bar{X}_i^j}(X_i^j - \bar{X}_i^j)\right. \\ & \left. \times \sum_{k,j=1}^{d}\frac{\partial^2 f(\bar{X}_i)}{\partial \bar{X}_i^k \partial \bar{X}_i^j}(X_i^k - \bar{X}_i^k)(X_i^j - \bar{X}_i^j)\right).\end{aligned}$$

After further analysis, we have

$$\begin{aligned}\beta^2 \le & \left(\frac{1}{24}\sum_{j=1}^{d}\frac{\partial^2 f(\bar{X}_i)}{\partial \bar{X}_i^{j^2}}\nabla_j{}^2\right)^2 + \frac{1}{4}\sum_{j=1}^{d}\left(\frac{\partial f(\bar{X}_i)}{\partial \bar{X}_i^j}\nabla_j\right)^2 \\ & + \frac{1}{64}\sum_{k,j=1}^{d}\left(\frac{\partial^2 f(\bar{X}_i)}{\partial \bar{X}_i^k \partial \bar{X}_i^j}\nabla_k\nabla_j\right)^2 + |C_1C_2| + \frac{1}{4}|C_1C_3| \\ & + \frac{1}{8}|C_2C_3|.\end{aligned}$$

Recall $C_2 = \sum_{j=1}^{d}\ |([\partial f(\bar{X}_i)]/[\partial \bar{X}_i^j])\nabla_j|$ and $C_3 = \sum_{k,j=1}^{d}\ |([\partial^2 f(\bar{X}_i)]/[\partial \bar{X}_i^k \partial \bar{X}_i^j])\nabla_k|$. Then, Theorem 1 has been proven. ∎

This theorem indicates that the length of each dimension that forms the region, represented by the leaf node $T(X_i)$, is a key factor in determining the bias. The bias of a rare tree can be reduced by a more suitable region of each leaf node. In this paper, the dip test used to choose a split value and dimension makes a suitable region of the leaf node, especially for the one that contains the rare class. In other words, the deviation between $f(X_i)$ and $\hat{f}(X_i)$ is reduced by the dip test.

Next, we introduce the following theorem which shows the upper bound of the density estimation variance, that is, σ^2.

Theorem 2: $\sigma^2 \le (\theta^2/[n^2\mathrm{Var}(V(\ell(x)))])$.

Proof: Based on (1) and (2), the estimated density of the data example X_i can be rewritten as $\hat{f}(x) = (|\mathcal{C}_{T(X_i)}|/nV(T(X_i)))$. Recall that $T(X_i)$ denotes the leaf node where X_i falls. Applying the above equation to σ^2, we have $\sigma^2 = (1/n^2)\mathrm{Var}(|\mathcal{C}_{T(X_i)}|/V(T(X_i)))$. Since the size of each leaf node is less than a threshold θ, we have $\sigma^2 \le (\theta^2/[n^2\mathrm{Var}(V(T(X_i)))])$. Therefore, we have proven Theorem 2. ∎

This theorem indicates that the variance of the rare tree as a density estimator depends on three factors, that is, the number of data examples in the input dataset, the volume of the leaf node, and the threshold of the leaf node size. The variance increases as the threshold θ increases, and it decreases as the number of data examples in the input dataset or the volume of the leaf node increases. For a specific application, the number of data examples and the volume of the leaf node are fixed. The variance only depends on the threshold of the leaf node size which we suggest to set based on the dataset (refer to Section IV-A for more details).

B. Rare-Class Extraction

Given a rare tree T built over the input dataset X, in this section, we describe the rare-class extraction step of Fast-RCM. In general, we first search all the data examples using the constructed rare tree. Then, we rank all the data examples according to their searching results over the rare tree,

and select a few high-ranked data examples as a rare-class candidate set. Finally, we refine the candidate set in order to accurately extract the rare class. We introduce our ranking method in Section IV-B1, and then describe the rare-class extraction process in Section IV-B2.

1) Ranking the Data Examples: As described in the above section, data examples of the rare class have two dominating features: 1) high estimated density and 2) short searching path length. Naturally, the combination of those two features can be used to discriminate the data examples of a rare class from the remaining data examples of the major classes. Therefore, we define a ranking function using those two dominating features, where data examples of the rare class would have higher ranking scores while in contrast data examples of the major class would have lower ranking scores. This ranking function is defined as follows.

Definition 7 (Ranking Function): Given a data example $X_i \in X$ and a rare tree T, the ranking score of X_i, denoted as $r(X_i)$, is defined as

$$r(X_i) = \frac{\alpha}{h(X_i)} + \beta \hat{f}(X_i)$$

where α and β are the two constants and $\alpha + \beta = 1$, $h(X_i)$ is the searching path length (Definition 5), and $\hat{f}(X_i)$ is the estimated density (Definition 4).

Since data examples of the major class have low estimated density and long searching path length, it is likely that they tend to have lower ranking scores. On the contrary, since data examples from the rare class have high estimated density and short searching path, it is likely that these data examples tend to have higher ranking scores. However, in the extreme case that the rare classes are overlapped with the major classes, data examples from the dense areas of the major classes tend to have higher ranking scores as well. As a result, we cannot accurately extract the rare class if we simply discriminate the highly ranked data examples as the rare classes.

2) Extraction: As aforementioned, in the overlapped case, we cannot accurately extract the rare classes by simply discriminating the high-scored data examples as a rare class. To address this issue, with the goal of extracting a more accurate rare class, we first filter a large percent of data examples from the major classes according to their ranking scores. Then, we refine and extract the rare class from the remaining data examples. Algorithm 3 shows the detailed steps of rare-class extraction. Specifically, we extract a rare class as follows.

1) In lines 3–5, we calculate the ranking scores of each data example, and select the top γ data examples as the rare-class candidate set, where γ is a positive integer.
2) In line 6, we use a specific clustering algorithm to automatically cluster the rare-class candidate set. In this paper, we use a simple yet effective clustering algorithm, k-means. We choose the k-means algorithm simply because of its time efficiency.
3) In line 7, we choose the clusters with large data size and small variance from the clustering results as the rare classes.

In the above steps of rare-class extraction, γ is a key threshold and a suitable value of γ will lead to both high recall and high precision of the rare class. γ is suggested to be larger than the size of the classes, which guarantees that the candidate set potentially contains sufficient data examples for rare-class extraction. Since in the RCM scenario, rare classes are usually hidden in a dataset with over 99% data examples from the major classes, we suggest to set $\gamma \geq |X| \times 1\%$. Moreover, in Section V-B2, we will experimentally show the effectiveness of the Fast-RCM by varying γ.

Algorithm 3 RareClassExtraction

1: **Input:** The unlabelled dataset X and the rare tree T
2: **Output:** the rare classes
3: $R \leftarrow \{R_i | \forall X_i \in X, R_i = r(X_i)\}$ ▷ Calculate the ranking scores for all data examples
4: $t \leftarrow$ the γ-th largest value in R
5: $C_0 \leftarrow \{X_i | \forall X_i \in X, R_i \geqslant t\}$ ▷ Select the top γ data examples
6: $C_1 \leftarrow$ clustering results of C_0 ▷ Clustering analysis
7: $C \leftarrow$ proper clusters from C_1 ▷ Select clusters with large dataset size and small variance
8: **return** C

C. Time Complexity

Assuming the procedure is provided with sorted date examples, the worst-case time complexity of the tree construction process is $\mathcal{O}(dn \log(n/\theta))$, where θ is the threshold of the node size and d is the dimensionality. For the rare-class extraction step, the worst-case time complexity of searching each data example is $\mathcal{O}(n \log(n/\theta))$ and the time complexity of the ranking process is $\mathcal{O}(n \log n)$. Hence, the overall time complexity of Fast-RCM is in $\mathcal{O}((d+1)n \log(n/\theta) + n \log n)$.

V. Experimental Evaluation

In this section, we first introduce the datasets used in our experiments. Then, we discuss how to determine two important parameters for Fast-RCM: 1) the threshold θ and 2) the filtering size γ. After that, we evaluate the performance of Fast-RCM on both synthetic datasets and real-world datasets. For the performance evaluation, we verify the effectiveness and efficiency of the Fast-RCM from two aspects: 1) the accuracy comparison in terms of the I-score of the rare classes (i.e., the harmonic mean of precision and recall of the rare classes) and 2) the runtime comparison.

A. Datasets

1) Synthetic Datasets: We choose the generative model for synthetic datasets such that it exhibits the challenging properties that we focus on in RCM. Overall, we generate 18 synthetic datasets: ten synthetic datasets with varying dataset size, denoted as $DD_1, DD_2, \ldots, DD_{10}$ and eight datasets with varying dataset dimensionality, denoted as $DS_1, DS_2, \ldots, DS_8$. Those synthetic datasets satisfy that:

1) the dataset size n vary from 102 000 to 1 002 000 with an incremental 100 000;
2) the dimensionality of the synthetic datasets ranges from 1 to 40 with an incremental 5;

TABLE I
REAL-WORLD DATASETS

Database	Dataset	d	m	n	Rare Classes	
					$\#y$	n_c
UCI	Breasts	30	2	222	1	10
	Iris	4	3	105	0	5
	Wine	13	3	178	2	10
	Digits	16	3	1659	0	10
	KDD99	41	2	7673	1	979
MOOC	911calls	6	2	134,331	1	1575
MCD	DSAA	4	3	56,470	1	272

3) the *pdf* of the rare class is Gaussian with a small standard deviation;
4) the *pdf* of the major class is Gaussian with a large standard deviation;
5) the rare class has 2000 data examples in low-dimensional datasets (with less than or equal to five dimensions), and 3000 data examples in high-dimensional datasets (with more than five dimensions).

2) Real-World Datasets: The real-world datasets used in our experiments are collected from three databases: 1) UCI database [39]; 2) Montgomery County database (MCD); and 3) MOOC database. Table I shows the detailed information of the real-world datasets, where n is the number of data examples in the dataset, d is the dimensionality of the dataset, m is the number of different classes in the dataset, $\#y$ is the label of the rare class, and n_r is the number of data examples of the objective rare class.

From the UCI database [39], we select five datasets: 1) Breasts; 2) Iris; 3) Wine; 4) Digits; and 5) KDD99. These datasets are from various application domains, such as medical, biology, computer science, and cybersecurity. Many existing RCM and RCD work also utilize these datasets in their experiments [3], [4], [7], [8], [12], [26], [31]. Especially, these five datasets are subsampled for creating RCM scenarios.

In addition to the above five datasets, we employ two other real-world datasets from the Montgomery County and MOOC databases, respectively.

From the MCD, we select one dataset, CALLS.[1] This dataset contains three kinds of emergency (911) calls: 1) fire call; 2) traffic call; and 3) EMS call. These emergency calls were collected from the Montgomery County between December 10, 2015 and November 20, 2016. Each emergency call has eight features: 1) longitude; 2) description; 3) zipcode; 4) title; 5) timestamp; 6) township; 7) address; and 8) *dummy variable*. Fig. 5 illustrates the data distribution of the number of calls in CALLS. From Fig. 5, we observe that the number of emergency calls in Montgomery in the days near January 23, 2016 significantly deviated from the normal level. For more details, a major blizzard produced up to 3 ft (91 cm) of snow in parts of the Mid-Atlantic and Northeast United States from January 23, 2016 to January 24, 2016. Based on this observation, we manually label the emergency calls from January 23, 2016 to January 24, 2016 as the rare class.

MOOC[2] is an online course platform aiming at unlimited participation and open access via the Web. From MOOC, we select a real-word course dataset DSAA[3] of the online course "data structures and algorithms." DSAA contains the records of 50 304 registered students among which only 272 students received the course certification. Each record of a registered student has four features: 1) video time; 2) post number; 3) reply number; and 4) quiz score. The 272 students who received the final course certifications are regarded as the rare class in RCM scenario.

[1]http://montcoalert.org/gettingdata/
[2]https://en.wikipedia.org/wiki/Massive_open_online_course

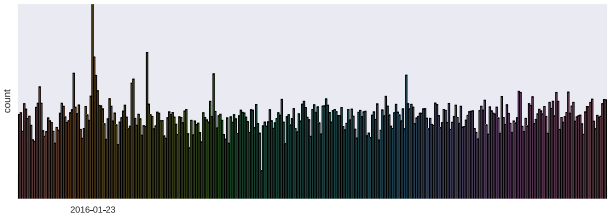
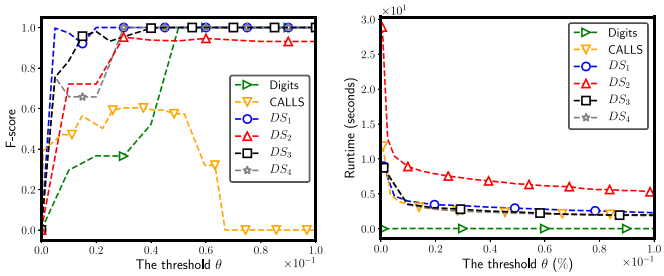


Fig. 5. Distribution of the number of the emergency calls. The peak value near January 23, 2016 reveals the extreme weather.



(a) (b)

Fig. 6. (a) Effectiveness and (b) efficiency under different θ.

B. Parameter Selection

In this section, we discuss how to determine two important parameters for Fast-RCM: 1) the terminate threshold θ for the rare tree construction and 2) the filtering size γ for rare-class extraction.

1) Selection of θ: The threshold θ used in Fast-RCM (refer to Algorithm 2) represents the maximum number of data examples that can be contained in a leaf node. We first discuss how to determine θ.

Settings: We determine θ by analyzing the running results of Fast-RCM with different values of θ. To this end, we select four synthetic datasets: DS_1, DS_2, DS_3, and DS_4, and two real-world datasets: *Digits* and *DSAA*. Then, for each of these datasets, we set γ to the number of rare examples from the dataset and run Fast-RCM by varying θ. Especially, we vary the value of θ from $n \times 0.5\%$ to $n \times 10\%$, where n denotes the number of examples from the dataset.

Results and Findings: We show the running results of Fast-RCM on the six datasets in Fig. 6, where the horizontal ordinate represents the ratio between θ and n. To be specific, Fig. 6(a) shows the F-score of the discovered rare class and Fig. 6(b) shows the runtime. First, Fast-RCM achieves quite high F-score of the discovered rare class with θ ranging from $n \times 2\%$ to $n \times 10\%$. This is because the termination signal θ

[3]https://github.com/HaiQW/DATA/tree/master/DSAA

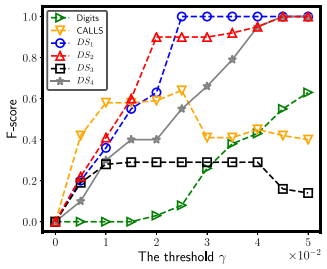


Fig. 7. Effectiveness of varying filtering size γ.

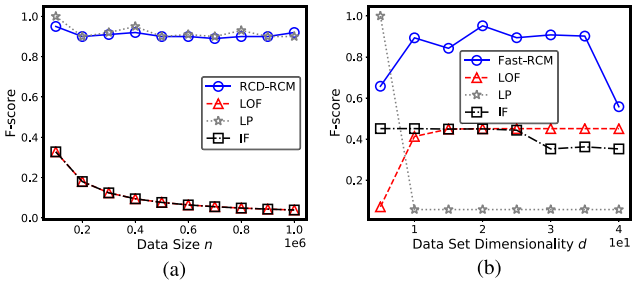


Fig. 8. F-score by varying n and d on the synthetic datasets. (a) Size. (b) Dimensionality.

controls the splitting process of the dataset and a proper value of θ may reorganize the given dataset better. Second, the runtime of Fast-RCM decreases as θ increases. This is because the time complexity of searching examples over the rare tree is $n \log(n/\theta)$.

In summary, the above running results tell that Fast-RCM achieves a high F-score with a small value of θ and requires a low runtime cost with a large value of θ. Therefore, aiming at finding the balance between the effectiveness and time efficiency, we suggest to set $\theta = n \times 5\%$.

2) Selection of γ: The parameter γ represents the number of examples selected to the candidate set (refer to Algorithm 3). Here, we discuss how to determine γ.

Settings: Similar to θ, we determine γ through analyzing the effectiveness of Fast-RCM on six datasets: DS_1, DS_2, DS_3, DS_4, *Digits*, and *DSAA*. To this end, for each of these datasets, we set $\theta = n \times 5\%$ and run Fast-RCM on the dataset by varying γ.

Results and Findings: We show the F-score of the discovered rare class in Fig. 7, where the horizontal ordinate represents the ratio between γ and the dataset size n. Note that we do not analyze the time efficiency in this experiment since that the runtime of refining the candidate set is negligible as compared to the whole time consumption. From Fig. 7, we observe that the F-score of the discovered rare class increases with the increasing of γ when γ is less than $n \times 5\%$. Also, there exist some special cases, for example, on the CALLS dataset, the F-score of the discovered rare class fluctuates within a narrow range. For another example, on the synthetic dataset DS_4, the F-score of the discovered rare class decreases when γ is larger than $n \times 4.5\%$. Those unexpected results might lie in the reason that there is a large overlap between the rare classes and the major classes. We did not report the F-score for $\gamma \geq n \times 5\%$ because the F-score has no rules or may suddenly drop to zero. This phenomenon means that Fast-RCM does not work for a large value of γ.

From the above analysis, we conclude that Fast-RCM can mine out a more accurate rare class if we set γ to an appropriate value. Moreover, since the size of the rare class is relatively small as compared to the rest, a small value of γ will make Fast-RCM work properly. In our experiment, we suggest to set $\gamma = n \times 5\%$.

C. Accuracy Comparison

After discussing how to determine the parameters, we evaluate the accuracy of Fast-RCM on both real-world datasets and synthetic datasets. To be specific, we compare the accuracy of Fast-RCM with FRANK [4], IF [14], LOF [13], and label propagation (LP) [40] using the F-score of the discovered rare class. FRANK [4] is a state-of-the-art RCE algorithm that transforms RCE to local community detection; IF [14] is a tree-based outlier detection algorithm; LOF [13] is a standard outlier detection algorithm; and LP [40] is a classical semisupervised learning algorithm.

Settings: We configure the experiment of the tested algorithms as follows.

1) For Fast-RCM, we set $\theta = n \times 5\%$ and set $\gamma = n \times 10\%$. Note that on KDD99, we specifically set $\gamma = n \times 10\%$ since it has a slightly large number of data examples from the rare class.
2) For FRANK, we run each of the three RCD algorithms (NNDM [7], CLOVER [8], and FRED [27]) to select different data examples from the rare class as three seeds. Then, we run FRANK three times using different seeds and report the average F-score of FRANK.
3) For LP, we use the same representative data example used by FRANK as seeds.
4) For LOF, we set the parameter k to the ground-truth number of the data examples in the rare class.
5) For IF, we construct 100 tree estimators, each of which is built upon 1000 randomly sampled examples.

Results and Findings: We report the running results of the tested algorithms on the synthetic datasets and real-world datasets in Figs. 8 and 9, respectively. First, we observe that Fast-RCM outperforms all the tested algorithms in most cases. Second, LOF did not mine out any data examples from the rare class even if we optimized the parameter k on Wine and Breasts. This might be because the rare class in the dataset is overlapped with major classes. We do not report the F-score of FRANK on CALLS as it ran out of memory. Third, although LP has the best F-score on Iris and FRANK has the best F-score on Breast, they have three disadvantages: 1) they need prior information; 2) they perform poorly in the remaining datasets; and 3) they are time consuming as shown in the following experiments.

In summary, from the above experimental analysis, we can conclude that: 1) Fast-RCM achieves the satisfactory performance on both synthetic and real-world datasets; 2) compared with LP and FRANK, Fast-RCM is a more proper

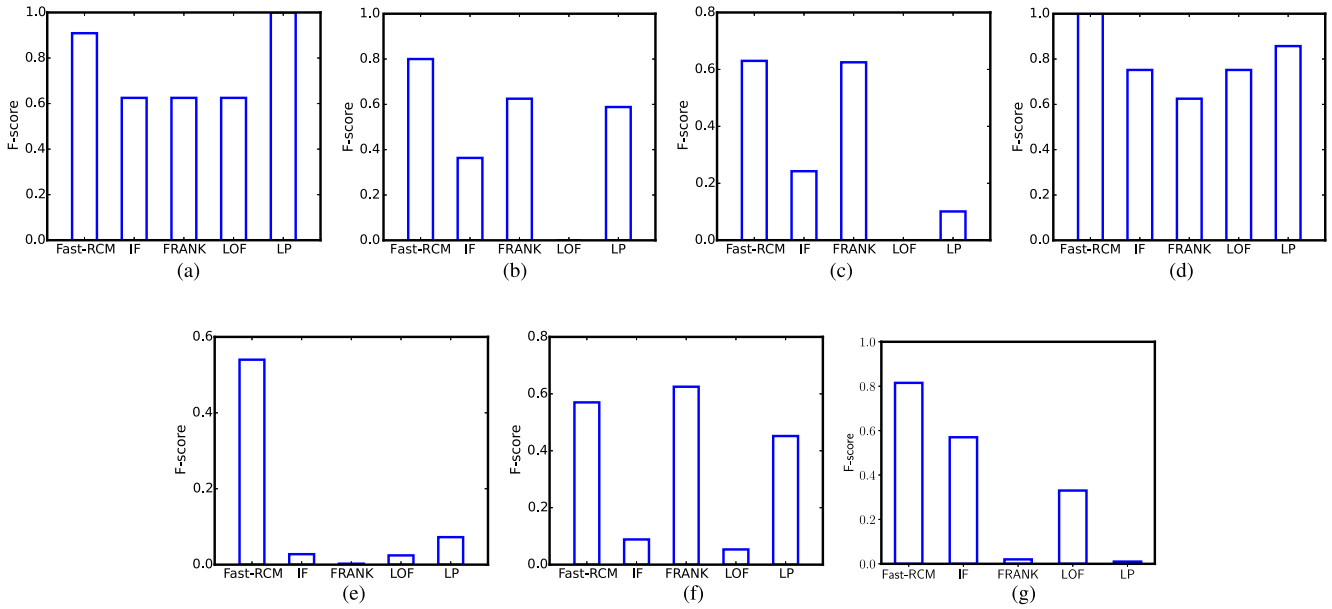


Fig. 9. F-score achieved by different algorithms under multiple real-world datasets. (a) Iris. (b) Wine. (c) Breasts. (d) Digits. (e) CALLS. (f) DSAA. (g) KDD99.

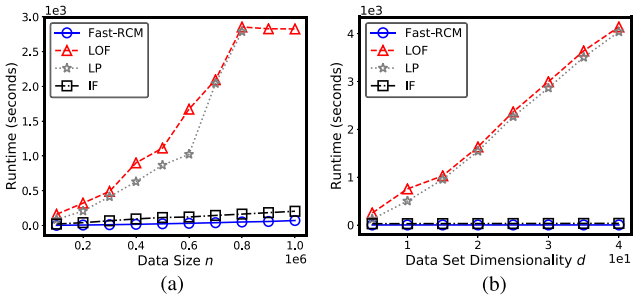


Fig. 10. Runtime for varying n and d. (a) Size. (b) Dimensionality.

TABLE II
RUNTIME (SECONDS) OF THE TESTED ALGORITHMS ON VARIOUS DATASETS

Synthetic Datasets by Varying Dataset Size					
Dataset	Fast-RCM	LOF	FRANK	IF	LP
DS_1	**2.22**	160.53	*	20.19	85.94
DS_2	**6.17**	321.45	*	40.53	208.05
DS_3	**10.92**	490.03	*	67.57	414.02
DS_4	**16.61**	900.35	*	94.26	630.08
DS_5	**24.22**	1113.63	*	114.33	867.77
DS_6	**31.19**	1672.66	*	122.05	1023.93
DS_7	**39.88**	2095.59	*	144.31	2042.12
DS_8	**50.03**	2854.43	*	163.52	2787.61
DS_9	**57.36**	2828.87	*	184.45	*
DS_{10}	**71.13**	2826.64	*	204.50	*
Synthetic Datasets by Varying Dataset Dimensionality					
Dataset	Fast-RCM	LOF	FRANK	IF	LP
DD_1	**1.64**	251.72	*	27.52	122.87
DD_2	**1.65**	755.11	*	28.83	502.29
DD_3	**1.71**	1028.94	*	29.97	954.39
DD_4	**1.71**	1632.47	*	30.79	1542.61
DD_5	**1.76**	2364.81	*	31.78	2264.84
DD_6	**1.82**	2995.25	*	32.64	2864.15
DD_7	**1.86**	3639.23	*	33.62	3505.18
DD_8	**1.90**	4139.55	*	34.83	4038.65

RCM solution as it requires no prior information; and 3) Fast-RCM can potentially find the rare class of interests in real applications.

D. Runtime Comparison

In this experiment, we evaluate the runtime complexity of Fast-RCM on the synthetic datasets.

Setting: We compare the runtime of five tested algorithms, Fast-RCM, IF, FRANK, LOF, and LP [40], with scaled n and d. We configure this experiment as follows.

1) For Fast-RCM, we set $\theta = n \times 5\%$ and set $\gamma = n \times 5\%$.
2) For FRANK, we run NNDM to select representative data examples of the rare class as seeds. Then, we run FRANK using these seeds and report its runtime.
3) For LP, we use the same seeds used by FRANK.
4) For LOF, we set the parameter k to the ground-truth number of the data examples in the rare class.
5) For IF, we construct 100 tree estimators, each of which is built upon 1000 randomly sampled examples.

Results and Findings: We report the runtime of the tested algorithms in Table II and illustrate the runtime behavior in Fig. 10. First, we observe that Fast-RCM is faster than IF, FRANK, LOF, and LP on all the tested datasets. We do not show the runtime of FRANK and LP in several cases in Table II as they did not finish running within 24 h. Second, in Fig. 10(b), we see linear runtime behavior with increasing n for Fast-RCM and IF and quadric runtime behavior for FRANK, LOF, and LP. Third, in Fig. 10(b), we see that the runtime growth, with an increasing d, is nearly zero for Fast-RCM and IF, and is linear for LOF and LP.

In summary, based on the above runtime analysis, we conclude that: 1) Fast-RCM is approximately five times faster than that of the state-of-the-art methods and 2) Fast-RCM has an approximately linear time complexity with respect to dataset

size n and dimensionality d. This linear time complexity is achieved by using the rare tree to reorganize the input dataset and rank data examples of the dataset. Thus, the Fast-RCM can be applied to large-scale datasets in reality to discover interesting patterns.

E. Summary of Experimental Results

In summary, Fast-RCM achieves significant advantage in accuracy, and is effective to extract interesting patterns of rare classes in both synthetic and real-word datasets under the unsupervised setting. For example, Fast-RCM can detect the natural disaster happening in Montgomery County from the CALLS dataset. The experimental results are summarized as follows.

1) From the experiments of parameter selection, we show that Fast-RCM can mine out a more accurate rare class if the threshold θ is within the range from $n \times 2\%$ to $n \times 10\%$ and the filtering size γ is appropriate. We suggest to set $\theta = n \times 5\%$ and set $\gamma = n \times 5\%$.
2) In the experiments of accuracy comparison, Fast-RCM achieves a higher accuracy than that of the state-of-the-art methods in most cases. We conclude that without any prior information, Fast-RCM is effective to extract rare classes from both synthetic and real-world datasets.
3) In the experiments of runtime comparison, Fast-RCM is approximately five times faster than that of the state-of-the-art methods. In addition, we observe that Fast-RCM has an approximately linear time complexity with respect to the dataset size n and the data dimensionality d. Thus, Fast-RCM can be applied to large-scale datasets in reality to discover interesting patterns.

VI. Conclusion

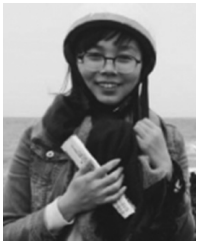
In this paper, we investigate the unsupervised RCM problem, which is first such an attempt to our knowledge. For the unsupervised RCM problem, we propose an efficient algorithm Fast-RCM. Fast-RCM achieves approximately linear complexity with respect to dataset size and dimensionality, and is approximately five times faster than that of the state-of-the-art approaches. Given an unlabeled dataset, Fast-RCM first builds a rare tree for the input dataset, and then extracts the rare class by traversing this rare tree. Leveraging multiple real-world and synthetic datasets, we verify the effectiveness and efficiency of the Fast-RCM.

For the next stage, we will focus on online RCM. To this end, we are going to maintain a dynamic rare tree for the streaming dataset. Then, we will try to extract the rare class from the streaming dataset once the dynamic rare tree changes its structure.

References

[1] D. Pelleg and A. W. Moore, "Active learning for anomaly and rare-category detection," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 1073–1080.
[2] S. Bay, K. Kumaraswamy, M. G. Anderle, R. Kumar, and D. M. Steier, "Large scale detection of irregularities in accounting data," in *Proc. ICDM*, 2006, pp. 75–86.
[3] H. Huang, Q. He, J. He, and L. Ma, "RADAR: Rare category detection via computation of boundary degree," in *Proc. Pac.–Asia Conf. Knowl. Disc. Data Min.*, 2011, pp. 258–269.
[4] H. Huang, K. Chiew, Y. Gao, Q. He, and Q. Li, "Rare category exploration," *Expert Syst. Appl.*, vol. 41, no. 9, pp. 4197–4210, 2014.
[5] Z. Liu, H. Huang, Q. He, K. Chiew, and Y. Gao, "Rare category exploration on linear time complexity," in *Proc. DASFAA*, 2015, pp. 37–54.
[6] W. Eberle, J. Graves, and L. Holder, "Insider threat detection using a graph-based approach," *J. Appl. Security Res.*, vol. 6, no. 1, pp. 32–81, 2010.
[7] J. He and J. G. Carbonell, "Prior-free rare category detection," in *Proc. SIAM Int. Conf. Data Min.*, 2009, pp. 155–163.
[8] H. Huang, Q. He, K. Chiew, F. Qian, and L. Ma, "CLOVER: A faster prior-free approach to rare-category detection," *Knowl. Inf. Syst.*, vol. 35, no. 3, pp. 713–736, 2013.
[9] S. Wang, H. Huang, Y. Gao, T. Qian, L. Hong, and Z. Peng, "Fast rare category detection using nearest centroid neighborhood," in *Proc. 18th Asia–Pac. Web Conf. Web Technol. Appl.*, 2016, pp. 383–394.
[10] D. Tu, L. Chen, X. Yu, and G. Chen, "Semisupervised prior free rare category detection with mixed criteria," *IEEE Trans. Cybern.*, vol. 48, no. 1, pp. 115–126, Jan. 2018.
[11] P. Laskov, P. Düssel, C. Schäfer, and K. Rieck, "Learning intrusion detection: Supervised or unsupervised?" in *Proc. Int. Conf. Image Anal. Process.*, 2005, pp. 50–57.
[12] J. He, H. Tong, and J. G. Carbonell, "Rare category characterization," in *Proc. 10th Int. Conf. Data Min.*, 2010, pp. 226–235.
[13] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 93–104, 2000.
[14] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Min.*, 2008, pp. 413–422.
[15] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surveys*, vol. 41, no. 3, p. 15, 2009.
[16] S. Hawkins, H. He, G. Williams, and R. Baxter, "Outlier detection using replicator neural networks," in *Proc. Int. Conf. Data Warehousing Knowl. Disc.*, 2002, pp. 170–180.
[17] D. Barbara, N. Wu, and S. Jajodia, "Detecting novel network intrusions using Bayes estimators," in *Proc. SIAM Int. Conf. Data Min.*, 2001, pp. 1–17.
[18] A. Valdes and K. Skinner, "Adaptive, model-based monitoring for cyber attack detection," in *Proc. Int. Workshop Recent Adv. Intrusion Detect.*, 2000, pp. 80–93.
[19] K.-L. Li, H.-K. Huang, S.-F. Tian, and W. Xu, "Improving one-class SVM for anomaly detection," in *Proc. Int. Conf. Mach. Learn. Cybern.*, vol. 5, 2003, pp. 3077–3081.
[20] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognit.*, vol. 58, pp. 121–134, Oct. 2016.
[21] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Disc. Data Min.*, 1996, pp. 226–231.
[22] S. Guha, R. Rastogi, and K. Shim, "ROCK: A robust clustering algorithm for categorical attributes," in *Proc. 15th Int. Conf. Data Eng.*, 1999, pp. 512–552.
[23] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in *Proc. Pac.–Asia Conf. Knowl. Disc. Data Min.*, 2002, pp. 535–548.
[24] F. E. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.
[25] N. Ye and Q. Chen, "An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems," *Qual. Rel. Eng. Int.*, vol. 17, no. 2, pp. 105–112, 2001.
[26] J. He and J. G. Carbonell, "Nearest-neighbor-based active learning for rare category detection," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 633–640.
[27] Z. Liu, K. Chiew, Q. He, H. Huang, and B. Huang, "Prior-free rare category detection: More effective and efficient solutions," *Expert Syst. Appl.*, vol. 41, no. 17, pp. 7691–7706, 2014.
[28] D. Zhou, K. Wang, N. Cao, and J. He, "Rare category detection on time-evolving graphs," in *Proc. IEEE Int. Conf. Data Min.*, 2015, pp. 1135–1140.
[29] Z. Liu, H. Huang, Q. He, K. Chiew, and L. Ma, "Rare category detection on $O(dN)$ time complexity," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2015, pp. 37–54.

[30] Z. Liu, K. Chiew, L. Zhang, B. Zhang, Q. He, and R. Zimmermann, "Rare category exploration via wavelet analysis: Theory and applications," *Expert Syst. Appl.*, vol. 63, pp. 173–186, Nov. 2016.
[31] J. He, Y. Liu, and R. D. Lawrence, "Graph-based rare category detection," in *Proc. 8th IEEE Int. Conf. Data Min.*, 2008, pp. 833–838.
[32] S. Maurus and C. Plant, "Skinny-dip: Clustering in a sea of noise," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2016, pp. 1055–1064.
[33] C. Zhang, B. E. Mapes, and B. J. Soden, "Bimodality in tropical water vapour," *Quart. J. Roy. Meteorol. Soc.*, vol. 129, no. 594, pp. 2847–2866, 2003.
[34] J. A. Hartigan and P. M. Hartigan, "The dip test of unimodality," *Ann. Stat.*, vol. 13, no. 1, pp. 70–84, 1985.
[35] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, 2001.
[36] S. Hassani, *Mathematical Methods: For Students of Physics and Related Fields*, vol. 720. New York, NY, USA: Springer, 2008.
[37] T. Hastie, R. Tibshirani, and J. H. Friedman, "The elements of statistical learning," in *Data Mining, Inference, and Prediction* (Springer Series in Statistics), 2nd ed. New York, NY, USA: Springer-Verlag, 2009.
[38] D. Zill, W. S. Wright, and M. R. Cullen, *Advanced Engineering Mathematics*, Jones & Bartlett Learn., 2011.
[39] A. Asuncion and D. Newman, "UCI machine learning repository," 2007.
[40] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 321–328.



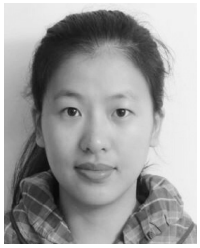
Haiqin Weng received the B.S. degree from the South China University of Technology, Guangzhou, China, in 2014. She is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China, under the supervision of Prof. Q. He.

Her current research interests include data mining and machine learning.

Shouling Ji (M'10) received the first Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, and the second Ph.D. degree in computer science from Georgia State University, Atlanta, GA, USA.

He is a ZJU 100-Young Professor with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China, and a Research Faculty with the School of Electrical and Computer Engineering, Georgia Institute of Technology. His current research interests include big data security and privacy, big data driven security and privacy, adversarial learning, graph theory and algorithms, and wireless networks.

Dr. Ji was the Membership Chair of the IEEE Student Branch at Georgia State from 2012 to 2013. He is a member of ACM.



Changchang Liu received the bachelor's and master's degrees from the University of Science and Technology of China (USTC), Hefei, China (graduated with Guo Moruo Scholarship, the highest honor in USTC), and the Ph.D. degree from the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA.

She is with IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. Her current research interests include Internet of Things security, statistical data privacy, and machine learning.

Ting Wang received the Ph.D. degree in computer science from the Georgia Institute of Technology, Atlanta, GA, USA.

He was a Research Staff Member and a Security Analytic Leader with IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. He is an Assistant Professor of computer science with Lehigh University, Bethlehem, PA, USA. His current research interests include computational privacy, cyber-security analytics, and network science.

Qinming He received the B.S., M.S., and Ph.D. degrees in computer science from Zhejiang University, Hangzhou, China, in 1985, 1988, and 2000, respectively.

He is a Professor with the College of Computer Science and Technology, Zhejiang University. His current research interests include data mining and computing virtualization.

Jianhai Chen (M'13) received the M.S. and Ph.D. degrees in computer science and technology from Zhejiang University (ZJU), Hangzhou, China.

He is currently a Lecturer with the College of Computer Science and Technology, ZJU. He is the Director of ZJU SuperComputing Team and has led ZJU team winning the First Prize of ASC World Supercomputing Contest four times and the Highest Computing Performance Award. His project titled designer won the First Prize of 2018 Xunlei World Blockchain Application Development Contest. His current research interests include cloud computing resource scheduling algorithms, blockchain system security, and high-performance computing parallel application optimization.

Dr. Chen is a member of CCF and ACM.