Rare Category Detection on Time-Evolving Graphs

Dawei Zhou
Arizona State University
davidchouzdw@gmail.com

Kangyang Wang
Arizona State University
wky91916@gmail.com

Nan Cao
IBM T.J. Watson Research
nan.cao@gmail.com

Jingrui He
Arizona State University
jingrui.he@gmail.com

***Abstract*—Rare category detection(RCD) is an important topic in data mining, focusing on identifying the initial examples from rare classes in imbalanced data sets. This problem becomes more challenging when the data is presented as time-evolving graphs, as used in synthetic ID detection and insider threat detection. Most existing techniques for RCD are designed for static data sets, thus not suitable for time-evolving RCD applications.**

To address this challenge, in this paper, we first propose two incremental RCD algorithms, *SIRD* and *BIRD*. They are built upon existing density-based techniques for RCD, and incrementally update the detection models, which provide 'time-flexible' RCD. Furthermore, based on *BIRD*, we propose a modified version named *BIRD*-LI to deal with the cases where the exact priors of the minority classes are not available. We also identify a critical task in RCD named *query distribution*. It aims to allocate the limited budget into multiple time steps, such that the initial examples from the rare classes are detected as early as possible with the minimum labeling cost. The proposed incremental RCD algorithms and various query distribution strategies are evaluated empirically on both synthetic and real data.

***Index Terms*—Rare Category Detection, Time-evolving Graph Mining, Incremental Learning**

I. Introduction

In the era of big data, tremendous amount of data in a variety of areas is being generated at an unprecedented speed. However, it is often the case that, only a small percentage of the data is of interest to us. For example, in synthetic ID detection [13], only a very small number of identities are faked ones generated by mixing the identifying information from multiple sources. Such identities are created for the sole purpose of committing financial fraud. Another example is insider threat detection [4], where only a small number of users in a big organization are malicious insiders, aiming to attack the organization or its employees via sabotage, espionage, etc. The small percentage of data of interest to us is called the minority class, or rare category, since such examples are often self-similar. Due to their rarity nature and the limited budget on querying the labeling oracle, who can provide the true label of any example at a fixed cost, it is difficult to identify examples from such classes via simple random sampling. To address this problem, rare category detection has been proposed to identify the very first example from the minority class, by requesting only a small number of labels from the oracle.

Most, if not all, of existing rare category detection techniques are designed for static data. However, in many real-world applications, the data is evolving over time, so is the minority classes. For example, in synthetic ID detection, each identity may keep updating his/her information over time, such as daily transactions and real-time online banking activities; in insider threat detection, the insiders intentionally change their behavior patterns over time to avoid being caught. For such applications, straight-forward application of existing RCD techniques would be very time-consuming by constructing the models from scratch at each time step. Furthermore, besides the limited budget on querying the labeling oracle, in these applications, it is also critical to detect the initial rare examples as early as possible to avoid further damage.

To address this problem, in this paper, for the first time, we study the problem of incremental RCD. To be specific, we first propose two incremental algorithms, i.e., *SIRD* and *BIRD*, to detect the initial examples from the minority classes under different dynamic settings. The key idea is to efficiently update our detection model by local changes instead of reconstructing it from scratch on the updated data at a new time step, so as to reduce the time cost of redundant and repeating computations. Furthermore, we provide a modified version – *BIRD*-LI, which relaxes the requirement of the exact priors with a soft upper bound for all the minority classes. Finally, we study a unique problem of query distribution in the dynamic settings, which distributes allocated labeling budget to different time steps, and propose five query distribution strategies.

The rest of our paper is organized as follows. In Section II, we briefly review the related work on both RCD and time-evolving graph mining. In Section III, we study incremental RCD and propose three algorithms for addressing different dynamic settings, i.e., *SIRD*, *BIRD* and *BIRD*-LI. Then, in Section IV, we introduce the unique problem of query distribution in the dynamic settings, and propose five strategies for allocating the labeling budget to different time steps. In Section V, we demonstrate our models on both synthetic and real data sets. Finally, we conclude this paper in Section VI.

II. Related Work

A. Rare Category Detection

Lots of technologies have been developed for the problem of RCD in the past few years. [12] proposed a mixture model-based algorithm, which is the first attempt in this area. In [6] [7], the author developed an innovative method to detect rare categories via unsupervised local-density-differential sampling strategy. [3] presented an active learning scheme via exploiting the cluster structure in data sets. In [8], RACH was proposed for rare category characterization by an optimization framework. More recently, in [11], two prior-free methods were proposed in order to address the rare category detection

problem without any prior knowledge. In [16], the authors proposed a framework named MUVIR, which could leverage multiple existing rare-category-detection methods in the multi-view vision. However, few works have been found to address the dynamic-setting rare category detection. In this paper, we study the problem of how to efficiently and incrementally learn from time-evolving graph and effectively detect rare categories over time.

B. Time-evolving Graph Mining

In recent years, more and more researches have been conducting on time-evolving graph mining. For example, in [10], the authors analyzed the properties of the time evolution of real graphs and proposed a forest fire graph-generative model; [2] studied the problem of community evolution and developed a novel method to measure the movement of individuals among communities; in [15], authors proposed a fast proximity tracking method for dynamic graphs; in [9], the authors focused on the difficulties of conversation dynamics and proposed a simple mathematical model in order to generate basic conversation structures; in [5], the authors proposed a new graph-pattern matching algorithm, which can avoid cubic-time computation; [1] raised a divide-and-conquer framework, which could find the k-nearest-neighbors efficiently on large time-evolving graphs; in [14], the authors launched a fast algorithm which could detect the node relationships for localizing anomalous changes in time-evolving graphs. In this paper, we propose several fast-updating RCD methods which could incrementally update the models based on local changes on time-evolving graphs. The correctness of our algorithms is demonstrated by both theoretical proof and experiments.

III. Incremental Rare Category Detection

In this section, we introduce the proposed framework for time-evolving rare category detection.

A. Notation

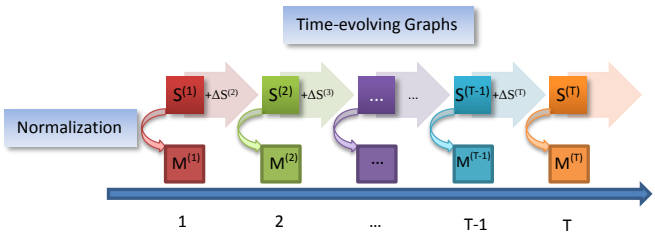


Fig. 1: Time-evolving Graphs

Suppose we are given n unlabeled examples $\{x_1, \cdots, x_n\}$, and observe $m^{(t)}$ updated edges at time step t. We assume $y_i = 1$ corresponds to the majority class with prior $p_1^{(t)}$, and the remaining classes are the minority classes with prior $p_c^{(t)}$. We use capital $S^{(t)}$ to represent the aggregated adjacency matrix and $\Delta S^{(t)}$ to denote the new edges and updated weights that appear at time step t. Specifically, we have $\Delta S^{(t)} = S^{(t)} - S^{(t-1)}$. We use $M^{(t)}$ to denote the normalized aggregated adjacency matrix, which is calculated from $S^{(t)}$.

In the following part of this paper, we use the convention in Matlab to represent matrix elements, e.g., $M^{(t)}(i,j)$ is the element at i^{th} row and the j^{th} column of matrix $M^{(t)}$, and $M^{(t)}(:,j)$ is the j^{th} column of matrix $M^{(t)}$, etc.

B. Static Rare Category Detection

Static RCD refers to a problem of repeatedly selecting examples to be labeled by oracle until all the minority classes in a static data set are discovered. One very successful approach for static RCD is to make use of manifold structure and identify rare category examples. In [7], authors developed a graph-based RCD method named GRADE. In GRADE algorithm, they first construct a pair-wise similarity matrix W' and its corresponding diagonal matrix D, whose elements are the row sum of W'. And then, they calculate the normalized matrix W as follows:

$$W = D^{-1/2} W' D^{-1/2}$$

Based on the normalized pair-wise similarity matrix W, they construct a global similarity matrix A by applying random walk with restart (RWR), which is shown as follows:

$$A = (I_{n\times n} - \alpha W)^{-1}$$

By constructing the global similarity matrix, the changes of local density would become sharper near the boundary of the minority classes. Based on this intuition, GRADE could identify minority classes with much less queries than random sampling. However, the time complexity of calculating the global similarity matrix and finding each example's $(K)^{th}$ nearest neighbor is $O(n^3 + K \cdot n^2)$, which is not efficient enough for time-evolving RCD applications.

C. Dynamic Rare Category Detection

In this subsection, we introduce two fast-updating incremental RCD algorithms (*SIRD* and *BIRD*) for dealing with the RCD problem on time-evolving graphs. These two methods greatly reduce the computation cost for both updating global similarity matrix and finding each example's K^{th} nearest neighbor. To specify this problem, we have the following assumptions: (i) the number of examples is fixed, and only edges change over time; (ii) dataset is imbalanced; (iii) minority classes are not separable from the majority classes.

1) Single Update: We first consider the simplest case: only one self-loop edge (a,a) changes at time step t. In other words, there is only one non-zero element (a,a) in $\triangle S^{(t)}$.

To address this problem, we first introduce Theorem 1 to update the global similarity matrix $A^{(t)}$ more efficiently. Due to page limitation, all proofs in this paper are omitted.

Theorem 1. *The global similarity matrix $A^{(t)}$ at time step t can be exactly updated from global similarity matrix $A^{(t-1)}$ at last time step $t-1$ by the following equation:*

$$A^{(t)} = A^{(t-1)} + \alpha \frac{A^{(t-1)} u v^T A^{(t-1)}}{I + v^T A^{(t-1)} u}$$

where u and v^T are the two vectors decomposed from updating matrix $\delta M^{(t)}$

In our methods, we use an approximate method to calculate two column vectors u and v. The details are described as follows. We firstly assume that the updated edges at time step

t have little impact on the row sum of adjacency matrix M^t when the number of updated edges are extremely smaller than total number of edges. Thus, we have

$$D^{(t)} \cong D^{(t-1)}$$

To normalize $S^{(t)}$ and $S^{(t-1)}$, we have:

$$M^{(t)} = (D^{(t)})^{-1/2} S^{(t)} (D^{(t)})^{-1/2} \quad (1)$$

$$M^{(t-1)} = (D^{(t)})^{-1/2} S^{(t-1)} (D^{(t-1)})^{-1/2} \quad (2)$$

Let Eq. 1 $\ominus$ Eq. 2, we have

$$\Delta M^{(t)} = (D^{(t-1)})^{-1/2} \Delta S^{(t)} (D^{(t-1)})^{-1/2} \quad (3)$$

As $\Delta M^{(t)} = uv^T$, we could easily assign $u = D(:,a)^{-1/2}$ and $v = \Delta S^{(t)}(a,b) D(:,b)^{-1/2}$.

Besides, as the time complexity of constructing a new neighbor information matrix $NN^{(t)}$ is $O(K^{(t)} \cdot n^2)$. We introduce the Theorem 2 to efficiently update $NN^{(t)}$.

Theorem 2. *Suppose there is only one self loop edge (a,a) being updated at time step t. If it satisfies the condition that $\frac{\alpha}{I+v^T A^{(t-1)} u} \leq \frac{\delta_i^{(t-1)}}{A_{i,a}^{(t-1)} \phi_a}$, the first $K^{(t)}$ elements in $NN^{(t)}(i,:)$ are the same as $NN^{(t-1)}(i,:)$.*

The single-update incremental RCD algorithm (*SIRD*) is shown in Algorithm 1. In Step 1 to Step2, we firstly initialize the diagonal matrix D and neighbor information matrix $NN^{(1)}$ at time step 1. In Step 4, let $K^{(t)}$ represents the number of examples in the largest minority class at time step t. Then, from Step 5 to Step 6, we update the global similarity matrix at each time step. Step 7 to Step 9 updates the rows in $NN^{(t)}$, of which the $K^{(t)}$ largest elements are changed. Step 11 to 20 is the query process. First of all, we calculates the class specific a^c in Step 13, which is the largest global similarity to the $k_c^{(th)}$ nearest neighbor. Then, in Step 14, we count the number of its neighbors whose global similarity larger than or equal to a^c, and let n_i^c denote the counts for each example x_i. In Step 16, we calculate the score of each example x_i, which represents the change of local density. At last, we select the examples with the largest score and let them be labeled by oracle. The query process only terminates as long as all the minority classes are discovered.

The efficiency of the updating process for Algorithm 1 is given by the following lemma.

Lemma 1. *The computational cost of updating process at each time step in Algorithm 1 is $O(n^2 + l \cdot K^{(t)} \cdot n)$.*

2) Batch Update: In most real world applications, we may always observe that a batch of edges change at the same period. Specifically, the updated aggregated adjacency matrix $\Delta M^{(t)}$ may have more than one non-zero elements. Hence, $\Delta M^{(t)}$ can not be decomposed into two column vectors, and Theorem 2 could not be applied in this condition. In this part, we introduce Theorem 3 to helps us to update the neighbor information matrix $NN^{(t)}$ when batch of edges are changed.

Theorem 3. *Suppose there are m edges $\{(a^1,b^1), \cdots, (a^m,b^m)\}$ being updated at time step t. The first $K^{(t)}$ elements in $NN^{(t)}(i,:)$ are the same as $NN^{(t-1)}(i,:)$, if it satisfies the condition that*

$$\frac{\alpha}{I + V^T A^{(t-1)} U} \leq \min_{i=1,\ldots,m} \{T_i\}$$

where $T_i = \min\{\frac{\delta_i^{(t-1)}}{A_{i,a^i}^{(t-1)} \phi_{b^i}}, \frac{\delta_i^{(t-1)}}{A_{i,b^i}^{(t-1)} \phi_{a^i}}\}$.

Algorithm 1 *SIRD* Algorithm

Input: $M^{(1)}$, $A^{(1)}$, $\Delta S^{(2)}, \ldots, \Delta S^{(T)}$, $p_c^{(t)}$, α.
Output: The set I of labeled examples

1: Construct the $n \times n$ diagonal matrix D, where $D_{ii} = \sum_{(j=1)}^{n} S^{(1)}$, $i = 1, \ldots, n$.
2: Sort row i of $A^{(1)}$ and saved into $NN^{(1)}(i,:)$, where $i = 1, \ldots, n$.
3: **for** t=2:T **do**
4: Let $K^{(t)} = \max_{c=2}^{C} n \times p_c^{(t)}$.
5: Let column vector $u = D(:,a)^{-1/2}$ and column vector $v = \Delta S^{(t)}(a,a) D(:,a)^{-1/2}$, where $\Delta S^{(t)}(a,a)$ is the non-zeros element in $\Delta S^{(t)}$.
6: Update the global similarity matrix as follows:

$$A^{(t)} = A^{(t-1)} + \alpha \frac{A^{(t-1)} u v^T A^{(t-1)}}{I + v^T A^{(t-1)} u}$$

7: **for** i=1:n **do**
8: Based on Theorem 2, identify whether the first $K^{(t)}$ elements of $NN^{(t)}$ (i,:) is changed. If true, update the first $K^{(t)}$ element in $NN^{(t)}(i,:)$; otherwise, let $NN^{(t)}(i,:) = NN^{(t-1)}(i,:)$.
9: **end for**
10: **end for**
11: **for** c = 2:C **do**
12: Let $k_c = n \times p_c^{(T)}$
13: Find the first k_c element in each row of $NN^{(T)}$. Set a^c to be the largest value of them.
14: Let $KNN^c(x_i, a^c) = \{x | NN^{(T)}(i,j) > a^c\}$, and $n_i^c = |KNN^c|$, where $i = 1, \ldots, n$ and $j = 1, \ldots, n$.
15: **for** index = 1: n **do**
16: For each nodes x_i has been labeled y_i, if $A^{(T)} > a^{y_i}$, $score_j = -\infty$; else, let $score_i = \max_{A^{(T)}(i,j) > \frac{a^c}{index}} (n_i^c - n_j^c)$
17: Select the examples x with largest score to oracle.
18: If the label of x is exact class c, break; else, mark the class that x belongs to as discovered
19: **end for**
20: **end for**

The Batch-update Incremental Rare Category Detection (*BIRD*) is shown in Algorithm 2. Step 1 and Step 2 are the initialization process. Step 3 to 12 updates the global similarity matrix $A^{(t)}$ and neighbor information matrix $NN^{(t)}$. Different from Algorithm 1, Step 5 to Step 8 iteratively updates the global similarity matrix $A^{(t)}$ based on $m^{(t)}$ changed edges. Another difference is that, in Step 10, T is the minimum value of the thresholds calculated from $m^{(t)}$ updated edges. At last, Step 13 to Step 20 is the query process, which is the same as what we have described in Algorithm 1.

The efficiency of batch-edges updating in Algorithm 2 is proved by the following lemma.

Lemma 2. *In Algorithm 2, the computational cost of the updating process at each time step is* $O(m^{(t)}n^2 + l \cdot K^{(t)} \cdot n)$.

Algorithm 2 *BIRD* Algorithm

Input: $M^{(1)}$, $A^{(1)}$, $\Delta S^{(2)}, \ldots, \Delta S^{(T)}$, $p_c^{(t)}$, α.
Output: The set I of labeled examples

Construct the $n \times n$ diagonal matrix D, where $D_{ii} = \sum_{(j=1)}^{n} S^{(1)}$, $i = 1, \ldots, n$.
Sort row i of $A^{(1)}$ and saved into $NN^{(1)}(i,:)$, where $i = 1, \ldots, n$.
for t=2:T **do**
  Let $K^{(t)} = \max_{l=c}^{C} n \times p_c^{(t)}$.
  for m = 1: $m^{(t)}$ **do**
    Let column vector $u = D(:, a^m)^{-1/2}$ and column vector $v = \Delta S^{(t)}(a^m, b^m) D(:, b^m)^{-1/2}$, where $\Delta S^{(t)}(a,a)$ is the non-zeros element in $\Delta S^{(t)}$.
    Update the global similarity matrix as follows:

$$A^{(t)} = A^{(t-1)} + \alpha \frac{A^{(t-1)} u v^T A^{(t-1)}}{I + v^T A^{(t-1)} u}$$

  end for
  for i=1:n **do**
    Based on Theorem 3, identify whether the first $K^{(t)}$ elements of $NN^{(t)}$ (i,:) is changed. If true, update the $K^{(t)}$ element in $NN^{(t)}(i,:)$; otherwise, let $NN^{(t)}(i,:) = NN^{(t-1)}(i,:)$.
  end for
end for
while not all the classes have been discovered **do**
  Calculate n_i for each examples, where $i = 1, \ldots, n$.
  for index = 1: n **do**
    For each nodes x_i has been labeled y_i, if $A^{(T)} > a$, $score_j = -\infty$; else, let $score_i = \max_{A^{(T)}(i,j) > \frac{a}{index}} (n_i - n_j)$
    Select the examples x with largest score to labeling oracle.
    mark the class that x belongs to as discovered.
  end for
end while

D. BIRD *with less information*

In many applications, it may be difficult to obtain the exact priors of all the minority classes. In this subsection, we introduce *BIRD*-LI, a modified version of *BIRD*, which requires only an upper bound p^t for all the minority classes existing at time step t. To be specific, *BIRD*-LI firstly calculates $NN^{(1)}$ and diagonal matrix D at the first, which is the same as *BIRD*. Then, the global similarity matrix $A^{(T)}$ and the neighbor information matrix $NN^{(T)}$ could be updated from the first time step to the time step T. The only difference between *BIRD* and *BIRD*-LI is the size of minority class $K^{(t)}$ is calculated based on an estimated upper bound prior instead of the exact ones for all minority classes. After the updating process, it calculates an overall score for the minority classes and select the examples with the largest overall score to be labeled by oracle. *BIRD*-LI only terminates the loop until all the classes are discovered.

IV. Query Dynamics

A. Query Locating

First of all, we introduce the query locating problem. In real world applications, it could be the case that we are given a series of unlabeled time-evolving graphs $S^{(1)}, S^{(2)}, \ldots, S^{(T)}$, and we need to select an optimal time step T_{opt} for identifying minority class.

Before talking about our methods, let us introduce the two main factors that affect the required number of queries in rare category detection. The first factor is $P(y = 2|x_i)$, which is the probability that example x_i belongs to minority class given the features of x_i. Many works have already studied it before, such as MUVIR [16], GRADE [7] and NNDM [6]. Another factor is the density D_i of x_i. Because when the proportions of minority classes are fixed, the density of minority classes located area would directly impact the hardness of RCD. When the density of neighborhood is higher, it means we need identify rare category examples from a larger set of candidates. Considering the second factor, we introduce the following theorem to estimate density D_i based on the global similarity matrix constructed before.

Theorem 4. *For each example* x_i*, the density of* x_i *is positive correlated with* $D_i^{(t)}$*, where* $D_i^{(t)} = \Sigma_{j=1}^{n} A_{i,j}^{(t)}, i = 1, \cdots, n$.

We let $score^{(t)} = P(y = 2|x_i^{(t)})$, which could be obtained using existing techniques such as MUVIR [16] or GRADE [7]. Under this circumstance, we propose to assign the hardness of identifying minority class at time step t as follows:

$$I^{(t)} = \left\{ k_c \max_{i=1,\ldots,k_c} \frac{score_i^{(t)}}{D_i^{(t)}} \right\}^{-1} \tag{4}$$

where k_c is the number of examples in minority class c.

Let $RS^{(t)}$ denote the number of required queries by random sampling at time step t. And, Let $C = \frac{RS^{(1)} - RS^{(T)}}{T}$. Intuitively, we could achieve optimal solution T_{opt}, when the difference between the "exact" saved number of queries and estimate saved number of queries $C * T_{opt}$ is maximized. The formulation is shown as follows:

$$\max_{t=1,\ldots,T} \frac{I^{(1)} - I^{(t)}}{I^{(1)} - I^{(T)}} \cdot (RS^{(1)} - RS^{(T)}) - C \cdot t \tag{5}$$

B. Query Distribution

In this subsection, we will talk about a more general problem: Query Distribution. In real world applications, it could be the case that the updated graphs come as streams, and we need to allocate our query budget into multiple incoming time steps. So, is there a method to allocate the queries properly into different time steps, and enable us to find the minority class examples with both minimum query budget and minimum time budget?

To further study the query dynamic problem, we propose 5 potential strategies for the query distribution problem: *S1*

Allocate all the budget at the first time step; *S2* Allocate all the budget at the last time step; *S3* Allocate all the budget at time step T_{opt}; *S4* Allocate the query budget evenly into different time steps; *S5* Allocate the query budget into different time steps following exponential distribution, such as $e^{-\alpha t}$.

For query distribution problem, we propose Algorithm 3. Different from the query process of Algorithm 2, in Step 3, we need to apply a strategy to calculate the certain budget $B^{(t)}$ for time step t. If we have not found the minority class within $B^{(t)}$ at time step t, then we go to the next time step. The overall algorithm stops either when minority classes are discovered or there is no budget to use.

We compare the performance of these five strategies with both synthetic data set and real data set in Section V.

Algorithm 3 *Query Distribution Algorithm*

Input: S,$M^{(1)}$,$A^{(1)}$,$NN^{(1)}$,$\Delta S^{(2)},\ldots,\Delta S^{(T)}$,$p^{(t)}$,$\alpha$.
Output: The set I of labeled examples and the L of their labels

1: **for** t = 1:T **do**
2: Let $K^{(t)} = \max_{l=c}^{C} n \times p_l^{(t)}$.
3: Calculate $B^{(t)}$ as Given Strategy S.
4: Calculate $NN^{(t)}$ as described in Algorithm 2.
5: **while** not all the classes have been discovered **do**
6: Find the $(K^{(t)})^{th}$ element in each row of $NN^{(t)}$. Set a^c to be the largest value of them.
7: Let $KNN^c(x_i, a^c) = \{x | NN^{(T)}(i,j) > a^c\}$, and $n_i^c = |KNN^c|$, where $i = 1,\ldots,n$ and $j = 1,\ldots,n$.
8: **for** index = 1: $B^{(t)}$ **do**
9: For each nodes x_i has been labeled y_i, if $A^{(T)} > a^{y_i}$, $score_j = -\infty$; else, let $score_i = \max_{A^{(T)}(i,j) > \frac{a^c}{index}} (n_i^c - n_j^c)$
10: Select the examples x with largest score to labeling oracle.
11: If the label of x is exact class c, break; else, mark the class that x belongs to as discovered
12: **end for**
13: **end while**
14: If all the minority classes are discovered, break.
15: **end for**

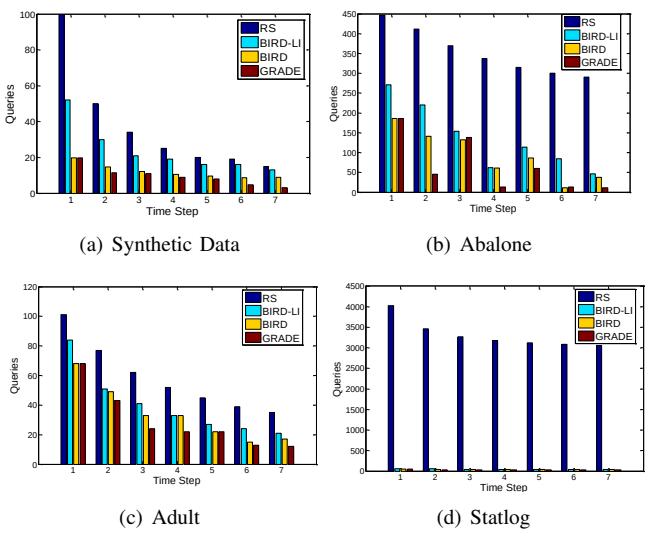
V. Experiments

A. Effectiveness

Name	n	d	m	Largest-Class	Smallest-Class
Abalone	4177	8	5	56.93%	0.41%
Adult	48842	14	2	1.30%	98.70%
Statlog	58000	9	6	79.16%	0.04%

TABLE I: Real Datasets

First of all, we demonstrate the effectiveness upon 1000 synthetic datasets and 3 real data sets. For the synthetic datasets, we generate 1000 synthetic datasets, and each of them contains 5000 examples, two classes. And, we initialize the priors of the minority classes as 1% and increase these priors by 1% in each time step. For the real datasets, it



(a) Synthetic Data (b) Abalone

(c) Adult (d) Statlog

Fig. 2: Effectiveness

is really difficult to find proper real datasets satisfying the scenario setting of both rare category detection and time-evolving graphs. In this case, we find 3 real datasets which meet the scenario of rare category detection. The details of these 3 real datasets are summarized in Table 1. And then, we generate additional 6 time-evolving graphs in the latter time steps. For these time-evolving graphs, we let the proportion of one certain minority class increase by 1% and simultaneously let the proportion of the majority class decrease by 1% in each time step. Fig. 2(a) shows the comparison results of 4 different methods: random sampling(RS), *BIRD*, *BIRD*-LI and GRADE. Notice that *BIRD* and *BIRD*-LI perform the query process upon the approximate aggregated adjacency matrix, while GRADE is performed on the exact adjacency matrix at each time step. Besides, we input *BIRD*-LI with a much looser prior upper bound, e.g., we input 5% as the upper bound instead of using the exact prior of 1%. And then, we perform the same comparison experiments on 3 real data sets, which is shown in Fig. 2(b), Fig. 2(c) and Fig. 2(d). In general, we have the following observations: (i) both *BIRD* and *BIRD*-Li outperform random sampling in any conditions; (ii) all of these 4 methods perform better when prior of minority class is getting larger; (iii) *BIRD* gives a comparable performance as GRADE; (iv) *BIRD*-LI is quite robust and requires only a few more queries than BIRD in most cases.

B. Efficiency of Batch Update

For both *BIRD* and GRADE, the most time consuming step is updating the global similarity matrix $A^{(t)}$ and neighbor information matrix $NN^{(t)}$ in each time step. In this subsection, we report the running time of updating $A^{(t)}$ and $NN^{(t)}$ from an initial time step to the second time step. To better visualize the performance, we run the experiment on an increasing size of graph, i.e., from 500 examples in graph to 1000 examples in graph. And for each certain size, we have 100 identical-setting datasets. Each point in Fig. 3 is computed based on the average value of the 100 datasets in identical settings. As

we talked before, the computation cost of GRADE is $O(n^3)$, and our method only costs $O(n^2)$. From Fig. 3, we can see the difference of running time is largely increasing over time. The difference is limited when the number of examples is 500. However, when the size of graph goes to 10000, the running time of BIRD is 6.227 seconds, while the running time of GRADE is 41.41 seconds, which is 7 times of BIRD. Moreover, the difference will be extraordinarily enlarged again when we run algorithms on a series of time steps. We ran the experiments with Matlab 2014a on a workstation with CPU 3.5 GHz 4 processors, 256 GB memory and 2 T disk space.

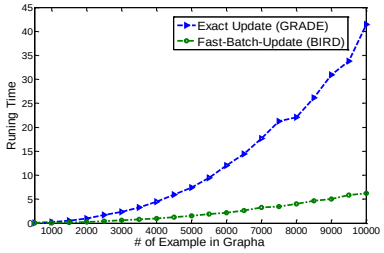
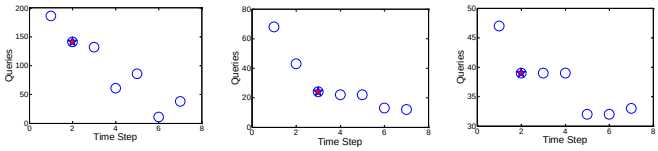


Fig. 3: Efficiency

C. Query Dynamics

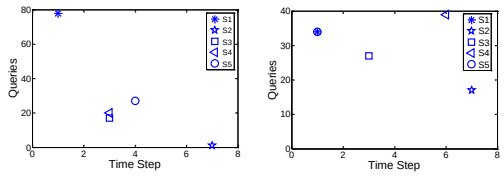
In this subsection, we perform the results of query locating and query distribution. In Fig. 4, we apply the query locating methods on 3 real datasets. As the proportion is increasing over time, the labeling request is decreasing in general. Besides, we also observe that T_{opt} is always located at left bottom of each graph, which meets our ALAP and AEAP intuitions.



(a) Abalone (b) Adult (c) Statlog

Fig. 4: Query Locating

Furthermore, by applying Algorithm 3, we perform the results of 5 different strategies on one binary-class synthetic dataset and one binary-class real dataset, i.e. Adult. In both Fig. 5(a) and F.g 5(b), we observe that Strategy $S1$ is always located at the left top of the figure, which hold the time optimal; Strategy $S2$ is always located at the right bottom of the figure, which hold the budget optimal; Strategy $S3$ is always located at the left bottom of the figure, which leverage both the time and budget factor. All of these 3 observations meet our intuitions.



(a) Synthetic Dataset (b) Real Dataset(Adult)

Fig. 5: Query Distribution

VI. Conclusion and Future Work

In this paper, we mainly focus on the problem of how to efficiently and incrementally identify under-represented rare category examples from time-evolving graphs. To the best of our knowledge, we are the first attempt on RCD under these dynamic settings. The major contribution of this paper could be summarized as follows: (1)A novel problem setting of rare category detection on time-evolving graphs; (2)Two fast incremental RCD algorithms in dynamic settings, i.e., *SIRD* and *BIRD*, and the analysis regarding their efficiency and effectiveness; (3)Fast update algorithm *BIRD*-LI for the cases where the exact priors of minority classes are unknown; (4)Preliminary study on query distribution, which is unique in the dynamic settings; (5)Extensive experiments on both synthetic and real data sets. In our future works, we will continue studying RCD under dynamic settings, especially the problem of query distribution. A very interesting and challenging research direction is how to allocate optimal query budget and detect rare categories in real time settings.

References

[1] L. Akoglu, R. Khandekar, V. Kumar, S. Parthasarathy, D. Rajan, and K.-L. Wu. Fast nearest neighbor search on large time-evolving graphs. In *ECML PKDD*. Springer, 2014.

[2] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *ACM SIGKDD*, 2006.

[3] S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *ICML*, 2008.

[4] W. Eberle, J. Graves, and L. Holder. Insider threat detection using a graph-based approach. *Journal of Applied Security Research*, 2010.

[5] W. Fan, X. Wang, and Y. Wu. Incremental graph pattern matching. *TODS*, 2013.

[6] J. He and J. G. Carbonell. Nearest-neighbor-based active learning for rare category detection. In *NIPS*, 2007.

[7] J. He, Y. Liu, and R. Lawrence. Graph-based rare category detection. In *ICDM*, 2008.

[8] J. He, H. Tong, and J. Carbonell. Rare category characterization. In *ICDM*, 2010.

[9] R. Kumar, M. Mahdian, and M. McGlohon. Dynamics of conversations. In *ACM SIGKDD*, 2010.

[10] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *ACM SIGKDD*, 2005.

[11] Z. Liu, K. Chiew, Q. He, H. Huang, and B. Huang. Prior-free rare category detection: More effective and efficient solutions. *Expert Systems with Applications*, 2014.

[12] D. Pelleg and A. W. Moore. Active learning for anomaly and rare-category detection. In *NIPS*, 2004.

[13] C. Phua, V. Lee, K. Smith, and R. Gayler. A comprehensive survey of data mining-based fraud detection research. *ICICTA*, 2010.

[14] K. Sricharan and K. Das. Localizing anomalous changes in time-evolving graphs. In *ACM SIGMOD*, 2014.

[15] H. Tong, S. Papadimitriou, S. Y. Philip, and C. Faloutsos. Proximity tracking on time-evolving bipartite graphs. In *SIAM SDM*, 2008.

[16] D. Zhou, J. He, K. Candan, and H. Davulcu. Multi-view rare category detection. In *IJCAI*, 2015.